VII ALIO/EURO

Workshop on Applied Combinatorial Optimization

Porto, Portugal, May 4 – 6, 2011

Proceedings Book

Institutional Support:

ALIO

EURO
The Association of European Operational Research Societies

U.PORTO
FACULDADE DE CIÊNCIAS
UNIVERSIDADE DO PORTO

INESCPORTO
LABORATÓRIO ASSOCIADO

Apdio
associação portuguesa de investigação operacional

Published by: ALIO-EURO 2011
May 4-6 2011
http://www.dcc.fc.up.pt/ALIO-EURO-2011/

Sponsors:

– Cámara Municipal do Porto

– Fundação para o Desenvolvimento Social do Porto

– Porto Cidade de Ciência

– Universidade do Porto

– Fundação para a Ciência e a Tecnologia

Institutional support:

– Asociación Latino-Iberoamericana de Investigación Operativa

– Association of European Operational Research Societies

– Instituto de Engenharia de Sistemas e Computadores do Porto

– Faculdade de Ciências da Universidade do Porto

– Associação Portuguesa de Investigação Operacional

Welcome Note

Dear Conference Participant,

It is our great pleasure to welcome you to Porto and to the 7^{th} edition of the ALIO-EURO workshop in Applied Combinatorial Optimization.

Porto is a city full of tradition and contrasting modernity. House of some of the most awarded contemporary architects in the world, here you can find modern vibrating buildings side by side with walls that preserve centuries of History. You can make a toast (always with Port Wine) at the modernist concert hall building of Casa da Música (House of the Music) or at the old cellars in Vila Nova de Gaia, on the left bank of river Douro. You can explore the renowned contemporary art museum of Serralves and enjoy its wonderful gardens. A stroll in the city park, towards the seaside and the mouth of river Douro is also a must for those who like walking. Plenty of interesting activities that we expect will contribute for good moments of leisure after the workshop.

In ALIO-EURO 2011 there will be presentations covering a wide range of subjects – over 70 high quality presentations and 4 keynote talks by distinguished researchers. We are very grateful to all authors for contributing to the success of the workshop. We hope that this selection will provide each of you with opportunities to learn something new, to discuss and exchange research ideas with other colleagues and to start new collaborations.

The high quality of the program is also due to the strong engagement of the Program Committee and Cluster Organizers in a thorough reviewing process. To all of them we address our sincere acknowledgment.

To conclude, we are grateful to the Faculty of Sciences of the University of Porto for hosting the workshop and for providing all the required facilities, and to all sponsors for the financial support provided.

We wish you a pleasant and fruitful stay in Porto.

The Organizing Committee

Local Organizing Committee:

Ana Viana (chair), Instituto Politécnico do Porto / INESC Porto

A, Miguel Gomes, Faculdade de Engenharia da Universidade do Porto / INESC Porto

João Pedro Pedroso, Faculdade de Ciências da Universidade do Porto / INESC Porto

Maria Teresa Costa, Instituto Politécnico do Porto / INESC Porto

Program Committee:

Ana Viana (Portugal)

Andrés Weintraub (Chile)

A. Miguel Gomes (Portugal)

Celso C. Ribeiro (Brazil)

Chris Potts (UK)

Hector Cancela (Uruguay)

Horacio Yanasse (Brazil)

Irene Loiseau (Argentina)

J. Valério de Carvalho (Portugal)

João Pedro Pedroso (Portugal)

M. Grazia Speranza (Italy)

Margarida Vaz Pato (Portugal)

Maria Teresa Costa (Portugal)

Maria Urquhart (Uruguay)

Olivier Hudry (France)

Paolo Toth (Italy)

Rafael Martí (Spain)

Ramon Alvarez-Valdes (Spain)

Richard F. Hartl (Austria)

Rolf Möhring (Germany)

TABLE OF CONTENTS

Plenary Talks

Session 1A – Energy I

Session 1B – Multiobjective Evolutionary Algorithms

Session 1C – Graph Theory

Session 2A – Cutting and Packing I

Session 2B – Metaheuristics Frameworks

Session 2C – Lot Sizing and Scheduling

Session 3A – Cutting and Packing II

Session 3B – Matheuristics

Session 3C – Applications of Combinatorial Optimization I

Session 4A – Cutting and Packing III

Session 4B – Scheduling and Metaheuristics I

Session 4C – Vehicle Routing Problem

Session 5A – Energy II

Session 5B – Mathematical Programing

Session 5C – Health

Session 6A – Logistics I

Session 6B – Scheduling and Metaheuristics II

Session 6C – Telecomunications

Session 7A – Logistics II

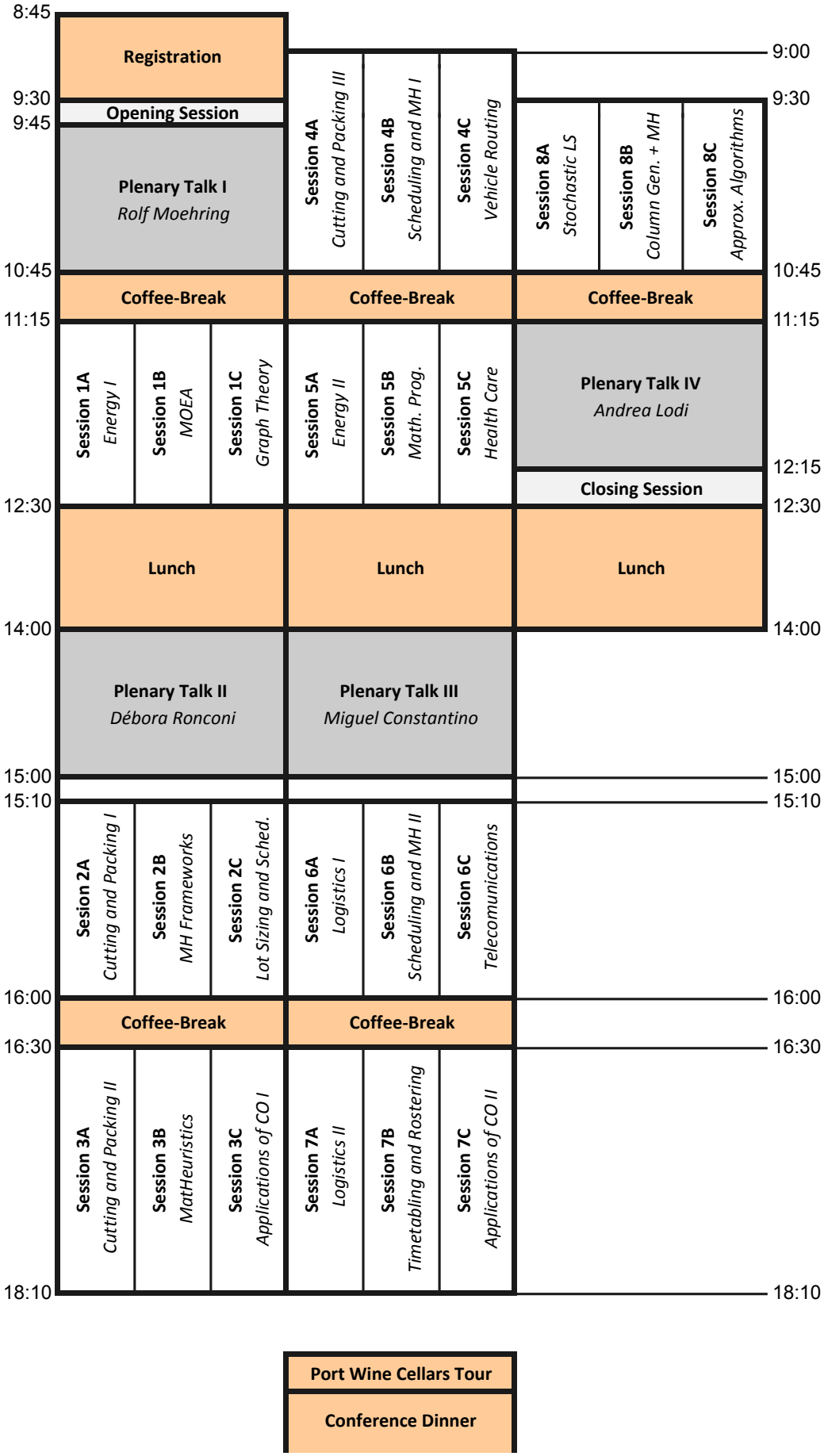
Session 7B – Timetabling and Rostering

Session 7C – Applications of Combinatorial Optimization II

Session 8A – Stochastic Local Search

Session 8B – Column Generation and Metaheuristics

Session 8C – Approximation Algorithms


8:45
Registration
9:30
Opening Session
9:45
Plenary Talk I
Rolf Moehring
10:45
Coffee-Break
11:15
Session 1A Energy I
Session 1B MOEA
Session 1C Graph Theory
12:30
Lunch
14:00
Plenary Talk II
Débora Ronconi
15:00
15:10
Sesion 2A Cutting and Packing I
Session 2B MH Frameworks
Session 2C Lot Sizing and Sched.
16:00
Coffee-Break
16:30
Session 3A Cutting and Packing II
Session 3B MatHeuristics
Session 3C Applications of CO I
18:10
Session 4A Cutting and Packing III
Session 4B Scheduling and MH I
Session 4C Vehicle Routing
Coffee-Break
Session 5A Energy II
Session 5B Math. Prog.
Session 5C Health Care
Lunch
Plenary Talk III
Miguel Constantino
Session 6A Logistics I
Session 6B Scheduling and MH II
Session 6C Telecomunications
Coffee-Break
Session 7A Logistics II
Session 7B Timetabling and Rostering
Session 7C Applications of CO II
9:00
9:30
Session 8A Stochastic LS
Session 8B Column Gen. + MH
Session 8C Approx. Algorithms
10:45
Coffee-Break
11:15
Plenary Talk IV
Andrea Lodi
12:15
Closing Session
12:30
Lunch
14:00
15:00
15:10
16:00
16:30
18:10
Port Wine Cellars Tour
Conference Dinner

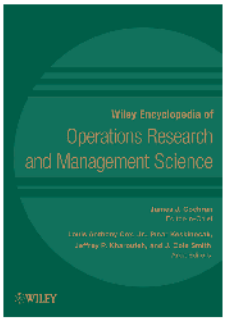
Key Operations Research titles from Wiley-Blackwell

Wiley Encyclopedia of Operations Research and Management Science

Edited by James J. Cochran, Louis Anthony Cox, Jr., Pinar Keskinocak, Jeffrey P. Kharoufeh, and J. Cole Smith

Hardback 8-volumes 6408 pages 2011
ISBN 978-0-470-40063-0

Read this reference work online at: www.wileyonlinelibrary.com/ref/eorms

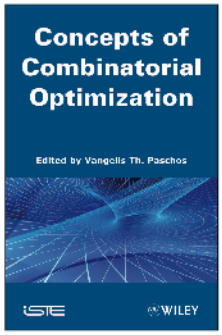


Combinatorial Optimization

Edited by Vangelis Th. Paschos

Hardback 3-volume set 1264 pages
2010 ISBN 978-1-84821-146-9

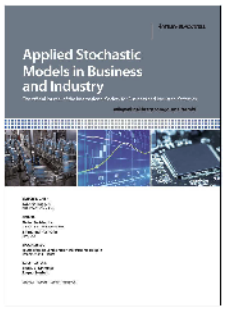
Read this book online at www.wileyonlinelibrary.com



Applied Stochastic Models in Business and Industry

Edited by: Fabrizio Ruggeri

Read this journal online at: www.wileyonlinelibrary.com/journal/asmb



Statistica Neerlandica

Edited by: P. H. Franses

Read this journal online at: www.wileyonlinelibrary.com/journal/stan

International Transactions in Operational Research

Edited by: Celso C. Ribeiro

Read this journal online at: www.wileyonlinelibrary.com/journal/itor

System Dynamics Review

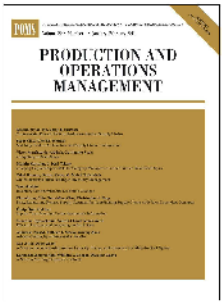
Edited by: Brian Dangerfield

Read this journal online at: www.wileyonlinelibrary.com/journal/sdr

Production and Operations Management

Edited by: Kalyan Singhal

Read this journal online at: www.wileyonlinelibrary.com/journal/poms



Journal of Business Logistics

Edited by: Matthew Waller and Stanley Fawcett

Read this journal online at: www.wileyonlinelibrary.com/journal/jbl

WILEY

Discover our full portfolio on Wiley Online Library

WILEY ONLINE LIBRARY wileyonlinelibrary.com

Routing in Graphs with Applications to Logistics and Traffic

Rolf Möhring *

* TU Berlin

Traffic management and routing in logistic systems are optimization problem by nature. We want to utilize the available street or logistic network in such a way that the total network "load" is minimized or the "throughput" is maximized. This lecture deals with the mathematical aspects of these optimization problems from the viewpoint of network flow theory and scheduling. It leads to flow models in which–in contrast to static flows–the aspects of "time" and "congestion" play a crucial role.

We illustrate these aspects on several applications:

1. Traffic guidance in rush hour traffic (cooperation with ptv).
2. Routing automated guided vehicles in container terminals (cooperation with HHLA).
3. Ship Traffic Optimization for the Kiel Canal (cooperation with the German Federal Water- ways and Shipping Administration).

All these applications benefit from new insights into routing in graphs. In (1), it is a routing scheme that achieves traffic patterns that are close to the system optimum but still respect certain fairness conditions, while in (2) it is a very fast real-time algorithm that avoids collisions, deadlocks, and other conflicts already at route computation. Finally, (3) uses techniques from (2) and enhances them with special purpose scheduling algorithms.

Recent Developments in Optimization Methods for Scheduling Problems

Debora P. Ronconi *

* Department of Production Engineering, EP-USP, University of São Paulo
Av. Prof. Almeida Prado, 128, Cidade Universitária, 05508-900, São Paulo SP, Brazil
dronconi@usp.br

In this talk, the combinatorial optimization scheduling problem will be addressed. A few approaches of exact and heuristic nature developed for different variants of scheduling problems will be described to illustrate the vitality of the topic.

Since the seminal paper by Johnson [4], scheduling problems have received significant attention, particularly in recent years with several publications each year. In general words, the scheduling problem consists of the allocation of resources to tasks over time, considering the physical restrictions of the process while optimizing one or more objectives. Resources can be machines in a workshop, processing units in a computing environment, runways at an airport, and so on; while tasks may be operations in a production process, landings at an airport, or executions of computer programs, just to name a few. A task may have a distinct due date, priority or release date. According to Baker [1], to classify the major scheduling models it is necessary to characterize the configuration of resources and the behavior of tasks. For instance, a model may contain one resource type or several resource types. In addition, if the set of tasks available for scheduling does not change over time, the system is called static, in contrast to cases in which new tasks arise over time, where the system is called dynamic. Generally speaking, the scheduling of jobs is a very complex problem due to its combinatorial nature and, amongst the combinatorial optimization problems, it can be classified as one of the most difficult problems. An overview of scheduling models can be found in [5].

In most theoretical scheduling papers, simple measures of performance have been applied, such as, for example, the completion time of the last job on the last machine, known as makespan. In general, the considered criteria are regular, i.e. nondecreasing with the completion time. Among them, we can mention the total tardiness criterion, whose difficulty arises from the fact that tardiness is not a linear function of completion time. On the other hand, scheduling problems involving not regular measures based on both earliness and tardiness costs have also been addressed in many recent studies. This type of problem became important with the advent of the just-in-time (JIT) concept, where early or tardy deliveries are highly discouraged. A practical example can be found in the chemical industry, where different products can be made through the same process and must be mixed as close as possible to a given instant in time to prevent their deterioration. Comprehensive reviews can be found in [2] and [3].

Due the good performance of optimization methods in several problems that appear in industrial settings, this talk will mainly focus on the application and development of optimization methods for job-scheduling problems in different environments. Selected published papers, which comprise problems addressed by the speaker, will be described.

As the solution of practical models is now largely automated by the use of commercial software, we will initially discuss different mixed-integer models that represent a useful scheduling environment: the flowshop problem with no storage constraints aiming to minimize the sum of earliness and tardiness of the jobs (see [8]). The formulation of combinatorial optimization problems such as mixed-integer models opens the possibility of applying different algorithms developed for general and specific problems. Since the pioneering work of Ralph Gomory in the late 1950s, integer programming is one of the fields in operational research that has made the most progress in the past few years. The most popular approaches are cutting planes and enumerations. Within the second approach, we can highlight the branch-and-bound algorithm, which is basically a sophisticated way to perform an enumeration. With the purpose of illustrating the application of this technique to a scheduling problem, a lower bound which exploits properties of the flowshop problem with blocking will be presented (see [6, 7]). In this environment there are no buffers between successive machines, and, therefore, intermediate queues of jobs waiting in the system for their next operations are not allowed. Some examples of blocking can be found in concrete block manufacturing, which does not allow stock in some stages of the manufacturing process.

On the other hand, there are several combinatorial optimization problems that are difficult to solve through the use of methods that are guaranteed to provide an optimal solution. In these cases, heuristic methods are typically used to quickly find solutions that are not necessarily optimal solutions, but are good quality solutions anyway. Due the practical importance of objectives associated with due dates, we will present heuristic approaches that focus on these performance measures. First, a constructive heuristic that explores specific characteristics of the flowshop problem with blocking will be presented [9]. In this case, performance is measured by the minimization of the total tardiness of the jobs. Then a GRASP-based heuristic is proposed, coupled with a path relinking strategy to search for better outcomes. Next, the minimization of the mean absolute deviation from a common due date in a two-machine flowshop scheduling problem will be addressed [11].
An online version of a single machine scheduling problem to minimize total tardiness will also be described. In this problem, orders get to the system randomly. Jobs have to be scheduled without knowledge of what jobs will come afterwards. The processing times and the due dates become known when the order is placed. A customized approximate dynamic programming method will be presented for this problem [10]. This talk will also comment on new research initiatives under development.

References

[1] K.R. Baker, *Introduction to Sequencing and Scheduling*, Addison-Wesley, John Wiley & Sons, New York, 1974.

[2] K.R. Baker and G.D. Scudder, Sequencing with earliness and tardiness penalties: A review, *Operations Research* 38, pp. 22–36, 1990.

[3] V. Gordon, J.M. Proth and C. Chu, A survey of the state-of-art of common due date assignment and scheduling research, *European Journal of Operational Research* 139, pp. 1–25, 2002.

[4] S.M. Johnson, Scheduling in a two-machine flowshop for the minimization of the mean absolute deviation from a common due date, *Naval Research Logistics Quartely* 1, pp. 61-67, 1954.

[5] M. Pinedo, *Scheduling: theory, algorithms, and systems*, Prentice-Hall, New Jersey, 2008.

[6] D.P. Ronconi, A Branch-and-Bound Algorithm to Minimize the Makespan in a Flowshop with Blocking, *Annals of Operations Research* 138, pp. 53-65, 2005.

[7] D.P. Ronconi and V.A. Armentano, Lower Bounding Schemes for Flowshops with Blocking In-Process, *Journal of the Operational Research Society* 52, pp. 1289-1297, 2001.

[8] D.P. Ronconi and E.G. Birgin, Mixed-integer programming models for flowshop scheduling problems minimizing the total earliness and tardiness, in *Just-in-Time Systems*, Y.A. Ríos-Solís and R.Z. Ríos-Mercado (Eds.), Springer Series on Optimization and Its Applications, P.M. Pardalos and Ding-Zhu Du (Series eds.), 2011, to appear.

[9] D.P. Ronconi and L.S. Henriques, Some Heuristic Algorithms for Total Tardiness Minimization in a Flowshop with Blocking, *Omega* 37, pp. 272-281, 2009.

[10] D.P. Ronconi and W.B. Powell, Minimizing Total Tardiness in a Stochastic Single Machine Scheduling Problem using Approximate Dynamic Programming, *Journal of Scheduling* 13, pp. 597–607, 2010.

[11] C.S. Sakuraba, D.P. Ronconi and F. Sourd, Scheduling in a two-machine flowshop for the minimization of the mean absolute deviation from a common due date, *Computers and Operations Research* 36, pp. 60–72, 2009.

Spatial Forest Optimization

Miguel Constantino *

* Centro de Investigação Operacional
Faculdade de Ciências, Universidade de Lisboa
miguel.constantino@fc.ul.pt

Spatial Forest Optimization is concerned with the design of forest landscapes. Forest landscapes evolve along time under the action of opposing forces. Vegetation growth is counterbalanced by natural hazards such as fire and pests, or through human intervention, such as harvesting. In managed forests usually the main objective is to maximize the value of timber harvested. However, other objectives can be considered, such as soil preservation, aesthetic values, biodiversity and wildlife conservation. Landscapes can be intentionally modified in order to accomplish or help to achieve these goals. For modeling purposes, a forest landscape is a region in the plane, composed of a finite number of smaller management units. A finite horizon divided into periods may be considered. Main decisions are, for each unit, either to harvest in some specific period or not harvesting at all. A set of contiguous units with similar characteristics in some time period is called a patch of the forest. The aim of spatial forest optimization is to optimize an objective function while ensuring certain characteristics of some patches.

In this talk we review a few combinatorial optimization problems that arise in the context of spatial forest optimization: One problem is the so-called "harvest scheduling subject to maximum area restrictions"- large harvested patches are forbidden, to prevent erosion and also for aesthetic reasons. Another one consists of selecting a "patch with a minimum required area." Such a patch may represent an old growth region suitable for wildlife habitat. A related problem consists of selecting a (nearly) convex region in the landscape. We introduce a simplified version of this problem and show it can be solved in polynomial time.

On Bilevel Programming and its Implications for Mixed Integer Linear Programming

Andrea Lodi *

* DEIS, Università di Bologna
Viale Risorgimento 2, 40136 Bologna, Italy
andrea.lodi@unibo.it

Bilevel programming is a rich paradigm to express a variety of real-world applications including game theoretic and pricing ones. However, what we are interested in this talk is to discuss the bilevel nature of two of the most crucial ingredients of enumerative methods for solving combinatorial optimization problems, namely *branching* and *cutting*.

Specifically, we discuss a new branching method for 0-1 programs called *interdiction branching* [3] that exploits the intrinsic bilevel nature of the problem of selecting a branching disjunction. The method is designed to overcome the difficulties encountered in solving problems for which branching on variables is inherently weak. Unlike traditional methods, selection of the disjunction in interdiction branching takes into account the best feasible solution found so far.

On the cutting plane side, we examine the nature of the so-called separation problem, which is that of generating a valid inequality violated by a given real vector, usually arising as the solution to a relaxation of the original problem. We show that the problem of generating a maximally violated valid inequality often has a natural interpretation as a bilevel program [2]. In some cases, this bilevel program can be easily reformulated as a single-level mathematical program, yielding a standard mathematical programming formulation for the separation problem. In other cases, no reformulation exists yielding surprisingly interesting examples of problems arising in the complexity hierarchies introduced by Jeroslow [1].

References

[1] R. Jeroslow, The polynomial hierarchy and a simple model for competitive analysis, *Mathematical Programming*, 32:146–164, 1985.

[2] A. Lodi, T.K. Ralphs, G. Woeginger, “Bilevel Programming and Maximally Violated Valid Inequalities", *Technical Report* OR/11/3, DEIS - Università di Bologna.

[3] A. Lodi, T.K. Ralphs, F. Rossi, S. Smriglio, “Interdiction Branching”, *Technical Report* OR/09/10, DEIS - Università di Bologna.

Multi-Objective Evolutionary Algorithms for Reactive Power Planning in Electrical Distribution Systems: A Comparative Case Study

Dulce Costa * Carlos Henggeler Antunes † António Gomes Martins †

* Department of Electrical Engineering, EST Setúbal, IPS
CAMPUS do IPS 2910-761 Setúbal, Portugal
dulce.costa@estsetubal.ips.pt

† DEEC, University of Coimbra
Polo II, 3030-290 Coimbra, Pólo II - Universidade de Coimbra, Portugal
{ch, amartins}@deec.uc.pt

ABSTRACT

Installation of capacitors in radial electrical distribution power systems is a generalized practice used by the utilities mainly to reduce power losses, improve system stability, perform power factor correction and get a better voltage profile. These benefits are associated with the ability of choosing the appropriate locations and capacity of the equipments to be installed. This problem has been extensively researched over the past decades. Nowadays more flexible optimization tools allow for the computation of solutions to more realistic models. This extended abstract shows how Multi-Objective Evolutionary Algorithms (MOEAs) are adequate tools to tackle this problem and provides a comparative study between some distinct approaches. Some modifications are introduced into an MOEA in order to tailor it to the characteristics of the multi-objective mathematical model.

Keywords: Reactive power compensation, Quality of service, Multi-objective models, Evolutionary algorithms

1. INTRODUCTION

Shunt capacitors installed in electrical distribution networks for reactive power compensation generate some positive effects, such as increasing voltage level at the load point, improving voltage regulation when capacitor banks are properly switched, reducing active and reactive power losses, improving system capacity by reducing currents, reducing the need of reinforcement by releasing system capacity. The importance of an adequate reactive power planning is definite, namely due to the growing utilization and dependency on electricity. The FERC report about the August 2003 North American electrical blackout [1], concluded that poor voltage profile and insufficient reactive planning were decisive factors to this incident. In the mid-20th century these devices were generally installed at the head of electrical distribution systems. Several mathematical models and algorithmic approaches have been reported in the literature [2], and the Capacitor Subcommittee of the IEEE Transmission and Distribution Committee has published several bibliographies on this theme until 1980, [3, 4, 5, 6]. The appearance of capacitors with smaller weight/capacity ratio enabled, from technical and economic perspectives, the allocation of compensation also along the feeders of distribution networks. Mainly in the 1990s new algorithms based on heuristic and meta-heuristic search techniques started to be applied: specific heuristics [7, 8], Simulated Annealing [9, 10, 11], Tabu Search [12, 13], Genetic/Evolutionary Algorithms [14, 15, 16]. The problem of the reactive power planning can be stated as identifying the best network locations and the appropriate dimension of capacitors to be installed in order to achieve the network operator's objectives subject to technical, operational and budget constraints. Mathematical model for this problem are generally of combinatorial nature, involving multiple objective functions, real-valued and integer variables, and linear and non-linear relationships.

2. MULTI-OBJECTIVE MATHEMATICAL MODEL

The multi-objective mathematical model has been formulated as a non-linear mixed integer problem considering two objective functions: minimizing investment costs and minimizing active power losses. These objectives are conflicting and of distinct nature. The constraints comprise operational and quality restrictions: voltage limits at each bus, impossibility to locate capacitor banks in some nodes, operational constrains due to the power flow in the system and the need to supply the required load at each node. The main purpose is to characterize a compensation scheme, which consists of a set of capacitors banks to be located in selected network locations, in order to achieve a compromise between active power losses and investment costs while satisfying all constraints. A detailed description of the model objective functions, power flow equations (physical laws in electrical networks) and other constrains can be found in [17].

3. MULTI-OBJECTIVE EVOLUTIONARY ALGORITHMS

Evolutionary Algorithms (EAs) have gained a growing importance to tackle multi-objective models, particularly for hard combinatorial problems, due to their capability of working with a population of individuals (solutions). Since they deal with a population of solutions and the aim is generally the characterization of a Pareto optimal front, EAs endowed with techniques to maintain diversity of solutions present advantages with respect to the use of scalarizing functions as in traditional mathematical programming approaches. A Pareto optimal front can be identified throughout the evolutionary process, which hopefully converges to the true non-dominated front for the problem under study. It must be noticed that, in real-world problems, this is, in general, a potential Pareto optimal front, classified as such because no other solutions dominating it could be found but no theoretical tools exist guaranteeing their true Pareto optimality. EAs can incorporate techniques aimed at guaranteeing the diversity of the Pareto optimal front in order to display the trade-offs between the conflicting objective functions in different regions of the search space. These advantages of using EAs are not just related with the computational effort required but also with the difficulty of using mathematical programming algorithms in most

high-dimensional combinatorial multi-objective problems.

4. CASE STUDY AND RESULTS

An actual Portuguese electrical radial distribution network has been used for a comparative case study. The network topology is displayed in 1. For more detailed information on this network see [17]. This network is located in a rural area and has a particular characteristic: the voltage profile without compensation does not respect the quality voltage limits, so the zero cost solution is not feasible. Therefore, it is necessary to install capacitors to have feasible solutions with respect to the voltage profile constraint. Three well known MOEA have been implemented: MOGA, SPEA and NSGA-II. Moreover, a local search scheme tailored for this problem has been included in NSGA-II to make the most of the problem specificities, namely regarding neighborhood exploration. In this local search scheme, a move leading to a neighbour solution is defined by changing the capacitor location in the network to a neighbour location, or the capacitor type corresponding to a capacity value. 2, 3, 5 and 4 display the set of initial solutions and the Pareto Frontiers obtained with each algorithm. All MOEA converge reasonably well to a set of dispersed non-dominated solutions. However, the front reached with the modified NSGA II totally dominates the other fronts 6. This approach not only increased the number of solutions computed, but also improved the middle front solutions and extended the Pareto front, achieving compromise solutions with lower costs/higher losses, and higher costs/lower losses.

Figure 1: Portuguese radial electrical distribution network.

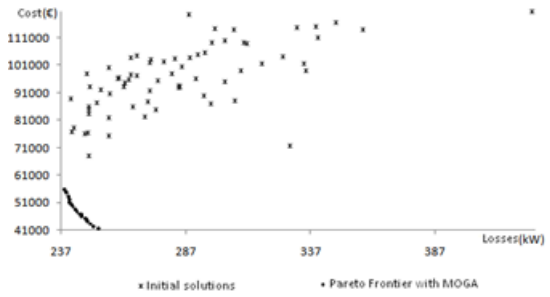


Figure 2: Initial solutions and Pareto Frontier obtained with MOGA.

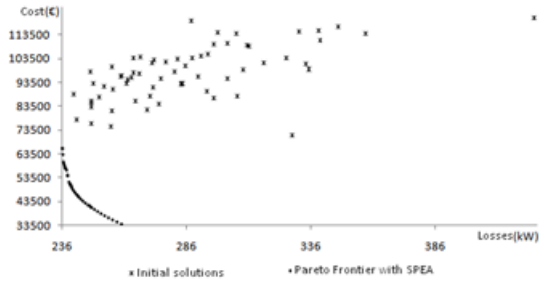


Figure 3: Initial solutions and Pareto Frontier obtained with SPEA.

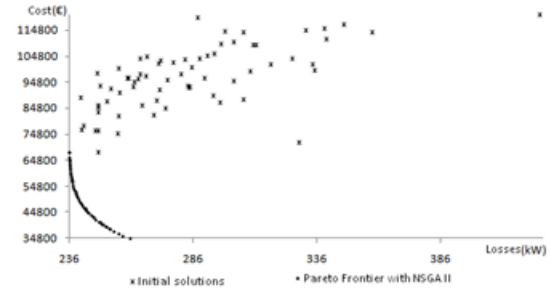


Figure 4: Initial solutions and Pareto Frontier obtained with NSGA II.

5. REFERENCES

[1] F. S. Report, "Principles for efficient and reliable reactive power supply and consumption," Docket No. AD05-1-000, Tech. Rep., 2005.

[2] N. M. Neagle and D. R. Samson, "Loss reduction from capacitors installed on primary feeders," *Transactions of the American Institute of Electrical Engineers, Power Apparatus and Systems PAS*, vol. Part III, no. PAS-75, pp. 950–959, 1956.

[3] I. C. Report, "Bibliography on power capacitors 1967-1970," *IEEE Transactions on Power Apparatus and Systems PAS*, vol. PAS 91, no. 5, pp. 1750–1759, 1972.

[4] ——, "Bibliography on power capacitors 1971-1974," *IEEE Transactions on Power Apparatus and Systems PAS*, vol. PAS 97, no. 4, pp. 1124–1131, 1978.

[5] ——, "Bibliography on power capacitors 1975-1980," *IEEE Transactions on Power Apparatus and Systems PAS*, vol. PAS 102, no. 7, pp. 2331–2334, 1983.

[6] I. V. M. W. G. Report, "Bibliography on reactive power and voltage control," *IEEE Transactions on Power Systems IEEETPS*, vol. 2, no. 2, pp. 361–370, May 1987.

[7] M. M. A. Salama and A. Y. Chikhani, "A simplified network approach to the var control problem for radial distribution systems," *IEEE Transactions on Power Delivery IEEETPD*, vol. 8, no. 3, pp. 1529–1535, 1993.

[8] N. R. J. Shao and Y. Zhang, "A capacitor placement expert system," *International Journal of Engineering Intelligent Systems for Electrical Engineering and Communications*, pp. 105–114, 1994.

[9] Y.-L. C. C.-C. Liu, "Optimal multi-objective var planning using an interactive satisfying method," *IEEE Transactions on Power Systems*, vol. 10, no. 2, pp. 664–670, 1990.

[10] H. Chiang, J. Wang, and O. Cockings, "Optimal capacitor placements in distribution systems part i: A new formula-

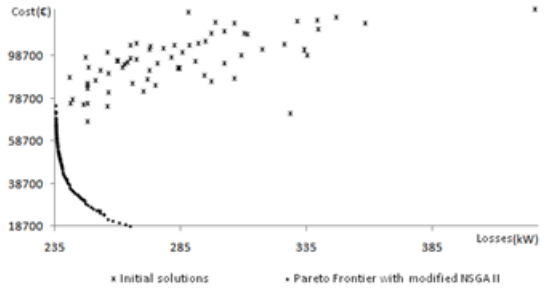


Figure 5: Initial solutions and Pareto Frontier obtained with NSGA II with local search.

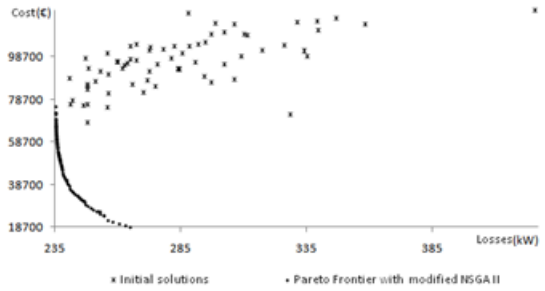


Figure 6: Pareto Frontiers

tion and the overall problem," *IEEE Transactions on Power Delivery*, vol. 5, no. 2, pp. 634–642, 1990.

[11] ——, "Optimal capacitor placements in distribution systems part ii: Solution algorithms and numerical results," *IEEE Transactions on Power Delivery*, vol. 5, no. 2, pp. 643–649, 1990.

[12] Y.-C. H. H.-T. Y. C.-L. Huang, "Solving the capacitor placement problem in a radial distribution system using tabu search approach," *IEEE Transactions on Power Systems*, vol. 11, no. 4, pp. 1868–1873, 1996.

[13] A. G. M. Dulce F. Pires, C. Henggeler Antunes, "A multi-objective model for var planning in radial distribution networks based on tabu search," *IEEE Transactions On Power Systems*, vol. 20, no. 2, pp. 1089–1094, May 2005.

[14] K. Iba, "Reactive power optimization by genetic algorithm," *IEEE Transactions on Power Systems*, vol. 9, no. 2, pp. 685–692, 1994.

[15] G. Levitin, A. Kalyuhny, A. Shenkman, and M. Chertkov, "Optimal capacitor allocation in distribution systems using a genetic algorithm and a fast energy loss computation technique," *IEEE Transactions on Power Delivery*, vol. 15, no. 2, pp. 623–628, 2000.

[16] L. L. J.T. Ma, "Evolutionary programming approach to reactive power planning," *IEE Proceedings - Generation Transmission and Distribution*, vol. 43, no. 4, pp. 365 – 370, July 1996.

[17] A. G. M. Dulce F. Pires, C. Henggeler Antunes, "An nsga-ii approach with local search for a var planning multi-objective problem," Research Report 8/2009, INESC Coimbra, Tech. Rep., 2009.

A new MIP based approach for Unit Commitment in power production planning

Ana Viana * ‡ João Pedro Pedroso * †

* INESC Porto
Campus da FEUP, Rua Dr. Roberto Frias 378, Porto, Portugal
aviana@inescporto.pt

‡ Polytechnic Institute of Engineering of Porto
Rua Dr. António Bernardino de Almeida 431, Porto, Portugal
agv@isep.ipp.pt

† Faculdade de Ciências, Universidade do Porto
Rua do Campo Alegre, 4169-007 Porto, Portugal
jpp@dcc.fc.up.pt

ABSTRACT

This paper presents a new iterative algorithm for optimising thermal unit commitment in power generation planning. The approach, based on a mixed-integer formulation of the problem, considers a piecewise linear approximation of the fuel cost function that is dynamically updated to better reflect problem requirements, converging to the optimal solution.
After thorough computational tests in a broad set of instances, it showed to be flexible, capable of easily incorporating different problem constraints, and to be able to solve large size problems.

Keywords: Unit Commitment, Approximation Algorithms, Scheduling

1. INTRODUCTION

The Unit Commitment problem (UCP) is the problem of deciding which power generator units must be committed/decommitted over a planning horizon (lasting from 1 day to 2 weeks, and generally split in periods of 1 hour), and the production levels at which they must be operating (Pre-Dispatch), so that a given objective is optimised. The committed units must generally satisfy the forecasted system load and reserve requirements, subject to a large set of other system, technological and environmental constraints.

This is a topic of major practical relevance because the effectiveness of the schedules obtained has a strong economical impact in any power generation company. Due to that and to its complexity, it has received considerable research attention and, after several decades of intensive study, is still a rich and challenging topic of research.

Proposed optimisation techniques for Unit Commitment encompass very different paradigms, ranging from exact approaches and Lagrangian Relaxation to some rule of thumb or very elaborate heuristics and metaheuristics. The combinatorial nature of the problem and its multi-period characteristics prevented exact approaches from being successfully used in practice: they resulted in very inefficient algorithms that were only capable of solving small size instances of no practical interest. Heuristic techniques, as those based in Priority Lists, were also not very successful as they tended to lead to low quality solutions. Concerning metaheuristics, they had a very promising behaviour when they first started being explored. The quality of the results was better than the ones achieved by well established techniques, and good solutions were obtained very quickly.

Some drawbacks can however be pointed out when metaheuristics go into play. One major drawback, if one considers that the ultimate goal is to design techniques that can be accepted and used by a company, is the dependence of these techniques on parameter tuning. Tuning the parameters is a time consuming and somehow complex procedure that requires deep knowledge on the algorithm implemented. Furthermore, it is vital for good algorithm performance. A second drawback has to do with the lack of information this techniques provide in terms of solution quality (i.e. how far it is from the optimal solution). Some proposals have been made to soften the referred drawbacks; but this is still an open line of research.

Currently, the dramatic increase in efficiency of mixed-integer programming (MIP) solvers requests for a thorough exploitation of their capabilities. Some research has been directed towards the definition of alternative, more efficient, mixed-integer linear programming (MILP) formulations of the problem e.g. [1, 2]. Extensive surveys on different optimisation techniques and modelling issues are provided by e.g. [3, 4].

This paper presents a new MILP approach to the UCP that further explores this line of research. Instead of considering a quadratic representation of the fuel cost, we consider a piecewise linear approximation of that function and, in an iterative process update, it by including additional pieces. The function update is based in the solutions obtained in the previous iterations.

The approach was tested in a well known set of instances from the literature and showed to be flexible, capable of easily incorporating different problem constraints, and of solving large size problems.

2. PROBLEM DESCRIPTION

Different modelling alternatives, reflecting different problem issues have been published: they consider fuel, multiarea and emission constraints (e.g. [5, 6, 7]) and, more recently, security constraints [8] and market related aspects [9].

The decentralised management of production brought up new issues to the area [10], in some markets the problem being now reduced to single-unit optimisation. However, for several decentralised markets the traditional problem is still very similar to that of centralised markets [1, 2]. The main difference is the objective function that, rather than minimising production costs, aims at maximising total welfare. Therefore, the techniques that apply for a centralised management of the production, will also be effective

at solving many decentralised market production problems.

In this paper we will consider the centralised UC model. The objective is to minimise total production costs over a given planning horizon. They are expressed as the sum of fuel costs (quadratic functions that depend on the production level of each unit) and start-up costs. Start-up costs are represented by constants that depend on the last period the unit was operating; two constants are defined: one for hot start-up costs, that is considered when the unit has been off for a number of periods smaller or equal to a given value; and another for cold start-up costs, considered otherwise. The following constraints will be included in the formulation: system power balance demand, system reserve requirements, unit initial conditions, unit minimum up and down times, generation limits and ramp constraints. For a mathematical formulation the reader is addressed to [11].

3. MIP APPROACH AND COMPUTATIONAL RESULTS

The approach considers a piecewise linear approximation of the quadratic fuel cost function (see Equation (1)). P_{it} are decision variables that represent the production level of unit i in period t; a_i, b_i and c_i are fuel cost parameters for unit i (measured in \$/h, \$/MWh and \$/MW^{2}h, respectively). There are binary variables y_{it} that indicate the state of unit i in period t (0 if unit is off, 1 otherwise).

$$F(P_{it}) = \begin{cases} c_i P_{it}^2 + b_i P_{it} + a_i & \text{if } y_{it} = 1 \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

The main contribution of this paper concerns a linearisation of this cost function. As it is convex, if we find a straight line tangent to it, and constrain the cost to be greater than the value of the straight line, we have a lower approximation of the cost. The process devised here is to dynamically find straight lines, at points whose cost is being underestimated, and add them to a set; we then impose that the cost of a any production level p must be greater than the maximum of those straight lines, evaluated at p.

For the sake of clarity, let us remove the indices i,t identifying the generator. For any generator and any period, we start by approximating its cost by means of two straight lines: one going through $(P_{min}, F(P_{min}))$, and another going through $(P_{max}, F(P_{max}))$, as can be seen in Figure 1.

After solving the problem with this approximation, we obtain a production level for this unit of, say, p. The operating cost at this point will be underestimated by the value of the highest of the straight lines at p; in Figure 1, the value F. In order to exclude this point from the feasible region, we add another straight line to our set; the line tangent to the quadratic function, evaluated at p, as represented in blue in Figure 2. As we add more and more straight lines, we are converging to an exact approximation of the true cost function, as can be seen in Figure 2 for another possible value p'.

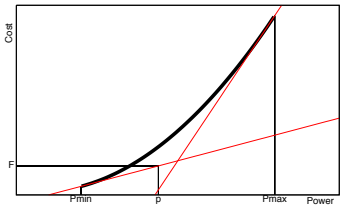


Figure 1: *Initial approximation of the cost function by two straight lines, going through the minimum and maximum operating power of the unit. If the current production level for this unit is p, its cost (in this iteration) will be approximated by F*

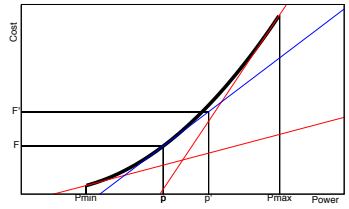


Figure 2: *Approximation of the cost function by the maximum of three straight lines, after obtaining production at level p on the previous iteration.*

3.1. Algorithm description

Initially, for each unit, the corresponding quadratic fuel cost function is approximated by two linear functions. Thereafter, more straight lines are iteratively added into a set, until having one iteration with all production levels being correctly evaluated, up to an acceptable error.

Let $\mathcal{N}$ be a set of integers identifying the power at which new tangents to the true cost are added; initially $\mathcal{P} = \{P_{min}, P_{max}\}$. At a given iteration, if the production level obtained in the MILP solution was p', we add this point to $\mathcal{P}$, except if there is a $p \in \mathcal{P} : |p' - p| < \varepsilon$.

In the MILP solved at each iteration, we add the constraints (making sure that they are only observed if the corresponding unit is switched on at the period considered)

$$F \geq \alpha_{in} + \beta_{in}(p - p_n) \quad \text{for } n = 1, \ldots, |\mathcal{P}|,$$

where p and F are instantiated to the actual producing levels P_{it} and costs F_{it} of a given unit, at a given period. For a given unit, the constants of the straight lines are obtained by:

$$\alpha_{in} = c_i p_n^2 + b_i p_n + a_i$$
$$\beta_{in} = 2 c_i p_n + b_i$$

In our implementation, we have set $\varepsilon = 1$; this allows an excellent approximation of the quadratic function in all the instances used (actually, we could observe no difference at all).

3.2. Computational results

The algorithm was tested in two sets of problems: one without ramp constraints but that has for long been a reference when comparing UC algorithms [12]; another where ramp constraints are included. CPU times were obtained with CPLEX 12.1, on a computer with a Quad-Core Intel Xeon processor at 2.66 GHz, running Mac OS X 10.6.6; only one core was assigned to this experiment.

Tables 1 and 2 present the results obtained with the algorithm proposed in this paper for different sets of instances. Problems P1 to P6, in Table 1, do not include ramp constraints. Those constraints are considered in problems R1 to R6 (Table 2). Problems R1 to R6, resulting from problems P1 to P6, set ramp up and down maximum values to the minimum production level of each unit. All problems consider a 24h planning horizon and the number of units ranges from 10 to 100.

Table 3 presents results reported in the literature for instances P1 to P6. Although the objective function value reported in this paper (565 828) for the 10 unit problem using the approximation algorithm is different from the one reported in other papers (565 825), the actual solution is the same. Small differences in values are justified by possible rounding of values by other authors.

In Tables 1 and 2, column *Quadr* provides the optimal result for the base problem and column *Lin* the result obtained by the approximation. Columns CPU_L and CPU_Q refer to the time spent (in seconds) to solve the quadratic problem and to reach convergence for the linear problem, respectively.

Prob.	Size	Lin	CPU_L	Quad	CPU_Q
P1	10	565 828	0.33	565 828	1.95
P2	20	1 126 000	7.46	1 126 000	241.
P3	40	2 248 280	134.	2 248 280	22716.
P4	60	3 368 950	2639.		
P5	80	4 492 170	192966.		
P6	100	5 612 690	157742.		

Table 1: Results for problems P1 to P6. Attempts to solve the problem with the quadratic formulation were not successful for instances with more than 50 units.

As far as the authors know, no optimal results had ever been established for problems P1 to P6, even for the smallest ones. We now show that for problems up to 40 units optimal results can be obtained by highly efficient MIP solvers. Furthermore, the effectiveness and efficiency of the approach proposed in this paper are reflected in the values of columns *Lin* and CPU_L, respectively. For problems up to 40 units the iterative approach is able to reach the optimal solution with dramatical cuts in CPU times, when compared to direct solution with the quadratic solver of CPLEX. For problems of bigger size, good lower bounds on the optimal result are also reachable as can be concluded by comparing those values with the best published values for the quadratic problem (see Table 3).

Similar conclusions may be taken for the ramp problem. The quadratic solver of CPLEX was capable of reaching optimal solutions for instances of up to 20 units. Optimal values for the same set of problems were also reached by the approximation algorithm, that was capable of solving instances of up to 80 units.

Prob.	Size	Lin	CPU_L	Quad	CPU_Q
R1	10	573 570	0.94	573 570	2.00
R2	20	1 144 450	258.	1 144 450	147.17
R3	40	2 284 670	12084.		
R4	60	3 424 310	1830.		
R5	80	4 565 420	41907.		
R6	100				

Table 2: Results for problems R1 to R6. Attempts to solve the problem with the quadratic formulation were not successful for instances with more than 20 units. With the linearisation algorithm, limiting CPU to 200000 seconds, allowed solution of instances with up to 80 units.

4. CONCLUSIONS AND FURTHER DEVELOPMENTS

The main contribution of this paper is a method for approximating the quadratic cost of electricity generating units, with an iterative method that converges to the exact solution.

Computational analysis shows that for problems without ramps the method is capable of reaching the quadratic optimal result whenever it is known, within much less computational time. For larger instances, where the quadratic problem optimal is not known, the method also provides high quality lower bounds for the results.

The paper also establishes optimal results for small size instances showing that currently, state-of-the-art MIP solvers can solve to optimality problems that were not solvable before.

Prob.	Size	LR [12]	GA [12]	LR–MA [13]
P1	10	565 825	565 825	565 827
P2	20	1 130 660	1 126 243	1 127 254
P3	40	2 258 503	2 251 911	2 249 589
P4	60	3 394 066	3 376 625	3 370 595
P5	80	4 526 022	4 504 933	4 494 214
P6	100	5 657 277	5 627 437	5 616 314
		ICGA [14]	GRASP [11]	CON [15]
P1	10	566 404	565 825	565 825
P2	20	1 127 244	1 126 805	1 126 070
P3	40	2 254 123	2 255 416	2 248 490
P4	60	3 378 108	3 383 184	3 370 530
P5	80	4 498 943	4 524 207	4 494 140
P6	100	5 630 838	5 668 870	5 615 410

Table 3: Previous results for problems P1 to P6.

Similar conclusions can be taken when ramp constraints are modelled. The method is also capable of reaching quadratic optimal results (now with extra computational time). Furthermore, for problems with more than 20 units where quadratic optimal solutions were not obtained, the approximate method was still effective.

As future work the authors plan to include additional features in the algorithm to make it more efficient for very large size problems.

5. ACKNOWLEDGEMENTS

Financial support for this work was provided by the Portuguese Foundation for Science and Technology (under Project PTDC/EGE-GES/099120/2008) through the "Programa Operacional Temático Factores de Competitividade (COMPETE)" of the "Quadro Comunitário de Apoio III", partially funded by FEDER.

6. REFERENCES

[1] M. Carrio and J. Arroyo, "A computationally efficient mixed-integer linear formulation for the thermal unit commitment problem," *IEEE Transactions in Power Systems*, vol. 21, no. 3, pp. 1371–1378, 2006.

[2] A. Frangioni, C. Gentile, and F. Lacalandra, "Tighter approximated milp formulations for unit commitment problems," *Power Systems, IEEE Transactions on*, vol. 24, no. 1, pp. 105 –113, Feb. 2009.

[3] N. Padhy, "Unit commitment – a bibliographical survey," *IEEE Transactions in Power Systems*, vol. 19, no. 2, pp. 1196–1205, 2004.

[4] H. Yamin, "Review on methods of generation scheduling in electric power systems," *Electric Power Systems Research*, vol. 69, pp. 227–248, 2004.

[5] F. Lee, "A fuel constrained unit commitment method," *IEEE Transactions on Power Systems*, vol. 4, pp. 1208–1218, 1989.

[6] Z. Ouyang and S. Shahidehpour, "Heuristic multi-area unit commitment with economic dispatch," *IEE Proceedings – C*, vol. 138, pp. 242–252, 1991.

[7] D. Srinivasan and A. Tettamanzi, "An evolutionary algorithm for evaluation of emission compliance options in view of the clean air act amendments," *IEEE Transactions on Power Systems*, vol. 12, no. 1, pp. 336–341, 1997.

[8] Y. Fu and M. Shahidehpour, "Fast scuc for large-scale power systems," *Power Systems, IEEE Transactions on*, vol. 22, no. 4, pp. 2144 –2151, Nov. 2007.

[9] J. Xu and R. Christie, "Decentralised unit commitment in competitive energy markets," in *The Next Generation of Electric Power Unit Commitment Models*, B. Hobbs, M. Rothkopf, R. O'Neill, and H. Chao, Eds. Kluwer Academic Publishers, 2001, pp. 293–315.

[10] B. Hobbs, M. Rothkopf, R. O'Neill, and H. Chao, Eds., *The Next Generation of Electric Power Unit Commitment Models*. Kluwer Academic Publishers, 2001.

[11] A. Viana, J. Sousa, and M. Matos, "Using GRASP to solve the unit commitment problem," *Annals of Operations Research*, vol. 120, no. 1, pp. 117–132, 2003.

[12] S. Kazarlis, A. Bakirtzis, and V. Petridis, "A Genetic Algorithm solution to the unit commitment problem," *IEEE Transactions on Power Systems*, vol. 11, pp. 83–92, 1996.

[13] J. Valenzuela and A. Smith, "A seeded memetic algorithm for large unit commitment problems," *Journal of Heuristics*, vol. 8, no. 2, pp. 173–195, 2002.

[14] I. G. Damousis, A. Bakirtzis, and P. Dokopoulos, "A solution to the unit commitment problem using integer-coded genetic algorithm," *IEEE Transactions on Power Systems*, vol. 19, pp. 1165–1172, 2004.

[15] A. Viana, J. Sousa, and M. Matos, "Fast solutions for UC problems by a new metaheuristic approach," *Electric Power Systems Research*, vol. 78, pp. 1385–1395, 20087.

Dispatch Hydroelectric Power Plant using Genetic Algorithm

Jessica Pillon Torralba Fernandes * Paulo de Barros Correia *

* Department of Energy, Faculty of Mechanical Engineering, University of Campinas - UNICAMP
Campinas, Brazil
{pillon, pcorreia}@fem.unicamp.br

ABSTRACT

This paper presents an optimization model for daily operation of Middle Sao Francisco River hydroelectric system in Brazil. The study considers eight hydroelectric power plants – Sobradinho, Luiz Gonzaga, Apolonio Sales, Paulo Afonso I, II, III, IV e Xingo – witch belongs to the Sao Francisco Hydroelectric Company. Its objective is to maximize the hydroelectric power plant efficiency and, simultaneously, to minimize the number of startups and shutdowns of generating units. The technique of resolution is made in two steps: Step 1 determines the load allocated in each hydroelectric power plant at each per hour and Step 2 defines the number of generating units in operation and the load of particular power plant. The mathematical formulation is non-linear mixed integer programs and solved with a Genetic Algorithm (GA) approach, and Linear Programming . This model was implemented with two computation programs, one a commercial optimization solver, and a in house GA solver coded with a programming language of four generation. One of programs was used as interface, while the fourth generation, the optimization model was implemented.

Keywords: Linear and non-linear optimization, Multiobjective optimization, Hydroeletric system, Generating units, Genetic algorithm

1. INTRODUCTION

Several objectives are adopted for the dispatch models of generating units in hydroelectric power plants. Generally, the problem of maximizing the efficiency of the Brazilian hydroelectric plants has as the main objective a model for the Optimal Load Dispatch (DOC). The DOC resolves the load allocation problem of the hydroelectric plants and it can be implemented as an Evolutionary Computation problem, specifically with Genetic Algorithm. It also allows calculating the global efficiency of the plants when the operating conditions, the hills curves and operatives restrictions are known.

According to [1], the efficiency of generating units is the main factor influencing the performance of generation of electricity in a hydroelectric power plant . The operation planning of generation systems covers the long, medium and short term. This article focuses on the short-term operation. The short-term programming requires a more detailed mathematical representation of the operatives restrictions and it is determined the curve of a generation plant, and then, the units are chosen to be dispatched. Thus, this paper proposes an optimization model of the Sao Francisco's hydroelectric plants daily operation. Its objective is to maximize the plant's efficiency and minimize the number of startups and shutdowns of the generating units simultaneously. The literature presents a significant number of works that relate the problem of dispatch with different approaches that vary according to the applicability of the same.[2] proposed a model of multiobjective optimal dispatch for the operation of a hydroelectric power plant. The model consists of two algorithms based on GA. The first algorithm is used to allocate the generating units and aims to maximize the efficiency of power plant at each time interval. The second step aims to maximize efficiency and minimize the number of startups and shutdowns of generating units.

The dispatch model proposed by [3], and [4], was divided into two subproblems called Dispatch of Units (DU) and Dispatch Generation (DG). DG was solved via Lagrangean Relaxation and DU was used with Genetic Algorithms. This methodology was applied to actual case study of the hydroelectric power plants system of Paranapanema in Brazil.

2. PHYSICAL ASPECTS

It is important that the physical aspects of generating units must be more detailed in the dispatch, such as operational restriction and the operating characteristics (for example their efficiencies), where costs and goals are more important.

- *Unit efficiencies*
  Generation unit efficiency depends on three variables: water head of the plant, water discharge and eletric power of the unit. The hill is a three-dimensional curve that plots efficiency as a fuction of the water head of the plant and the eletric power of unit, as shown in Figure 1.

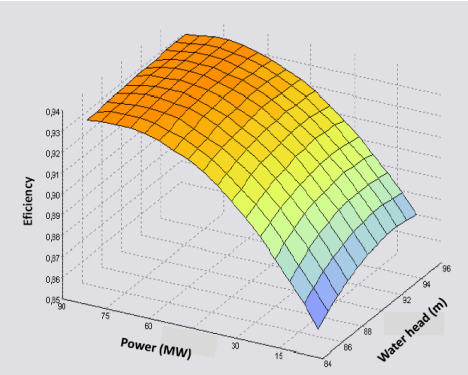


Figure 1: *Hill curve of a real hydroelectric power plant.*

- *Demand*
  The load of the plant is determined by long- and mid-term planning. A short-term scheduling model estimates the plant's daily load curve. The Figure 2 shows a typical load curve of one day. In this case, the demand to be met by power plants of Middle Sao Francisco river.
- *Startups and Shutdowns of generating units*
  In some studies the costs of startups and shutdowns of the generating units have a great importance, since it decreases

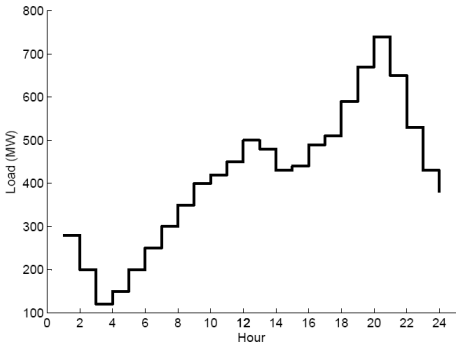


Figure 2: *Typical daily load curve.*

the life of units and increases the maintenance of windings and mechanical equipament them.

A study presented by [5] showed how startups affect the cost of short term hydro operation and how these costs affect short term scheduling strategies of power producing companies in Sweden. Overall, the study points to an approximate value of 3US$/*MW*.

- *Plant Production Factor*

  Power output in a hydroelectric plant per unit turbine flow. It varies according to plant gross head, and is expressed in *MW/m3/s*. For purposes of illustration, the Figure 3 shows the productivity of a specific plant from Brazil.

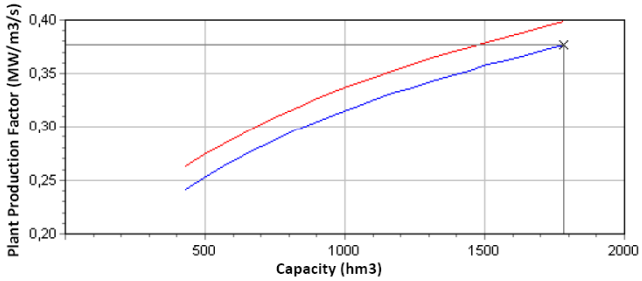


Figure 3: *Plant Production Factor.*

3. GENETIC ALGORITHM

Math and computational techniques have been developed for decades with the principles of Darwin's evolution theory, defining what is known as Evolutionary Computation. Inside its branches, Genetic Algorithms (GA) are the most used [6]. GA were developed by Holland [7], who analyzed the phenomena of the process of natural selection of species and the genetic selection of races. Each individual in the GA is a coding of a possible solution of a problem. This encoding can be binary or real.

The first step towards its implementation is the generation of an initial population, that for most problems is randomly generated. However, depending on the application forms, the individuals can be selected heuristically to compose a more favorable population [8]. GA use some genetic operators like crossover and mutation, and these operators are applied to generate new solutions inside a feasible set of solutions.

Also, the operators are randomized to provide diversities in the overall population seeking global optimal solutions. The advantage of GA is that its use does need differentiable functions, so they can be applied to problems with discontinuities, which are very common in dispatch problems.

4. PROBLEM DESCRIPTION

4.1. The Sao Francisco river

The Sao Francisco is a river in Brazil. With a length of 3200 kilometres, the Sao Francisco originates in the Canastra mountain range in the central-western part of the state of Minas Gerais and traverses the states of Minas Gerais (MG), Bahia (BA), Pernambuco (PE), Sergipe (SE) and Alagoas (AL).

Cascade Middle Sao Francisco River is formed by uses of the HPPs Sobradinho, Luiz Gonzaga, Apolônio Sales (Moxotó), Paulo Afonso I, II, III, IV and Xingó. These HPPs are the core of the system producing electric power from the Northeast, *Companhia Hidro Eletrica do Sao Francisco* (CHESF). The Figure 4 shows the location of the Middle Sao Francisco in Brazil, along with the HPPs.

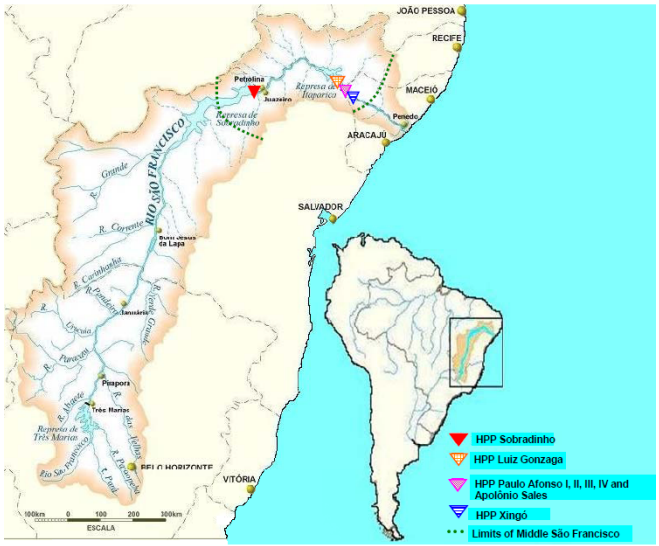


Figure 4: *System of the Middle Sao Francisco with him HPPs located in Brazil.*

The Figure 5 illustrates the HPPs Cascade Middle Sao Francisco.

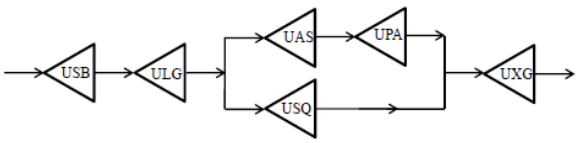


Figure 5: *Cascade Middle Sao Francisco river in Brazil.*

4.2. Mathematical Formulation

The problem presented is solved in two steps, as follows Diagram 6.

The dispatch is described by Equations 1 to 9

Figure 6: *Diagram of the proposed problem.*

$$\text{Max} \sum_{j=1}^{24} \sum_{i \in UHE} \sum_{k \in M_i} \eta_i(p_{i,k,j}) y_{i,k,j} \quad (1)$$

$$\text{Min} \sum_{j=2}^{24} \sum_{i \in UHE} \sum_{k \in M_i} \left| y_{i,k,j} - y_{i,k,j-1} \right| \quad (2)$$

s.a.

$$\sum_{i \in UHE} \sum_{k \in M_i} p_{i,k,j} = d_j - \overline{G} \quad (3)$$

$$\sum_{j=1}^{24} \sum_{k \in M_i} \frac{p_{USB,j}}{\rho_{USB}(x^0_{USB}, p_{USB,k,j})} = 24 Q_{USB} \quad (4)$$

$$\sum_{j=1}^{24} \sum_{k \in M_i} \frac{p_{ULG,j}}{\rho_{ULG}(x^0_{ULG}, p_{ULG,k,j})} = 24 Q_{ULG} \quad (5)$$

$$\sum_{j=1}^{24} \sum_{k \in M_i} \frac{p_{USQ,j}}{\rho_{USQ}(x^0_{USQ}, p_{USQ,k,j})} = 24 Q_{USQ} \quad (6)$$

$$\sum_{j=1}^{24} \sum_{k \in M_i} \frac{p_{UXG,j}}{\rho_{UXG}(x^0_{UXG}, p_{UXG,k,j})} = 24 Q_{UXG} \quad (7)$$

$$p^{min}_{i,k,j} y_{i,k,j} \leq p_{i,k,j} \leq p^{max}_{i,k,j} y_{i,k,j} \quad (8)$$

$$y_{i,k,j} \in \{0,1\} \quad (9)$$

for $i \in UHE = \{USB, ULG, USQ, UXG\}$, $k = \{1, ..., n\}$ and $j = 1, ..., 24$, where

This problem has a multiobjective character because its objective functions 1 and 2 seek to maximize productivity and minimize the number of startups and shutdowns, respectively.

Equations 4 to 7 represent the daily average for each mill. The variable $k_{i,j}$ indicates whether unit i is dispatched ($k_{i,j} = 1$) or not dispatched ($k_{i,j} = 0$).

USB	HPP Sobradinho
ULG	HPP Luiz Gonzaga (Itaparica)
USQ	HPP Paulo Afonso IV
UXG	HPP Xingó
i	Power plant index
k	Generating unit index
j	Time period index
Q_i	Avarage flow of that the HPP i must keep during the day
$p_{i,j}$	Power generated by the HPP i in period j
x^0_i	Reservoir level of the HPP i in the last period of the previous day
UHE	Set of power plants $UHE = \{USB, ULG, USQ, UAS\}$
ρ_i	Plant Production Factor function of the HPP i
η_i	Efficiency function of the power plant i
$\overline{G}$	Generation of HPP UPA e UAS
M_i	Set of UG of the power plant i
$y_{i,k,j}$	Indicates if the UG k of the power plant i in period j is dispatched
d_j	Demand of the four power plants UHE in period j
M_i	Set of UG of the power plant i
$k_{i,j}$	Indicates if the UG k of the power plant i in period j is dispatched
$p^{min}_{i,j}(k_{i,j})$	Minimum power to $k_{i,j}$ UG
$p^{max}_{i,j}(k_{i,j})$	Maximum power to $k_{i,j}$
$\overline{G}$	Generation of HPP UPA and UAS

Table 1: Variables used in the mathematical formulation.

5. METHODOLGY

The problem above is solved in two steps, as Figure 7. The Step 1

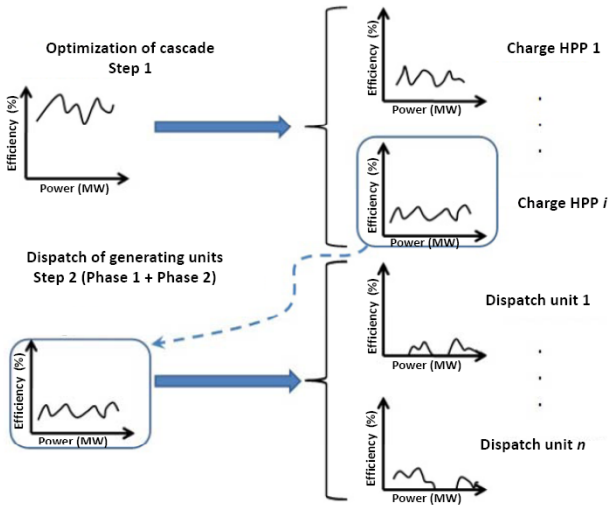


Figure 7: *Illustration of the problem.*

determines how much each power plant must generate at each time interval. It provides an initial solution which takes into account the service and video-streaming market averages per hydroelectric power plant.

From this initial solution, the Step 2 determines the number of units in operation and the load of a particular plant. This last step is divided into two phases which are solved iteratively until convergence.

5.1. Step 1

This Step 1 solves a simplified problem given below, which does not decide on the number of machines in operation.

$$\text{Min} \sum_{j=1}^{24} \sum_{i \in UHE} \frac{p_{i,j}}{\overline{\rho}_i} \tag{10}$$

s.a.

$$\sum_{i \in UHE} p_{i,j} = d_j - \overline{G} \tag{11}$$

$$\sum_{j=1}^{24} \frac{p_{i,j}}{\overline{\rho}_{USB}} = 24 Q_{USB} \tag{12}$$

$$\sum_{j=1}^{24} \frac{p_{ULG,j}}{\overline{\rho}_{ULG}} = 24 Q_{ULG} \tag{13}$$

$$\sum_{j=1}^{24} \frac{p_{USQ,j}}{\overline{\rho}_{USQ}} = 24 Q_{USQ} \tag{14}$$

$$\sum_{j=1}^{24} \frac{p_{UXG,j}}{\overline{\rho}_{UXG}} = 24 Q_{UXG} \tag{15}$$

$$p_{i,j}^{min}(1) \leq p_{i,j} \leq p_{i,j}^{max}(n_{i,j}) \tag{16}$$

for $i \in UHE = \{USB, ULG, USQ, UXG\}$ and $j = 1, ..., 24$.

5.2. Step 2

Due to its mixed character, the problem in this step is decomposed into two phases, iteratively solved until convergence. Both phases are resolved by the GA techniques.

5.2.1. *Phase 1*

The dispatch problem formulation in this phase is described by the following objective function and constraints, with time j and HPP i are fixed.

$$\text{Max} \sum_{i \in UHE} \sum_{k \in M_i} \eta_i(p_{i,k,j}) y_{i,k,j} \tag{17}$$

s.a.

$$\sum_{i \in UHE} \sum_{k \in M_i} p_{i,k,j} = d_j - \overline{G} \tag{18}$$

$$\sum_{k \in M_i} \frac{p_{i,j}}{\rho_i(x_i^0, p_{i,k,j})} = 24 Q_i \tag{19}$$

$$p_{i,k,j}^{min} y_{i,k,j} \leq p_{i,k,j} \leq p_{i,k,j}^{max} y_{i,k,j} \tag{20}$$

$$y_{i,k,j} \in \{0,1\} \tag{21}$$

for $i \in UHE = \{USB, ULG, USQ, UXG\}$, $k \in M_i$ e $t = 1, ..., 24$.

5.2.2. *Phase 2*

The dispatch problem formulation in the second Phase is described by the following objective function and constraints, with HPP i fixed.

$$\text{Max} \sum_{j=1}^{24} \sum_{k=1}^{n} \eta_i(p_{k,j}) y_{k,j} \tag{22}$$

$$\text{Min} \sum_{j=2}^{24} \sum_{k=1}^{n} \left| y_{k,j} - y_{k,j-1} \right| \tag{23}$$

s.a.

$$\sum_{k \in M_i} p_{k,j} = d_j - \overline{G} \tag{24}$$

$$\sum_{j=1}^{24} \sum_{k=1}^{n} \frac{p_j}{\rho(x^0, p_{k,j})} = 24 Q \tag{25}$$

$$p_{k,j}^{min} y_{k,j} \leq p_{k,j} \leq p_{k,j}^{max} y_{k,j} \tag{26}$$

$$y_{k,j} \in \{0,1\} \tag{27}$$

for $i \in UHE = \{USB, ULG, USQ, UXG\}$, $k \in M_i$ and $j = 1, ..., 24$.

To the Step 2, was one chosen HPPs Sobradinho and Paulo Afonso IV to be the study of case.

6. RESULTS

It was considered a daily horizon with a half-hour discretization containing all the HPPs in cascade, according to the schedule data held on September 10, 2007. The daily load curve to be attended by the cascade, the initial state of the reservoirs and expected inflows for each day, were the available data provided by CHESF.

The Step 1 produced a graph that shows the result in terms of generating for each HPPs of cascade, shown in Figure 8. Basically, all the HPPs followed the curve of charge and its ranged according to her keeping the levels of its reservoirs within the allowed limit.

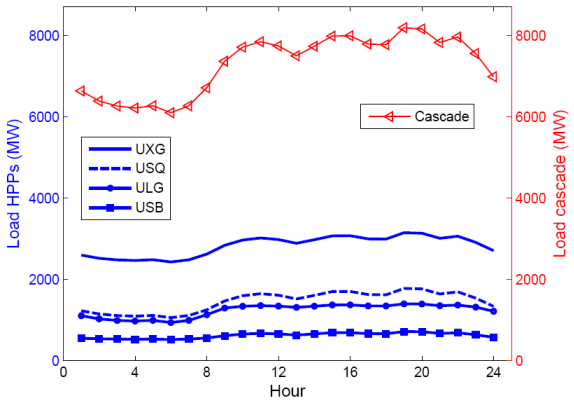


Figure 8: *Generation of cascade and HPPs.*

In Step 2 obtained the graphics of generation and centrifugation for HPPs Sobradinho and Paulo Afonso IV, also indicating the limits of maximum and minimum generation, shown in Figures 9 and 10.

7. CONCLUSIONS

This paper approached the dispatch problem by a mathematical model that maximizes the energy efficiency of power plant taking into account the operational restrictions translated in terms of reservoir levels, the swallowing of the turbines, the goal of generation and video-streaming of the HPP.

The genetic algorithm is a powerful optimization tool that has been used very often in solving similar problems in the proposed work. The efficiency of its use in simulation of this work showed an appropriate discovery of dispatch. The result achieved with its use was a great diversity of solutions with startups and shutdowns different that the best solution will be found depending on the priority of the problem.

The applicability of this model can be used for optimization of other HPPs in cascade.

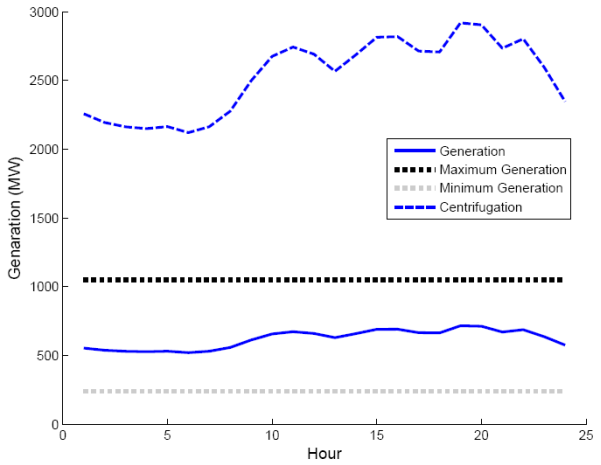


Figure 9: *Generation and Centrifugation for Sobradinho.*

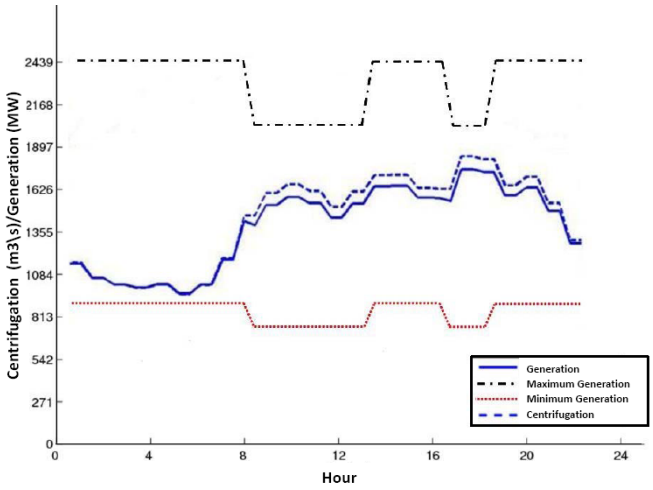


Figure 10: *Generation and Centrifugation for Paulo Afonso IV.*

8. REFERENCES

[1] C. T. Salmazo, "Modelo de otimizacao eletro-energergico de curto prazo (pre-despacho) aplicado ao sistema copel," Master's thesis, Faculdade de Engenharia Eletrica e de Computacao, Universidade Estadual de Campinas, 1997.

[2] G. Conalgo and P. Barros, "Multiobjective dispatch of hydrogenerating units using a two-step genetic algorithm method," *IEEE Congress on Evolutionary Computation*, pp. 2554 – 2560, 2009.

[3] E. F. D. Santos, "Um modelo de pre-despacho em usinas hidreletricas usando algoritmos geneticos," Master's thesis, Faculdade de Engenharia Eletrica e Computacao, Universidade Estadual de Campinas, 2001.

[4] A. S. A. Encina, "Despacho otimo de unidades geradoras em sistemas hidreletricos via heuristica baseada em relaxacao lagrangeana e programacao dinamica," Ph.D. dissertation, Faculdade de Engenharia Eletrica e de Computacao, Universidade Estadual de Campinas, 2006.

[5] O. Nilsson and D. Sjelvgren, "Hydro unit start-up costs and their impact on the short term scheduling strategies of swedish power producers," *IEEE Transactions on Power Systems*, vol. 12, pp. 38 – 43, 1997.

[6] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, 3, Ed. Sringer, 1996.

[7] H. J. Holland, *Adaptation in Natural and Artificial Systems*. The University of Michigan Press, 1975.

[8] E. G. M. Lacerda and A. C. P. L. Carvalho, *Introducao aos Algoritmos Geneticos*. Universidade Federal do Rio Grande do Sul, 1999.

Algebraic Group Theory driven Divide and Evolve of multi-objective Problems

Nail El-Sourani * Markus Borschbach *

* Chair of Optimized Systems, University of Applied Sciences
FHDW, Haupstrasse 2, D-51465 Bergisch Gladbach
{nail.el-sourani, markus.borschbach}@fhdw.de

ABSTRACT

Most real world problems remain as a multi-objective solution space. To overcome the well known computational complexity of such problems, the divide and evolve is a feasible solution, if the sub-problems remain solvable. This paper envisions a road-map, when and how to apply algebraic group theory structures into a multi stage evolutionary approach. It solves certain combinations of objectives from group stage to group stage in a nested group structure, until the reference problem at hand even reaches the distinct solution of the problem. Further, the quality of the solution, i.e. the overall number of steps to reach the solution results in a low number of steps (albeit not the lowest possible). Performance and integrity of this approach are consequently verified.

Keywords: Group theory, Divide and evolve, Evolution strategy, Discrete optimization

1. INTRODUCTION

The universe of combinatorial optimization problems is a quite diverse space of problems. Evolutionary solutions for so far infeasible complexity spaces provide an opportunity if an algebraic group theory based structure can be identified. The Rubik's Cube is introduced as a reference and benchmark problem to fulfill an integrity and performance profile of a consequently applied algebraic group theory driven divide and evolve approach. The main task is to find a structure of subgroups which, when transformed for application as fitness function(s) in an evolutionary approach, enable an overall multi-objective optimization problem - previously non-solvable or only with high computational cost - to be solved in reasonable time. The problem at hand, introduced and formalized in this paper, is multi-objective in the sense that a scrambled Cube has to be solved (first objective) using a preferably small number of moves (second objective).

On a general level, a group-theoretic structure has to be found, which divides the infeasible problem domain into solvable tasks, represented by algebraic groups. The phase-transition of solutions from one group to the following one is realized by specific fitness functions for each group-transition. Each transition itself solves a partly multi-objective subproblem with varying, subgroup-induced prime objectives. Making use of the nested group structure guarantees a steady improvement of individuals and promotes a stable population towards the end of each evolution phase. Each group induces a combination of constraints which remain fulfilled and subsequently add up until the final group-transition.

Large population sizes and the presented evolutionary phase-transition mechanic increase individual diversity to ensure efficient transitions from group to group and finally the overall unique solution. This remains different from the general combinatorial optimization task which, in general, defines an equal number of solutions. In the reference problem however, the sequences of moves found for group-transitions remain non-deterministic and therefore different. The overall solution is a single unique point in the search space. By deriving a statistical analysis of the search space, a simulation onset based on an integrity verification is provided. Accordingly, all computationally feasible states up to a certain complexity have been generated. The presented approach has been approved upon this onset and further a random selection of more complex points of the search space to ensure a solution from every point of the search space (including the known most complex). In the case of this reference problem, each solution in the search space is evaluated by the exact and shortest solution known so far.

2. DIVIDE AND CONQUER THE RUBIK'S CUBE

2.1. Structure and Notation

The classic 3^3 Rubik's Cube is widely known and the one subject to this paper. It consists of 26 pieces: 8 corner pieces, 12 edge pieces and 6 center pieces, distributed equally on the six sides of the Cube. Each side of the Cube will be called *face*, each 2-dimensional square on a face will be referred to as *facelet*.

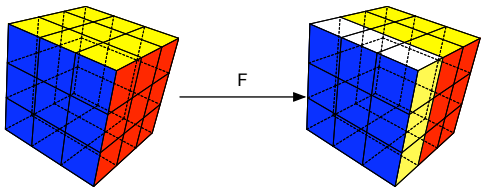


Figure 1: *Classic* 3^3 *Rubik's Cube, effect of CW turn of front face.*

Corners, edges and centers are all *cubies* - representing the physical object. A corner shows 3 facelets, an edge 2 and a center 1. Each side of the Rubik's Cube can be rotated clockwise (CW) and counterclockwise (CCW). Every such single move changes the position of 4 edges and 4 corners - note that the center facelet on every of the Cube's faces always stays in the same position (see Figure 1). Thus, the color of a solved face is always determined by its center color. For each edge and corner it is of great importance to distinguish between *position* and *orientation*: i.e. an edge can be in its right position (defined by the two adjacent center colors) but in the wrong orientation (flipped).

There are several known notations [11] for applying single moves on the Rubik's Cube. We will use F, R, U, B, L, D to denote a clockwise quarter-turn of the front, right, up, back, left, down face and Fi, Ri, Ui, Bi, Li, Di for a counterclockwise quarter-turn. Every such turn is a *single move*. In Cube related research half-turns $(F2, R2, U2, B2, L2, D2)$ are also counted as single move, we will do so as well. This notation is dependent on the users viewpoint to the cube rather than the center facelets' colors.

2.2. Algebraic Characteristics

A *group* G is a set together with multiplication and identity e ($eg = g$), inverse ($gg^{-1} = g^{-1}g = e$) and an associative law. A *subgroup* $H < G$ is a subset H that is closed under group operations. $S \subseteq G$, written $G =< S >$ is a *generator* of G if any element of G can be written as a product of elements of S and their inverses. The *order* of the group is the number of elements in it, $|G|$.

All possible states of a Rubik's Cube are described by the group generated by its applicable moves $G_C =< F,R,U,B,L,D >$, also called the *Cube Group* ($|G_C| = 4.3 \cdot 10^{19}$). All configurations of the Rubik's Cube can be reached by using combinations of single moves in this group, thus the single moves *generate* G_C. Further, there is always a neutral element, i.e. $F \cdot FFFF = FFFFF = F$ and $F^4 = 1$ (also showing the order of each generator in G_C is 4) and an inverse: $Fi \cdot F = 1$ and $Fi = FFF$

Given a group G and a subgroup $H < G$, a *coset* of H is the set $Hg = hg : h \in H$; thus, $H < G$ partitions G into cosets. The set of all cosets is written $H \backslash G$.

Let $H =< L,R,F,B,U2,D2 >$ be a subgroup of G_C, representing a Cube where only the edge positions matter, as no edge orientations can be altered. Thus, $H \backslash G_C$ depicts the left coset space which contains all possibly attainable states when only flipping edge cubies (changing an edges orientation). For extended explanation refer to [6], [3].

2.3. Related Work

Solving the Rubik's Cube is a challenging task. Both the size of the solution space induced by the number of attainable states and multiple desirable side-objectives next to restoring the Cube (favorably in the smallest possible number of moves and lowest calculation complexity) make this an interesting optimization problem. Although invented in 1974, the number of moves required to solve any state of Rubik's Cube (the so-called *God's Number*) has just recently been found to be 20 [12].

Various algorithms were devised to decrease the upper bound. However, all those approaches are strictly exact methods and the most recent ones rely on terabytes of pre-calculated lookup-tables. This is reflected in the research road-map of lowest upper bounds by Rokicki [12] to finally prove it to be 20. This number was attained by applying the same method he had used earlier for pushing the upper bound to 26, 25 and then 23 moves - using the very same algorithm only on more powerful hardware and a longer calculation time [11], [12].

Evolutionary Algorithms have been successfully applied in a variety of fields, especially highly complex optimization problems [2], [9], [14]. Oftentimes, superior solutions - as compared to classical algorithms have been achieved - notably in multi-objective cases (for example multi-constraint knapsack problems [5]). This gives rise to the idea of applying Evolutionary Algorithms to the Rubik's Cube problem. All relevant approaches are based on dividing the solution space of the Rubik's Cube into mathematical groups, starting with Thistlethwaite using 4 [13], then Reid combining two of Thistlethwaite's groups resulting in total of 3 [10] and finally Kociemba's [8] and Rokicki's approach using 2 subgroups. This makes the group theoretic approach a reasonable starting point for designing Evolutionary Algorithms. It is of particular interest to us to determine how such an EA can solve the Cube without relying on extensive lookup-tables. Only a few evolutionary approaches dedicated to solve the Rubik's Cube exist. In 1994 Herdy devised a method which successfully solves the Cube [7] using pre-defined sequences as mutation operators that only alter few cubies, resulting in very long solutions. Another approach by Castella could not be verified due to a lack of documentation. Recently Borschbach and Grelle [1] devised a 3-stage Genetic Algorithm based on a common human "SpeedCubing" [11] method, first transforming the Cube into a 2x2x3 solved state, then into a subgroup where it can be completed using only two adjacent faces (two-generator group).

2.4. Rubik's Cube as an Individual

The Rubik's Cube is represented using 6 2D matrices containing values from 1 to 6, each representing one color. Every quarter- and half-turn can be applied to this representation, yielding a total of 18 different single moves while still leaving the Cube's integrity intact. Thus, mutation is easily realized by not modifying a single facelet's color but applying a sequence of moves to the Cube. This guarantees that the Cube's integrity stays intact at all times and makes a separate integrity test superfluous. Every individual remembers the mutations it has undergone, i.e. a list of moves that have been applied. To keep this list as small as possible, redundant moves are automatically removed. For example an individual that has been mutated with F and is then mutated with $FRRiB$ will only remember the optimized sequence $F \cdot FRRiB = F2B$, preventing redundancy. Essentially, this is realized via a while-loop, eliminating redundant moves in each pass until no further optimizations can be made: e.g. $F2BBiR2R2F$ is optimized to Fi by first removing BBi, then removing $R2R2$ and finally transforming $F2F$ into Fi.

3. FITNESS FUNCTION BASED ON ALGEBRAIC GROUPS

3.1. Divide and Conquer

Translating the classic Thistlethwaite Algorithm [13] into an appropriate Fitness Function for an Evolutionary Algorithm essentially forces the design of four distinct subfunctions. As each subgroup of G_0 has different constraints, custom methods to satisfy these constraints are proposed. The groups provided by Thistlethwate [13] are: $G_0 =< F,R,U,B,L,D >$, $G_1 =< F,U,B,D,R2, L2 >$, $G_2 =< U,D,R2,L2,F2,B2 >$, $G_3 =< F2,R2,U2,B2,L2, D2 >$, $G_4 = I$.

Obviously, $G_0 = G_C$. The functional principle of Thistlethwaite's Algorithm is to put the Cube into a state where it can be solved by only using moves from G_i which again has to be achieved by only using moves from G_{i-1} for $i = 1,\ldots 4$, thus named *nested groups*. This provides the basis of the presented divide and conquer ES-approach. As we use randomly generated mutation sequences (albeit dependent of the current fitness phase/group in the final version), first attempts while working in the whole of the group G_C would consistently fail to solve due to the very high order of $|G_C|$ - and thus the solution space.

The divide and conquer ES-approach however evolves a transition sequence for an individual in the current coset space $G_{i+1} \backslash G_i$ to the next one ($i = i+1$). These coset spaces, each describing a reduced form of the 3^3 Rubik's Cube puzzle, induce different kinds of constraints. This directly results in the total number of attainable states being reduced by using only moves from some subgroup G_{i+1}. The exact orders for each group are calculated exemplary for $G_1 \backslash G_0$ (complete calculations are found in [3], [4]):

The first coset space $G_1 \backslash G_0$ contains all Cube states, where the edge orientation does not matter. This is due to the impossibility of flipping edge cubies when only using moves from G_1. As there are 2^{11} possible edge orientations,

$$|G_1 \backslash G_0| = 2^{11} = 2048 \quad (1)$$

the order of $|G_1|$ is

$$|G_1| \equiv \frac{|G_0|}{|G_1 \diagdown G_0|} = 2.11 \cdot 10^{16} . \quad (2)$$

3.2. Algebraic group-based Fitness Calculation

$G_0 \rightarrow G_1$
To reach G_1 from any scrambled Cube, we have to orient all edge pieces right while ignoring their position. The fitness function for this phase simply increases the variable $phase_0$ by 2 for each wrong oriented edge. Furthermore, we add the number of moves that have already been applied to the particular individual in order to promote shorter solutions, yielding a multi-objective optimization problem. Finally, we adjust the weight between w (number of wrong *oriented* edges) and c (number of moves applied to current Cube individual). This will be done similarly in all subsequent phases.

$$phase_0 = 5 \cdot (2w) + c \quad (3)$$

With a total of 12 edges which can all have the wrong orientation this gives $max\{2w\} = 24$. The Cube has been successfully put into G_1 when $phase_0 = c$. Reaching G_1 is fairly easy to accomplish, thus making the weight-factor 5 a good choice.

$G_1 \rightarrow G_2$
In order to fulfill G_2 the 8 corners have to be oriented correctly. Edges that belong in the middle layer get transferred there. Tests with the Thistlethwaite ES showed it somewhat problematic to do this in one step. Oftentimes, the algorithm would get stuck in local optima. To solve this, the process of transferring a Cube from G_1 to G_2 has been divided into two parts. First, edges that belong into the middle layer are transferred there. Second, the corners are oriented the right way. The first part is fairly easy and the fitness function is similar to that from $phase_0$ except for w (number of wrong *positioned* edges), i.e. edges that should be in the middle layer but are not.

$$phase_1 = 5 \cdot (2w) + c \quad (4)$$

In the second part, for each wrong positioned corner, 4 penalty points are assigned as they are more complex to correct than edges. Obviously, in order to put the Cube from G_1 to G_2 both phases described here have to be fulfilled, which yields:

$$phase_2 = 10 \cdot (4v) + phase_1 \quad (5)$$

where v represents the number of wrong *oriented* corners. The weighing factor is increased from 5 to 10 to promote a successful transformation into G_2 over a short sequence of moves.

$G_2 \rightarrow G_3$
We now have to put the remaining 8 edges in their correct orbit. The same is done for the 8 corners which also need to be aligned the right way. Thus, the colors of two adjacent corners in one circuit have to match on two faces. In G_3 the Cube will only have opposite colors on each face. Let x (number of wrong *colored* facelets) and y (number of wrong *aligned* corners), then

$$phase_3 = 5 \cdot (x + 2 \cdot y) + c . \quad (6)$$

$G_3 \rightarrow G_4(solved)$
The Cube can now be solved by only using half-turns. For the fitness function we simply count wrong colored facelets. Let z be the number of wrong colored facelets, then

$$phase_4 = 5 \cdot z + c . \quad (7)$$

To summarize, 5 different fitness functions are needed for the Thistlethwaite ES. $phase_i$ is solved if $phase_i = c, i = 0, ..., 4$ and with the properties of nested groups we can conclude, given the above, a solved Cube implies:

$$\sum_{0}^{4} phase_i = c . \quad (8)$$

Fulfilling the above equation satisfies the constraints induced by the groups $G_0, \ldots, G_4$, with the final fitness value c describing the final solution sequence length. The weight factors chosen are based on consecutive testing throughout development. The ratio depends on the size of the nested groups. Finding optimal weights presents a separate optimization problem and may be subject to future work.

4. REMARKS ON SELECTION, GROUPS AND DIVIDE AND CONQUER

In the specific case of the Rubik's Cube, the unsolvable complete solution space of $|G_C| = 4.3 \cdot 10^{19}$ using non-restricted, randomly generated mutation sequences consisting of single moves, spawned the idea of dividing the problem into smaller subproblems. The idea itself however is not exclusive to this application.

The general problem in this type of situation is to find a consistent divide and conquer strategy, equivalent to the original problem. However, oftentimes many problems already provide such in form of classical, non-ES algorithms. With this work we intend to show how such existing divide and conquer concepts can be used and transformed into heuristics suitable for adaption into fitness functions to enable quick and efficient deployment of divide and conquer EAs. Next, it is necessary to provide suitable mutation operators and selection methods. Mutation operators in our case are still randomly generated only adhering to the single moves provided by the current subgroup, which again depends on the current fitness phase. However, this only needs a minor tweak from the original idea, removing some entries from the list of single moves that can be randomly chosen from.

Finding an appropriate selection function for efficient EA design in large solution spaces is a far more challenging and, at times, creative process. Even more so when building a divide and conquer EA where essentially each phase proves to be a single, classical ES-loop and the input (starting population) of the current loop is to be the solution provided by the previous one. A first version of our Rubik's Cube ES for example would evolve until *one* individual fulfilling the current fitness phase had been found to form the starting population of the subsequent phase by duplication. However, in problems where there exist more than one solution, typically multi-dimensional solutions in multi-objective optimization, most often one of these dimensions outweighs the others in importance. In the present two-dimensional Rubik's Cube example objective dimensions are *distance_to_phase_solve* (variables v, w, x, y, z in equations (3) - (7)) and *current_sequence_length* (variable c in equations (3),(4),(6),(7),(8)) - where *distance_to_phase_solve* is the primary, to be fulfilled under all circumstances.

This property can be exploited in scenarios where the already smaller solution spaces acquired by divide and conquer are still large. Key is to provide subsequent ES-loops with a high diversity of individuals which fulfill at least the prime objective (e.g. *distance_to_phase_solve* but may - or even should - differ in the other (e.g. *current_sequence_length*). Even if some individuals with non-optimal, even relatively bad secondary objective values, form part of the starting population for the subsequent ES loop - the gain in diversity provides new search paths in the solution space and ultimately increases overall ES efficiency. Using atypically large μ and λ further helps to increase diversity.

In our exemplary ES for solving the Rubik's Cube these mechanics have been applied as follows. After some solution to a phase has

been found, the ES does not immediately start calculation of the next group-transition (which would take only this one individual as basis for further calculation) but continues evolution until at least μ *different* individuals have been found to form the start population for the next phase. To further increase diversity we used large $(\mu,\lambda) = (1000,50000)$.

5. BENCHMARKS AND CONCLUSIONS

To provide a brief performance overview 100 random scrambles of minimum length 10 and maximum length 50 were generated and and solved in 5 repetitions. Solution lengths and calculation time are of particular interest to us. The test was conducted with the TWES using $(\mu,\lambda) = (1000,50000)$, weighing factors $(5,5,5,5,5)$, mutation lengths $(5,5,13,15,17)$ and maximum generations before reset (250).

avg.	Run 1	Run 2	Run 3	Run 4	Run 5
Generations	95.72	100.63	92.71	99.66	92.22
Moves	50.67	50.32	50.87	50.23	49.46
Time(s)	321.78	381.68	393.99	312.98	287.93

Table 1: Solutions of 100 random scrambles, 5 repetitions, Thistlethwaite ES.

As seen in Table 1, the solution sequences hit an average of about 50 single moves, further demonstrating a consistent performance throughout the repetitions. Most scrambles are solved in 35-45 moves, outliers are responsible for the higher average count. Extensive additional benchmarks can be found in [3].

The benchmarks are promising, yielding comparable results to the classic TWA. Outliers calculated by TWES provide both significantly shorter and longer solutions. This is most probably due to inter-group dependencies and future focus lies on increasing our TWES' tendency to such shorter results. Instead of obtaining static solutions dictated by the lookup-table used in the classic TWA, the dynamic evolution process enables those shorter solution sequences not previously possible.

Regarding the Rubik's Cube optimization problem, our evolutionary approach is evidently competitive with the exact method it adepts. As this was the first such attempt - based on the first group theoretic exact approach using lookup-tables (Thistlethwaite) - future work promises further improvement. This algorithm only solves the classic 3^3 Rubik's Cube, just as the exact method it is based on, does. However, our modular EA can also be used to solve higher dimensional Rubik's Cubes by appropriately substituting the current fitness functions.

The next developmental step will adept approaches that reduce the number of subgroups to 3 and then 2, potentially yielding further improvement in solution sequence length. Conveniently, our implementation already provides such possibilities for extensions, enabling quick testing of different subgroup combinations

6. REFERENCES

[1] M. Borschbach, C. Grelle, S. Hauke, "Divide and Evolve Driven by Human Strategies. Simulated Evolution and Learning (SEAL)," pp. 369-373. LNCS 6457, Springer (2010)

[2] W. Boyzejko, M. Wodecki, "A Hybrid Evolutionary Algorithm for some Discrete Optimization Problems," In: Proceedings of the 5th International Conference on Intelligent Systems Design and Applications, pp. 326–331. IEEE Computer Society, Washington (2005)

[3] N. El-Sourani, "Design and Benchmark of different Evolutionary Approaches to Solve the Rubik's Cube as a Discrete Optimization Problem," Diploma Thesis, WWU Muenster, Germany (2009)

[4] N. El-Sourani, S. Hauke, M. Borschbach, "An Evolutionary Approach for Solving the Rubik's Cube Incorporating Exact Methods. Applications of Evolutionary Computations." pp. 80-90. LNCS 6024, Springer (2010)

[5] K. Florios, G. Mavrotas, D. Diakoulaki, "Solving multiobjective, Multiconstraint Knapsack Problems Using Mathematical Programming and Evolutionary Algorithms," European Journal of Operational Research 203, 14–21 (2009)

[6] A. Frey, D. Singmaster, "Handbook of Cubic Math." Enslow, Hillside (1982)

[7] M. Herdy, G. Patone, 'Evolution Strategy in Action," 10 ES-Demonstrations. Technical Report, International Conference on Evolutionary Computation (1994)

[8] H, Kociemba, "Cube Explorer," `http://kociemba.org/Cube.htm`

[9] H. Muehlenbein, T. Mahnig, "FDA - A Scalable Evolutionary Algorithm for the Optimization of Additively Decomposed Functions," Evol. Comput. 7, 353–376 (1999)

[10] M. Reid, "Cube Lovers Mailing List," `http://www.math.rwth-aachen.de/~Martin.Schoenert/Cube-Lovers/Index_by_Author.html`

[11] T. Rokicki, "Twenty-Five Moves Suffice for Rubik's Cube," `http://Cubezzz.homelinux.org/drupal/?q=node/view/121`

[12] T. Rokicki, `http://cube20.org`

[13] M.B. Thistlethwaite, "The 45-52 Move Strategy," London CL VIII (1981)

[14] E. Zitzler, "Evolutionary Algorithms for multi-objective Optimization: Methods and Applications," Penn State (1999)

Multi-objective Evolutionary Course Timetabling

A. L. Márquez * C. Gil * R. Baños * A. Fernández *

* University of Almería
Carretera de Sacramento S/N, La Cañada de San Urbano, 04120 Almería
{almarquez, cgilm, rbanos, afdezmolina}@ual.es

ABSTRACT

Multi-Objective Evolutionary Algorithms (MOEAs) are highly flexible procedures capable of producing a set of *optimal compromise* solutions called Pareto Front. These solutions represent the best values that can be obtained for each objective without reducing the optimality of the other objectives of the solution. Taking this into account, timetabling problems that are usually dealt with a weighted sum of penalization functions can be considered a multi-objective problem. This paper presents a study of the use of different MOEAs to solve several instances of a particular type of timetabling problems called Course TimeTabling (CTT).

Keywords: Multi-objective, Timetabling, MOEA

1. INTRODUCTION

Course Timetabling problems consist of the weekly planning of lectures for a set of courses. There are many formulations for this problem, which differ greatly, especially when they consider how to deal with the hard and soft constraints imposed by the problem definition. The hard constraints must be completely satisfied, while the soft constraints are considered penalizations that have to be optimized. Among the techniques used to solve this problem are Evolutionary Algorithms [1, 2], or meta-heuristics [3] such as procedures based on Tabu Search [4] or Simulated Annealing [5]. A more complete study on different timetabling problems can be found in [6], discussing several kinds of timetabling problems and different methods that could be used to solve them.

A timetable is a set of encounters organized in time. An encounter is a combination of resources (rooms, people or equipment), some of which can be specified by the problem while others must be organized as part of the solution. It has long been known that timetabling is an NP-complete problem [7], which means that there is no known method to solve it in a reasonable amount of time.

It is usually considered that the solution to be found (whether with an evolutionary algorithm, tabu search, simulated annealing, or any other technique) is a weighted sum of the values of the problem objectives (the soft constraints), effectively turning the problem into a single-objective one. On the other hand, a Pareto Front-based multiobjective approximation [8] can also be used when considering many weighted sums as several different objectives to optimize, or even defining as many objectives as there are constraints.

The remainder of this paper is organized as follows: Section 2 shows the main concepts behind multi-objective optimization, while section 3 briefly explains the basics of several MOEAs. In section 4 the problem of course timetabling is described, along with the main restrictions that apply to a particular instance. Finally, sections 5 and 6 explain the experimental results and conclusions respectively.

2. CONCEPTS IN MULTI-OBJECTIVE OPTIMIZATION

The use of Multi-Objective Optimization as a tool to solve Multi-Objective Problems (MOP) implies explaining some key concepts that are of invaluable importance. Without them it would be inaccurate to describe what a good approximation to the Pareto Front is in terms of criteria such as closeness to the Pareto set, diversity, etc [9, 10, 11, 12].

Multi-Objective Optimization is the exploration of one or more decision variables belonging to the function space, which simultaneously satisfy all constraints to optimize an objective function vector that maps the decision variables to two or more objectives.

$$minimize/maximize(f_k(s)), \forall k \in [1,K] \tag{1}$$

Each decision vector $s=\{(s_1, s_2, .., s_m)\}$ represents accurate numerical qualities for a MOP. The set of all decision vectors constitutes the *decision space*. The set of decision vectors that simultaneously satisfies all the constraints is called *feasible set (F)*. The objective function vector (f) maps the decision vectors from the decision space into a K-dimensional objective space $Z \in \Re^K$, $z=f(s)$, $f(s)=\{f_1(s), f_2(s),..., f_K(s)\}$, $z \in Z$, $s \in F$.

In order to compare the solutions of a given MOP with $K \geq 2$ objectives, instead of giving a scalar value to each solution, a partial order is defined according to Pareto-dominance relations, as detailed below.

Order relation between decision vectors: Let s and s' be two decision vectors. The dominance and incomparability relations in a minimization problem are:

$$\begin{cases} s \text{ dominates } s' \ (s \prec s') \quad iff \\ f_k(s) < f_k(s') \wedge f'_k(s) \not> f'_k(s'), \ \forall k' \neq k \in [1,K] \end{cases} \tag{2}$$

$$\begin{cases} s, \ s' \text{ are incomparable } (s \sim s') \quad iff \\ f_k(s) < f_k(s') \wedge f'_k(s) > f'_k(s'), \ k' \neq k \in [1,K] \end{cases} \tag{3}$$

Pareto-optimal solution: A solution s is called *Pareto-optimal* if there is no other $s' \in F$, such that $f(s') < f(s)$. All the Pareto-optimal solutions define the *Pareto-optimal set*, also called *Pareto Front*.

Non-dominated solution: A solution $s \in F$ is *non-dominated* with respect to a set $S' \in F$ if and only if $\nexists s' \in S'$, verifying that $s' \prec s$.

Obtaining a set of non-dominated solutions is not the only important objective when solving this kind of problem. Obtaining a wide and evenly distributed Pareto Front is also of key importance because such a set of solutions is more useful for the decision making process. This happens because a wide and evenly distributed Pareto Front h

3. IMPLEMENTED MOEAS

The following MOEAs have been used to perform the experiments needed to gather the data used in this paper:

- *NSGA-II*, Non-dominated Sorting Genetic Algorithm II [13]. It makes use of a population as well as a temporary helper population where it stores the descendant individuals. It then joins both populations and classifies them by using a fast non-dominated sorting to separate the solutions into several fronts, with a domination relationship between them. To generate the next population, only the first fronts are kept, while the other solutions are disregarded. As an estimation of solution density, the Crowding distance is calculated, in order to use a crowding comparison operator to guide the selection process towards a uniform front. In this way, the population holds the Pareto front and becomes the solution at the end of the procedure.
- *PESA*, Pareto Envelope-based Selection Algorithm [14]. This MOEA uses a hypergrid for analyzing the density information of the individuals. PESA keeps the non-dominated individuals in an archive, updating it each time a new solution is inserted by removing the old solutions that become indifferent or dominated by the new one. The archive holds the Pareto front, which becomes the solution at the end of the procedure.
- *SPEA2*, Strength Pareto Evolutionary Algorithm [15]. It uses a strength indicator in order to measure the solution quality of the individuals stored in the archive. At the end of the procedure, the archive becomes the final solution, storing the generated Pareto front. The main operations in this MOEA consist of generating the fitness of the solutions, calculating the density information for each solution within the solution set, and then truncating the archive once it becomes full, by removing the worst quality solutions in the densest areas.
- *msPESA*, Mixed Spreading PESA [16]. This MOEA is a derivative of PESA that implements a different hypergrid policy allowing the grid resolution to increase without penalizing performance. In this case, the hypergrid has one dimension less than the PESA hypergrid, so the memory requirements are greatly reduced for larger populations. The logic behind this consists of using the same number of cells in the grid as there are solutions. Ideally this would mean that as the algorithm optimizes the Pareto front, the solutions would end up evenly spread alongside the front. On inserting a solution into the archive, it performs a local search procedure in order to improve the quality of the solution, or it even inserts more than one possible solution. Inserting a new solution into the archive does not enforce a strong elitism, since all the solutions are kept, and they are only removed when the archive is full. This increases genetic variety during the first iterations of the MOEA.

4. PROBLEM DEFINITION: COURSE TIMETABLING

The implemented MOEAs use the problem proposed by Di Gaspero and Schaerf [4] , which considers q lectures $(c_1,\ldots,c_q)$, p periods $(1,\ldots,p)$ and m rooms $(r_1,\ldots,r_m)$. Each course c_i consists of l_i periods that will be scheduled in different time slots with s_i assigned students. Each room r_j has a capacity cap_j, defined by the number of available seats. There are also g lecture groups called *curricula*, such that any pair of courses of a curriculum have students in common.

The objective of the problem is to satisfy every hard constraint in each and every one of the final solutions of the problem, while the soft constraints may not be fully satisfied, deteriorating the solution quality. The following definitions show the constraints for a basic definition of this timetabling problem:

Lectures (hard) The number of lectures of course c_i must be exactly l_i.

Room Occupancy (hard) Two distinct lectures cannot take place in the same period and room.

Conflicts (hard) Lectures of courses in the same curriculum or taught by the same teacher must be scheduled at different times.

Availabilities (hard) Lecturers may not be available for some periods.

Room Capacity (soft) The number of students that attend a course must be less than or equal to the number of seats in each of the rooms that host its lectures.

Minimum Working Days (soft) The set of periods p is split in wd days of p/wd periods each (assuming that p is divisible by wd). Each period therefore belongs to a specific week day. The lectures of each course c_i must be spread over a minimum number of days d_i (with $d_i \leq l_i$ and $d_i \leq wd$).

Curriculum Compactness (soft) The daily schedule of a curriculum should be as compact as possible, avoiding isolated lectures, i.e. one lecture for a given curriculum that is not adjacent to any other lecture within the same day.

There are other categories of constraints and requirements existing on a practical level, rather than on an academic one, such as:

Lecture Management A teacher must not give lectures in more than 4 consecutive periods.

4.1. Timetabling Solver

As an initial treatment, an attempt to schedule the classes is made by sorting the rooms in descending order of available seats, which greatly helps to schedule the initialization of the Individuals (of the initial population, which has not yet been evolved). This pretreatment tries to fit all the lectures in time slots where they fit and are not violating any hard constraints. Individuals that are created from another one (descendants) clone them (they become exact copies). This behavior helps to reduce the amount of hard constraint violations.

During the *evaluation* of each Individual the violations of hard constraints are checked. In case of violation, it will most likely happen during the first generations because Individuals that comply with the hard constraints have not yet evolved. Once a violation of a hard constraint happens, then the evaluation procedure will try to correct it by randomly making additional changes to the schedule in a mutation-like manner. This will always be applied after the mutation operation. Only changes that do not produce hard constraint violations are allowed. This means that both the mutation operation and the additional corrections performed at the beginning of the evaluation process allow valid individuals to appear after a brief time interval. Once the hard constraints have been removed, all the optimization efforts will be centered on minimizing the violations of soft constraints.

Mutations follow a pattern inspired by Simulated Annealing, which means that as the number of function evaluations increases, the amount of time slot exchanges slowly decreases. At the beginning of the procedure, up to three movements are made in the mutation. At the end of the process only one change is allowed. Choosing the amount of initial maximum movements is related with performance issues, since each movement implies checking for compliance with all the hard constraints beforehand, higher numbers of

changes impair performance significantly. No crossover operation has been implemented in order to avoid generating timetables that violate hard constraints (the constraint on the amount of lectures for each course).

The objectives chosen for optimization are the sum of the values of *CurriculumCompactness*, *RoomCapacity* and *MinimumWorkingDays* as the first objective, and *CurriculumCompactness* as the second one. The intention is both to minimize the whole set of objectives, while placing a special focus on the importance of having a dense time schedule in order to reduce the problem of dead hours that is so inconvenient for both teachers and students. This also allows for easy sorting based on the first objective in order to identify the best global solutions, while in some situations it is more interesting to choose solutions with a higher penalty for CurriculumCompactness because it usually has an impact on the other constraints. Usually, the higher the penalty on CurriculumCompactness, the lower the penalty on the other objectives.

The problem instance is loaded in memory as a set of linked objects, which allows easy analysis of the relations between the different courses, rooms, curricula and constraints. With that information, the timetable is constructed as a string-indexed vector that holds a matrix of courses. The string index represents the assigned room while the matrix of courses it references is the timetable assigned to that room, using the matrix indexes to represent the time period and day of the week.

5. EXPERIMENTAL RESULTS

The results obtained by the MOEAs depend on the implementation of the individual, because the operations needed to build a proper, working, timetable are not as simple as the operations needed to optimize the ZDTn functions used as benchmarks. Furthermore, representing a timetable as well as groups of students, teacher and space constraints implies additional challenges to add to the evolutionary operations.

The configuration parameters for the experiments were 100 individuals for archive size in PESA and msPESA (10 for their working populations), 100 individuals for SPEA2 archive and work population, and 100 for NSGA-II (its helper population has the same size as the main one). The local search parameter for msPESA is to generate 10 new individuals with two moves each, and all the procedures were set to finish after performing 10^6 function evaluations.

Table 1 shows the best results found by the tabu search procedure used in [4] as a reference to compare with the results generated by the MOEAs implemented for this thesis. Note that in the original settings for the results obtained with the tabu search, there is no specification of any limits in the amount of time or number of function evaluations used in the experiments.

The experiments with MOEA have been performed by choosing the soft constraints as objectives. The assigned weights are 1x for each violation of RoomCapacity, 1x for each violation of Curriculum Compactness and 5x for each violation of MinimumWorkingDays. In the tabu search procedure, the sum of all penalizations generates the value of the solution. Therefore, the lower the sum, the better the solution.

An interesting convergence phenomenon appeared when performing the experiments: different solutions shared the same penalization score. This means that as the experiments progress further, the Pareto front tends to converge towards a local minimum, unless by chance a better timetable is found, which effectively substitutes in a few generations all the solutions with the previous penalization.

Since the Pareto dominance criterion is not met, due to the convergence to the best solution, it is far more difficult for the MOEAs to solve the timetabling problem with this criterion. This is why in table 1 the solutions are given as a single scalar (the best solution found after calculating the weight of all the penalizations, of all the solutions returned by the MOEAs), instead of giving the Pareto fronts generated by each procedure. The values given are the result of the weighted sum of the objectives, as used for the generation of the Optimal solution of the different instances..

	Test1	**Test2**	**Test3**	**Test4**
Optimal Solution	214	8	36	43
NSGA-II	364	52	99	84
SPEA2	253	59	66	97
PESA	236	28	81	68
msPESA	235	11	61	67

Table 1: Comparison of the best solution found by each procedure after 10^6 function evaluations. The optimal solution is given as reference [4].

As table 1 illustrates, PESA and msPESA are the best procedures for this problem after running 1,000,000 objective function evaluations.

6. CONCLUSIONS

Table 1 shows that msPESA is the best procedure in all situations, with the limit of 10^6 evaluations imposed on the procedures. The use of a local search procedure allowed it to improve the solution quality faster than other MOEAs. Though it does not reach optimal results, it comes close, especially for the problems *test1* and *test2*.

Given the added difficulties to obtain solutions to the timetabling problem, these results are interesting, considering how close the PESA-based methods were to the optimal solution for some of the test instances.

7. ACKNOWLEDGEMENTS

This work has been financed by the Spanish Ministry of Innovation and Science (TIN2008-01117) and the Excellence Project of Junta de Andalucía (P07-TIC02988), in part financed by the European Regional Development Fund (ERDF).

8. REFERENCES

[1] D. Corne, P. Ross, and H. lan Fang, "Evolutionary timetabling: Practice, prospects and work in progress," in *In Proceedings of the UK Planning and Scheduling SIG Workshop, Strathclyde*, 1994.

[2] B. Paechter, A. Cumming, H. Luchian, and M. Petriuc, "Two solutions to the general timetable problem using evolutionary methods," in *proceedings of the IEEE Conference on Evolutionary Computation*, vol. 1994, 1994.

[3] E. K. Burke and S. Petrovic, "Recent research directions in automated timetabling," *European Journal of Operational Research*, vol. 140, no. 2, pp. 266 – 280, 2002.

[4] L. Di Gaspero and A. Schaerf, "Neighborhood portfolio approach for local search applied to timetabling problems," *Journal of Mathematical Modeling and Algorithms*, vol. 5, no. 1, pp. 65–89, 2006. [Online]. Available: http://www.diegm.uniud.it/satt/papers/DiSc06.pdf

[5] P. Kostuch, "The university course timetabling problem with a three-phase approach," in *Practice and Theory of Automated Timetabling V*, ser. Lecture Notes in Computer

Science, E. Burke and M. Trick, Eds. Springer Berlin / Heidelberg, 2005, vol. 3616, pp. 109–125. [Online]. Available: http://dx.doi.org/10.1007/11593577_7

[6] A. Schaerf, "A survey of automated timetabling," *Artificial Intelligence Review*, vol. 13, pp. 87–127, 1999, 10.1023/A:1006576209967. [Online]. Available: http://dx.doi.org/10.1023/A:1006576209967

[7] T. Cooper and J. Kingston, "The complexity of timetable construction problems," in *Proceedings of the First International Conference on the Practice and Theory of Automated Timetabling (ICPTAT '95)*, 1995, pp. 511–522.

[8] D. Datta, C. M. Fonseca, and K. Deb, "A multi-objective evolutionary algorithm to exploit the similarities of resource allocation problems," *J. of Scheduling*, vol. 11, no. 6, pp. 405–419, 2008.

[9] K. Deb, *Multi-Objective Optimization using Evolutionary Algorithms*. John Wiley & Sons, 2001.

[10] E. Talbi, *Metaheuristics: From Design to Implementation*. New York: John Wiley & Sons, Inc., 2009.

[11] C. C. Coello, G. Lamont, and D. van Veldhuizen, *Evolutionary Algorithms for Solving Multi-Objective Problems*, 2nd ed., ser. Genetic and Evolutionary Computation. Berlin, Heidelberg: Springer, 2007.

[12] M. Voorneveld, "Characterization of pareto dominance," *Operations Research Letters*, vol. 31, no. 1, pp. 7 – 11, 2003.

[13] K. Deb, A. Pratab, S. Agrawal, and T. Meyarivan, "A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II," *IEEE Transactions on evolutionary computation*, vol. 6, no. 2, pp. 182–197, 2002.

[14] D. Corne, J. Knowles, and M. Oates, "The Pareto Envelope-based Selection Algorithm for Multiobjective Optimization," in *Proceedings of the Parallel Problem Solving from Nature VI Conference*, M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J. J. Merelo, and H.-P. Schwefel, Eds. Paris, France: Springer. Lecture Notes in Computer Science No. 1917, 2000, pp. 839–848.

[15] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the Strength Pareto Evolutionary Algorithm," Gloriastrasse 35, CH-8092 Zurich, Switzerland, Tech. Rep. 103, 2001.

[16] C. Gil, A. Márquez, R. Baños, M. Montoya, and J. Gómez, "A hybrid method for solving multi-objective global optimization problems," *Journal of Global Optimization*, vol. 38, no. 2, pp. 265–281, 2007. [Online]. Available: http://www.springerlink.com/content/f3n1284ur211p587

Automated Design of Software Architectures for Embedded Systems using Evolutionary Multiobjective Optimization

R. Li * R. Etemaadi * M.T.M. Emmerich * M.R.V. Chaudron *

* Leiden Institute of Advanced Computer Science (LIACS), Leiden University
Postbus 9512, 2300RA, Leiden, The Netherlands
{ruili, etemaadi, emmerich, chaudron}@liacs.nl

ABSTRACT

The design of software architecture for embedded system is one of the big challenges in the research field of modern software engineering. It requires software architects to address a large number of non-functional requirements that can be used to quantify the operation of system. Furthermore, these quality attributes often conflict with each other, for instance, improving system performance often needs more powerful hardware, which could increase the production cost and power consumption in the meantime. In most cases, software designers try to find a set of good architectures by hand. However because of large and combinatorial design space, this process is very time-consuming and error-prone. As a consequence, architects could easily end up with some suboptimal designs. In this paper, we introduce our AQOSA (Automated Quality-driven Optimization of Software Architecture) toolkit which can improve these aforementioned non-functional properties in an automated manner. More precisely, beginning with some initial architectures, AQOSA toolkit can use its optimizer to not only produce several alternatives, but also apply trade-off analysis to these newly created architectures according to multiple attributes of interests.

Keywords: Component-Based Software Architecture, Evolutionary Multiobjective Optimization

1. INTRODUCTION

Modern embedded systems are large and complicated and therefore difficult to develop and maintain. For example, real-time systems, which nowadays are intensively applied to application domains such as automobile and multimedia, are often built to guarantee the safety and robustness requirements. To meet these requirements makes the design of real-time systems very challenging.

Under such circumstances, software architecture which is an important field of study in software engineering receives more and more attentions in the last few years. More technically speaking, software architectures describe various aspects of the system, mostly their deployment, behavioral, and structural features. With them, designers have the opportunity to analyze the quality properties of software at a high level and thus can make optimal architectural decisions to satisfy the quality attributes at the very early architectural stage of the project.

In many cases, quality properties conflict with each other, that is, improving one quality property can have a negative impact on others, and thus to construct a system that satisfies all its requirements could be difficult. One possible solution is to use optimization techniques to generate several feasible architectures according to initial models and then select optimal solutions from all alternatives through the trade-off analysis with respect to all quality requirements.

In current practice, this process is normally performed manually to the system design. The drawback of this is that it can be time-consuming and error-prone work, especially for large and complex architectures. For complex applications, having some of this work automated could be a considerable cost saver. To this end we propose our AQOSA toolkit which was developed to automatically improve the non-functional properties of an architectural design and thus enable architects to focus on the higher-level design decisions.

The paper is organized as follows. Section 2 summaries some existing methods which are different from ours. Section 3 explains our proposed AQOSA toolkit, especially the execution procedure, in detail. The case study as well as some experimental results is presented in Section 4. Finally, conclusions and future works are given in Section 5.

2. RELATED WORK

As we emphasized at the very beginning of this paper, it is almost impossible for software architects to manually find optimal architecture designs from not only large but also discontinuous design search space. Researchers have proposed several approaches, especially some metaheuristic-based methods which can automate this process. For instance, Martens et al. [1] introduced approach which could automatically improve software architectures modelled with the Palladio Component Model based on trade-off analysis of performance, reliability, and cost.

ArcheOpterix [2] is another generic framework which optimize architecture models with evolutionary algorithms. It supports only one degree of freedom for exploration, that is allocation of software components. Besides, two quality criteria (data transmission reliability and communication overhead) are defined and the evaluation is based on formal mathematical analysis. Similar to Marten's approach, ArchiOpterix suffers from the limitation on search freedom and has chance to be trapped by some suboptimal solutions.

To alleviate this issue, our proposed AQOSA toolkit, which deploys both advanced model technology and evolutionary multiobjective optimization algorithms with specially designed genetic encoding scheme, allows not only more quality attributes but also more complex degrees of freedom like exploration of architecture topology.

3. AQOSA TOOLKIT

The detailed working process of AQOSA toolkit is illustrated in Figure 1. As can be seen, the automated optimization process starts with some initial software architectures, which could be designed

by domain experts by using some advanced model design tools. Next, these architectures are evaluated and corresponding multiple quality criteria of interests are obtained. More specific, processor utilization, data flow latency, and cost metrics are addressed in this study. At the current stage, the simulation-based approach[1] is used for AQOSA evaluator. Note that the precision of evaluation is highly dependent on the modeling details and the features supported by simulator.

As mentioned earlier, some conflicting quality attributes, such as utilization and cost, are often involved in performance analysis. Thus the domination principle could be adopted by evolutionary optimizer for doing trade-off analysis on quality attributes which are extracted through an extractor based on our performance metrics. Some good architectures are then selected from current available solutions. Furthermore, the evolutionary optimizer could automatically produce new candidate architectures by using reproduction operators like "crossover" and "mutation".

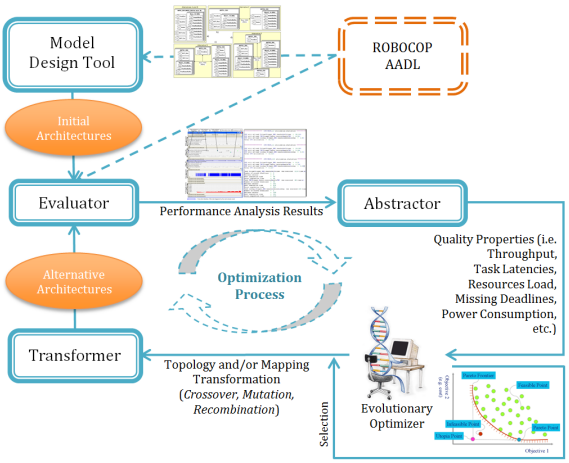


Figure 1: The detailed working scheme of AQOSA (**A**utomated **Q**uality-Driven **O**ptimization of **S**oftware **A**rchitecture) toolkit.

Next, we will explain some key components and related techniques in detail.

3.1. Modeling and Evaluation Engine

For software architecture modeling, as a natural extension of previous work [3] AQOSA integrates ROBOCOP [4] (Robust Open Component Based Software Architecture for Configurable Devices Project) modeling language. Furthermore, AQOSA also supports AADL [5] (Architecture Analysis & Design Language) which is now widely recognized industrial standard in modeling embedded and real-time architectures. The architect can easily design the initial architecture in OSATE (Open Source AADL Tool Environment) and then import it into AQOSA framework. To use ADeS [6] as the core part of our AQOSA simulation engine, we made some modifications of ADeS in scheduling and added new features for data flow latencies evaluating. More specifically speaking, our evaluation engine first loads an AADL model and creates necessary objects for simulation. After that, it generates system events based on the behaviour annex of the model and follow the events through the model connections till end of flows. For complex and concurrent events, the scheduling module decides which process can take the processor.

At present, we implement three quality properties: processor utilization, data flow latency and architecture cost. By design, AQOSA toolkit can be easily extended to support other quantitative quality criteria of software architectures by introduce new evaluation plug-ins, i.e. for communication lines loads evaluation, we just needed to add a new listener which implements the measurement of the bus load to our simulation engine. Another advantage of AQOSA is that it provides some very flexible API for the interaction between evaluator and various optimization frameworks such as Opt4J and JMetal[2].

[1] As compared to analysis-based approach.

3.2. Evolutionary Optimizer

3.2.1. *Evolutionary multiobjective optimization*

Evolutionary multiobjective optimization (EMO) [7] derives from single objective evolutionary optimization (EO) algorithms and is recognized as a fast growing fields of research. It is relatively simple to implement and wide-spread applicable. In this work, two representative multiobjective optimization algorithms (NSGAII [8] and SPEA2 [9]) from literatures are chosen and applied to one architecture design task for the car radio navigation (CRN) system.

3.2.2. *Search problem formulation*

From EMO algorithm perspective, architecture design problem can be generalized as following optimization task (see Equation 3.2.2):

$$\begin{aligned} &\min\ f_m(x), \quad m = 1,2,\ldots,M \\ &s.t.\ g_j(x) \gtrsim 0 \quad j = 1,2,\ldots,N \end{aligned} \tag{1}$$

Here, x is a solution and can be of any domain, e.g., real or binary. In the given context, x could be a valid architecture from embedded system design domain. For each solution x, there exists $m = 3$ objectives, i.e. f_1 : Processor utilization, f_2 : Cost, and f_3 : Data flow latency. $g_j(x)$ represents a number of constraints which any feasible solution must satisfy. The aim is not only provide one optimal solution but rather to provide a broad variety of nondominated solutions representing trade-offs in the three objectives.

3.2.3. *Generic degree of freedom to exploration*

With specially designed genotype representation, the following degrees of freedom to exploration are implemented: (1) System hardware topology (hypergraph), i.e. processor/bus can be added or removed from the system, (2) Allocation of service instances, (3) Replacement between different hardwares, i.e. one component can be replaced by its counterparts from available hardware repository. Figure 2 shows three system topologies which are supported and valid for car radio navigation (CRN) architecture design (i.e. case study in Section 4).

4. CASE STUDY AND EXPERIMENTAL RESULTS

4.1. Car Radio Navigation System

To validate our proposed AQOSA toolkit, we applied it to one benchmark application - the car radio navigation (CRN) system [10]. The CRN system is constructed according to the component-based paradigm. An overview of the software architecture is depicted in Figure 3.

As can be seen, the CRN system contains three major functional blocks:

[2] `http://opt4j.sourceforge.net` and `http://jmetal.sourceforge.net`

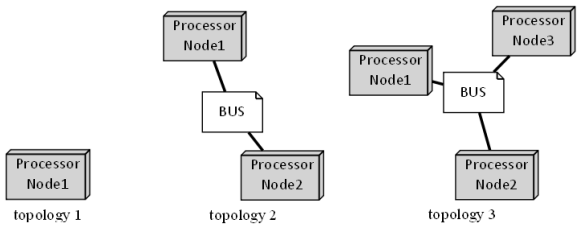


Figure 2: Possible topologies supported by genotype representation: Single processor node (left), Two processor nodes with single bus (middle), and Three processor nodes with single bus (right).

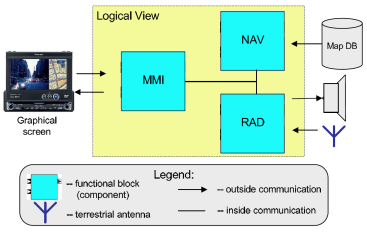


Figure 3: Overview of the car radio navigation system functionality.

- The Man-Machine Interface (MMI), that takes care of all interactions with the end-user, such as handling key inputs and graphical display output.
- The Navigation functionality (NAV) is responsible for destination entry, route planning and turn-by-turn route guidance giving the driver visual advices. The navigation functionality relies on the availability of map database and positioning information.
- The Radio functionality (RAD) is responsible for tunner and volume control as well as handling of traffic message channel information services.

The major challenge is to determine a set of optimal architectures with respect to quality attributes such as processor utilization, data flow latency, and cost. Technically speaking, we investigate how to distribute these aforementioned functionalities over the available resources (processor node in Figure 2) to meet some global requirements. Vector representation in Figure 4 illustrates how the genotype is used to describe possible architecture topologies (Figure 2) as well as mapping of services.

4.2. Experimental Setup and Results

The experimental setup is as follows: two standard evolutionary multiobjective optimization algorithms from Opt4J, Non-dominated Sorting Genetic Algorithm (NSGA-II) and Strength Pareto Evolutionary Approach 2 (SPEA2), will be used. Furthermore, the following parameter settings are adopted: initial population size: 50, parent population size: 25, number of offspring: 25, archive size: 100, number of generation: 50, crossover rate is set to 0.95, constant mutation probability is 0.01. For each algorithm we run

MMI	NAV	RAD	Node1	Node2	Node3	Bus	0/1	0/1	0/1
Allocation of Functionalities			Types of Processors (Repository size: 16)			Type of Bus (Respository size: 8)	Incidence Matrix (Bus and Nodes)		

Figure 4: Genotype vector for possible software architectures representation (884,736 possibilities).

AQOSA 20 runs ($\approx$ 10 hours). The resulting archive of optimal solutions can be visualized in the 3-D Pareto front with respect to processor utilization, cost, and data flow latency in Figure 5.

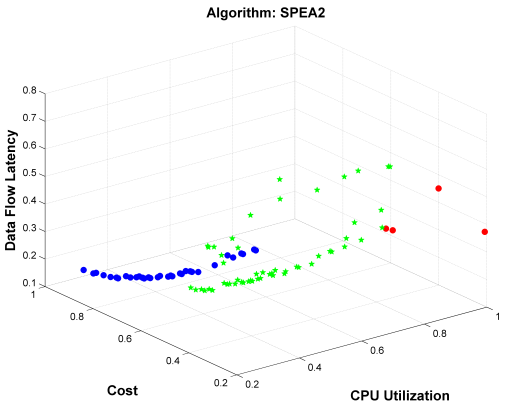


Figure 5: Resulting Pareto front approximations of archive population (non-dominant solutions) after 50 generations of one typical run of SPEA2. Colors are used to distinguish between different found architecture topologies.

An interesting finding is that resulting pareto front consists of three segmentation (with clearly gap in between). This could be the result of discontinuities in the search space caused by structural transitions. By identifying and mapping each individual from archive back to corresponding design architecture, solutions from same segmentation share the same architectural topology[3] (i.e. Figure 2). This discovery is consistent with our understanding of CRN system, for instance, solutions with topology 3 (solutions with blue color) normally have lower processor utilization and higher cost for the hardware. On the contrary, solutions with topology 1 (red color) have higher processor utilization and lower cost.

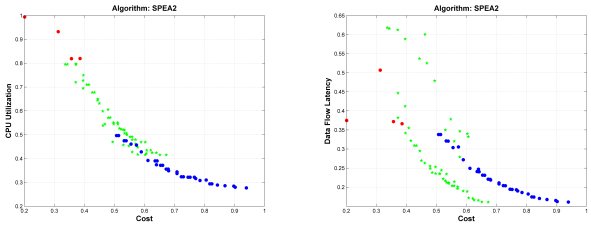


Figure 6: Plot between two objectives of archive population (non-dominant solutions): Cost vs. Processor utilization (left) and Cost vs. Data flow latency (right).

The 2-D plot of two quality attributes is presented in Figure 6. In this way, the software architect can make trade-off decision much easier. For instance, the left plot shows the processor utilization over the cost per candidate architecture while the right one indicates the data flow latency over cost. There is no obvious conflict between processor utilization and data flow latency and the corresponding plot is excluded here. Further more, both the attainment surface of one typical run of SPEA2 and the box-plots of the hypervolume indicator [11] for ref. point $(1,1,1)^T$ of archive population for NSGA-II, SPEA2, and random search over 20 runs are presented in Figure 7

From figure 7 (left), it gets clear that final solutions from archive are mutually non-dominated with respect to three quality attributes investigated. Another observation is that NSGA-II and SPEA2 show the comparable performance with each other (student's t-test

[3] All three algorithms which we studied show the same behaviour.

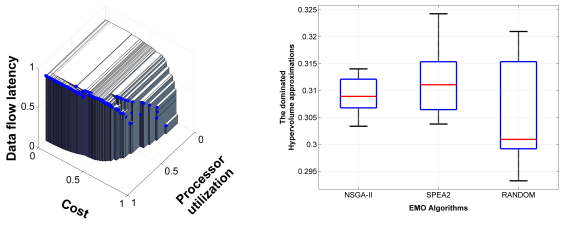


Figure 7: The dominated Hypervolume approximation of one typical run of SPEA2 (left) and the box-plots of the hypervolume indicator for NSGA-II, SPEA2, and random search on CRN design problem over 15 runs (right).

with 1% confidence level), and the results are very similar. Random search, by contrast, shows worst performance.

5. CONCLUSIONS AND OUTLOOK

We presented so-called AQOSA (Automated Quality-driven Optimization of Software Architecture) toolkit. It not only can help software architects to reduce the workload for modeling and evaluating real-world problems, but also can automatically improve quality attributes by using evolutionary multiobjective optimizers. We applied AQOSA on the car radio navigation (CRN) system. The preliminary results are very promising.

For future research several questions are of interest: First, more challenging application (i.e., from automobile industry) will be modeled and tested by using AQOSA. Secondly, besides aforementioned attributes which we studied in this work other non-functional qualities such as power consumption and safety will be integrated. Algorithms such as SMS-EMOA [12] are also worth investigating for the resulting many-objective problems.

6. ACKNOWLEDGEMENTS

This work has been supported by the Dutch national project OMECA (Optimization of Modular Embedded Computer-vision Architectures) and European project SCALOPES (an ARTEMIS project on SCalable LOw Power Embedded platformS).

7. REFERENCES

[1] A. Martens, H. Koziolek, S. Becker, and R. Reussner, "Automatically improve software architecture models for performance, reliability, and cost using evolutionary algorithms," in *Proceedings of the first joint WOSP/SIPEW international conference on Performance engineering*, 2010, pp. 105–116.

[2] A. Aleti, S. Björnander, L. Grunske, and I. Meedeniya, "Archeopterix: An extendable tool for architecture optimization of AADL models," in *ICSE 2009, MOMPES Workshop 2009, May 16, 2009, Vancouver, Canada*, 2009, pp. 61–71.

[3] R. Li, M. R. Chaudron, and R. C. Ladan, "Towards automated software architectures design using model transformations and evolutionary algorithms," in *GECCO (Companion)*. ACM, 2010, pp. 2097–2098.

[4] E. Bondarev, M. R. Chaudron, and P. de With, "A process for resolving performance trade-offs in component-based architectures," in *Component-Based Software Engineering*, ser. LNCS, vol. 4063, 2006, pp. 254–269.

[5] P. H. Feiler, D. Gluch, and J. J. Hudak, "The architecture analysis & design language (AADL): An introduction," Carnegie Mellon University, Technical Report CMU/SEI-2006-TN-011, 2006.

[6] R. S. Jean-François Tilman, Amélie Schyn, "Simulation of system architectures with AADL," in *Proceedings of 4th International Congress on Embedded Real-Time Systems, ERTS 2008.*, 2008.

[7] K. Deb, "Multiobjective optimization," J. e. a. Branke, Ed. Springer-Verlag, 2008, ch. Introduction to Evolutionary Multiobjective Optimization, pp. 59–96.

[8] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan, "A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II," in *Parallel Problem Solving from Nature PPSN VI*, ser. LNCS, 2000, vol. 1917, pp. 849–858.

[9] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the Strength Pareto Evolutionary Algorithm for Multiobjective Optimization," Tech. Rep., 2002.

[10] E. Wandeler, L. Thiele, M. Verhoef, and P. Lieverse, "System architecture evaluation using modular performance analysis: a case study," *Int J Softw Tools Technol Transfer (STTT)*, vol. 8, no. 6, pp. 649–667, 2006.

[11] E. Zitzler, L. Thiele, M. Laumanns, C. Fonseca, and V. da Fonseca, "Performance assessment of multiobjective optimizers: an analysis and review," *IEEE Trans. on Evolutionary Computation*, vol. 7, no. 2, pp. 117–132, April 2003.

[12] N. Beume, B. Naujoks, and M. Emmerich, "SMS-EMOA: Multiobjective selection based on dominated hypervolume," *European Journal of Operational Research*, vol. 181, no. 3, pp. 1653–1669, 2007.

New Characterizations for Subfamilies of Chordal Graphs

L. Markenzon [*1] P.R.C. Pereira [†] C.F.E.M. Waga [‡]

[*] NCE - Universidade Federal do Rio de Janeiro
P. O. Box: 2324, RJ, Brazil 20010-974
markenzon@nce.ufrj.br

[†] Instituto Militar de Engenharia
Praça General Tibúrcio, 80, Rio de Janeiro, Brazil 22290-270
prenato@ime.eb.br

[‡] IME - Universidade do Estado do Rio de Janeiro
Rua São Francisco Xavier, 524, Rio de Janeiro, Brazil, 20550-900
waga@ime.uerj.br

ABSTRACT

In this paper, we give new characterizations for some subfamilies of chordal graphs, such as k-intercats and SC k-trees, based on properties of their minimal vertex separators. We also establish the relationship among these families and interval graphs.

Keywords: Chordal graph, k-tree, ur-chordal

1. INTRODUCTION

Chordal graphs are an extensively studied class of graphs, as their peculiar clique-based structure allows a more efficient solution for many algorithmic problems. The investigation of new properties of the family brings up the possibility of solving problems more efficiently, with a different approach.

In this context, the minimal vertex separators play a decisive role. Their determination has been already studied in at least two recent papers [1, 2]. The presentation of a very simple algorithm [3] to perform this task renews the chance to find better results for several problems. Based on properties of the minimal vertex separators of chordal graphs and their multiplicities, we propose in this paper new characterizations for some known subfamilies of chordal graphs such as k-intercats and SC k-trees, which generalizes mops and maximal planar chordal graphs. The new structural characterizations lead to simple and efficient recognition algorithms. We are also able to prove inclusion relations among these families and other subfamilies of chordal graphs such as interval graphs.

2. BACKGROUND

Basic concepts about chordal graphs are assumed to be known and can be found in Blair and Peyton [4] and Golumbic [5]. In this section, the most pertinent concepts are reviewed.

Let $G = (V,E)$ be a graph, with $|E| = m$, $|V| = n > 0$. The *set of neighbors* of a vertex $v \in V$ is denoted by $Adj(v) = \{w \in V \mid (v,w) \in E\}$. For any $S \subseteq V$, we denote $G[S]$ the subgraph of G induced by S. S is a *clique* when $G[S]$ is a complete graph. A vertex v is said to be *simplicial* in G when $Adj(v)$ is a *clique* in G.

A subset $S \subset V$ is a *separator* of G if two vertices in the same connected component of G are in two distinct connected components of $G[V-S]$. The set S is a *minimal separator* of G if S is a separator and no proper set of S separates the graph. A subset $S \subset V$ is a *vertex separator* for non-adjacent vertices u and v (a *uv*-separator) if the removal of S from the graph separates u and v into distinct connected components. If no proper subset of S is a *uv*-separator then S is a *minimal uv-separator*. When the pair of vertices remains unspecified, we refer to S as a *minimal vertex separator*. It does not necessarily follow that a minimal vertex separator is also a minimal separator.

The next theorem presents a characterization of chordal graphs in terms of minimal vertex separators.

Theorem 1. [5] *A graph is chordal if and only if every minimal vertex separator of it induces a clique.*

The *clique-intersection graph* of a chordal graph G is the connected weighted graph whose vertices are the maximal cliques of G and whose edges connect vertices corresponding to non-disjoint maximal cliques. Each edge is assigned an integer weight, given by the cardinality of the intersection between the maximal cliques represented by its endpoints. Every maximum-weight spanning tree of the clique-intersection graph of G is called a *clique-tree* of G.

Theorem 2. [4] *Let $G = (V,E)$ be a chordal graph and $T = (V_T, E_T)$ a clique-tree of G. The set $S \subset V$ is a minimal vertex separator of G if and only if $S = Q' \cap Q''$ for some edge $(Q', Q'') \in E_T$.*

Observe that the set of minimal vertex separators related to one clique-tree is actually a multiset, since the same minimal vertex separator can appear several times. Blair and Peyton [4] proved that, for a chordal graph G, the same multiset is always obtained.

Theorem 3. *Let $G = (V,E)$ be a chordal graph. The multiset $\mathscr{S}^*$ of the minimal vertex separators of G is the same for every clique-tree of G.*

From Theorem 3 it is clear that $|\mathscr{S}^*| = \ell - 1$, being ℓ the number of maximal cliques of G. We define the *multiplicity* of the minimal vertex separator S, denoted by $\mu(S)$, as the number of times that S appears in $\mathscr{S}^*$. The set of minimal separators $\mathbf{S}$ ($\mathscr{S}^*$ without repetitions) has cardinality η.

Two important subfamilies of chordal graphs, the k-trees and the interval graphs, can be defined as follows [6].

Definition 1. *A k-tree, $k > 0$, can be inductively defined as follows:*

[1] Partially supported by grant 305372/2009-2, CNPq, Brazil.

- *Every complete graph with k vertices is a k-tree.*
- *If $G = (V,E)$ is a k-tree, $v \notin V$ and $Q \subseteq V$ is a k-clique of G, then $G' = (V \cup \{v\}, E \cup \{\{v,w\} | w \in Q\})$ is also a k-tree.*
- *Nothing else is a k-tree.*

The simplicial vertices of a k-tree are also called *k-leaves*.

Definition 2. *An interval graph is the intersection graph of a set of intervals on the real line. It has one vertex for each interval in the set, and an edge between every pair of vertices corresponding to intervals that intersect.*

3. GENEALOGY OF CHORDAL GRAPHS

Interval graphs and k-trees are well known in the literature. Our goal is to establish the relation among these families and three other genealogical branches of chordal graphs. The first branch, defined by Proskurowski [7], is the family of k-caterpillars and its descendent, the k-intercats. The second one, defined by Kumar and Madhavan [8], is the family of *ur*-chordal graphs and its descendent, the *ur*-interval graphs. The last one, defined by Markenzon *et al.* [9], is the family of SC k-trees and its descendent, the k-path graphs. The definitions of all these families are reviewed in this section.

Kumar and Madhavan defined several families based on structural properties of the clique-tree. We are going to focus on two of these families.

Definition 3. [8] *A chordal graph is called uniquely representable chordal graph (briefly ur-chordal graph) if it has exactly one clique tree. An interval graph that is uniquely representable is called an ur-interval graph.*

Theorem 4 presents a characterization of *ur*-chordal graphs.

Theorem 4. [8] *Let $G = (V,E)$ be a connected chordal graph. G is an ur-chordal graph if and only if (i) there is no proper containment between any two minimal vertex separators and (ii) all minimal vertex separators have multiplicity 1.*

The concept of a k-path appeared first in [10], as a generalization of paths. It is the base of the formal definition of k-path graphs.

Definition 4. [10] *In a graph $G = (V,E)$, a k-path of length $p > 0$ is a sequence $\langle B_0, C_1, B_1, C_2, B_2, \ldots, C_p, B_p \rangle$, where:*

- *$B_i \subset V$, $0 \leq i \leq p$, are distinct k-cliques of G;*
- *$C_i \subseteq V$, $1 \leq i \leq p$, are distinct $(k+1)$-cliques of G;*
- *$B_{i-1} \subset C_i$, $B_i \subset C_i$ and no other k-clique B_j, $0 \leq j \leq p$, $j \neq i-1$ and $j \neq i$, is a subset of C_i, $1 \leq i \leq p$.*

Definition 5. [9] *Let $G = (V,E)$ be a k-tree with $n > k$ vertices. G is a k-path graph if there is a maximal k-path $\langle B_0, C_1, B_1, \ldots, C_p, B_p \rangle$, $p > 0$, such that the subgraph of G induced by $C_1 \cup \ldots \cup C_p$ is isomorphic to G.*

Observe that k-paths and k-path graphs are often confused. However, for $k > 1$, the concepts can be quite distinct; actually, there are k^2 different maximal k-paths in a k-path graph; the k-cliques $B_1, \ldots, B_{p-1}$ belong to all maximal k-paths.

The recognition of a k-tree as a k-path graph can be easily accomplished, due to the characterization provided by the next theorem.

Theorem 5. [9] *Let $G = (V,E)$ be a k-tree with $n > k+1$ vertices. G is a k-path graph if and only if G has exactly two simplicial vertices.*

The inductive definition of a *simple-clique k-tree* (*SC k-tree*) follows. Note that its construction is similar to the one presented in Definition 1, except that it is more restrictive. It is worth to mention two particular cases of the family: SC 2-trees are the maximal outerplanar graphs (*mops*) and SC 3-trees, the maximal planar chordal graphs.

Definition 6. [9] *A Simple Clique k-tree (SC k-tree), $k > 0$, can be inductively defined as follows:*

- *Every complete graph with $k+1$ vertices is a SC k-tree.*
- *If $G = (V,E)$ is a SC k-tree $v \notin V$ and $Q \subset V$ is a k-clique of G not previously chosen in the existing SC k-tree, then $G' = (V \cup \{v\}, E \cup \{\{v,w\} | w \in Q\})$ is also a SC k-tree.*
- *Nothing else is a SC k-tree.*

The definition of k-caterpillars and k-intercats is also based on the concept of k-paths and were presented in [7]. Firstly we define the *body* of a graph.

Definition 7. *Let G be a chordal graph and H the set of its simplicial vertices. We call $G[V-H]$, the subgraph induced by $V-H$, the body of G.*

Definition 8. *Let G be a k-tree and P its body. G is a k-caterpillar if P is: (i) an empty graph or (ii) a complete graph or (iii) a k-path graph.*

Definition 9. *Let G be k-caterpillar and P its body. G is an interior k-caterpillar (k-intercat, for short) if: (i) P is an empty graph or (ii) P is a complete graph with k vertices or (iii) there is a maximal k-path in P $\langle B_0, C_1, B_1, \ldots, C_p, B_p \rangle$ such that for any k-leaf v of G, v is adjacent to all vertices of some k-clique B_i.*

4. NEW CHARACTERIZATIONS

In this section we present three theorems that establish the relations among all the families mentioned. It is interesting to note that these theorems actually provide new characterizations for some of these families such as the SC k-trees and the k-intercats. For the latter, the characterization leads to a simple linear recognition algorithm.

Theorem 6. *Let $G = (V,E)$ be a k-tree with $n > k+1$ vertices. The three following statements are equivalent:*

1. G is a SC k-tree.

2. All minimal vertex separators of G have multiplicity one, that is $\eta = n-k-1$.

3. G is an ur-chordal graph.

Proof:

($1 \Longleftrightarrow 2$) Definition 1 provides the construction of a k-tree G. It is possible to build at the same time the clique-tree of G: each new vertex v, together with the k-clique Q, chosen in the current graph, forms a new maximal clique and, consequently, a new vertex of the clique-tree. Two maximal cliques of G have the same subset Q; so, Q is a minimal vertex separator of G. By Definition 6, in a SC k-tree Q can be chosen only once.

($2 \Longleftrightarrow 3$) Kumar and Madhavan [8] proved that a chordal graph is uniquely representable if and only if (i) there is no proper containment between any two minimal vertex separators and (ii) all minimal vertex separators have multiplicity 1. By Rose [11], every minimal vertex separator of a k-tree has cardinality k; so, there is no containment between them.

□

The concept of asteroidal triple is fundamental for a compact characterization of interval graphs. Three vertices u,v,w of G form an *asteroidal triple (AT)* if for every pair of them there is a path connecting the two vertices that avoids the neighborhood of the remaining vertex. Brandstadt *et al.* [6] refer to the following theorem:

Theorem 7. *G is an interval graph if and only if G is chordal and contains no AT.*

Besides the efficient recognition of k-intercats, the next theorem also shows that a k-tree is an interval graph if and only if it is a k-intercat.

Theorem 8. *Let G be a k-tree with $\eta \geq 2$ minimal vertex separators and P its body. The three following statements are equivalent:*

1. *G is a k-intercat.*
2. *G is an interval graph.*
3. *P has exactly $\eta - 2$ minimal vertex separators.*

Proof:

(1⇒2) Let $\langle B_0, C_1, B_1, \ldots, C_p, B_p \rangle$ be a longest k-path of G. Let G' be the subgraph of G induced by the vertices of this k-path. G' has two simplicial vertices (Theorem 5): $v' \in B_0$ and $v'' \in B_p$. As G' is a k-path graph, it is an interval graph [12]. Let H be the set of simplicial vertices of G. By definition, each $w \in H$, except v' and v'', is adjacent to a k-clique B_i, $1 \leq i \leq p-1$.

Let us add a vertex $v \in H$ to G' and suppose, by absurd, that vertex v with vertices u and w of G' form an asteroidal triple. Vertex v is adjacent to some B_i, $1 \leq i \leq p-1$. As $B_i = C_i \cap C_{i+1}$, B_i is a minimal vertex separator of G'. The removal of B_i separates G' in two components. Two cases can happen:

case 1) B_i separates u and w. As B_i is a minimal vertex separator, all paths linking u and w cannot avoid the neighborhood of v.

case 2) After removing B_i, u and w belong to the same connected component. Since u and w are not adjacent, they belong to different maximal cliques of G'. The clique-tree of a k-path graph is a path. As v is adjacent to B_i, the vertex corresponding to the new maximal clique C' can be inserted between cliques C_i and C_{i+1}. Suppose, without loss of generality, that $u \in C_q$ and $u \notin C_{q+1}$, $i < q$. Suppose also that $w \in C_t$, $t > q$. B_q separates u and w and it belongs to the neighborhood of u. All paths between v and w cannot avoid B_q. So, it is impossible to have an asteroidal triple and G is an interval graph.

(2⇒3) Let $T = (\{Q_1, ..., Q_p\}, \{(Q_i, Q_{i+1}) | 1 \leq i \leq p-1\})$ be a clique-tree of G such that T is a path. We know that simplicial vertices belong to just one maximal clique, and we know that in a k-tree at most one simplicial vertex belongs to a maximal clique. So, $Q_1 = v' \cup S_1$ and $Q_p = v'' \cup S_p$.

The body P of G (and its clique-tree) is obtained by the removal of all simplicial vertices of G. This task will be performed in two steps. Firstly, we remove all vertices of $H - \{v', v''\}$, being H the set of simplicial vertices of G. Let $v \in Q_i$, $i \neq 1, p$, be a simplicial vertex and $Q_i = \{v\} \cup S_i$. As $|Q_i \cap Q_{i+1}| = |Q_i \cap Q_{i-1}| = k$, then $Q_{i-1} \cap Q_{i+1} = S_i$. So, the maximal clique Q_i does not exist anymore and so the corresponding vertex of the clique-tree; (Q_{i-1}, Q_{i+1}) is a new edge in the clique-tree. Observe that S_i is a minimal vertex separator (because it is an edge) of the clique-tree of the remaining graph. After the removal of all vertices of $H - \{v', v''\}$, the remaining graph is a k-path graph.

Secondly, we remove vertices v' and v''. All minimal vertex separators of a k-path graph are distinct. So, after the removal of these two vertices, the maximal cliques Q_1 and Q_p do not belong to P and the two minimal vertex separators S_1 and S_p are not minimal vertex separators of P.

(3⇒1) By Definition 9, P is subgraph of G; G is a k-tree, so P is also a k-tree. As all simplicial vertices of G were removed, a vertex of P belongs to at least one minimal vertex separator of G. Let v be a simplicial vertex of P. The minimal vertex separator that contains v in G is not a minimal vertex separator of P. In a k-tree, there are not adjacent simplicial vertices. So, as P has $\eta - 2$ minimal vertex separators, P has exactly two simplicial vertices and P is a k-path graph.

Let $\langle B_0, C_1, B_1, \ldots, C_p, B_p \rangle$ be a maximal k-path of G. Observe that $\langle B_1, C_2, B_2, \ldots, C_{p-1}, B_{p-1} \rangle$ is a maximal k-path of P and only B_1 and B_{p-1} are not minimal vertex separators of P. So, all simplicial vertices of G are adjacent to a k-clique B_i, $1 \leq i \leq p-1$, *i.e.*, G is a k-intercat.

□

By definition, we know already that *ur*-interval graphs are interval graphs; in [11], Pereira *et al.* proved that k-path graphs are also interval graphs. Recalling that an interval graph has a clique-tree that is a path, the following theorem shows that the k-path graphs actually satisfy the definition of three important families.

Theorem 9. *A graph G is a k-tree, an interval graph and an ur-chordal graph if and only if it is a k-path graph.*

Proof:

(⇒) By Theorem 6, a k-tree that is an *ur*-chordal has all minimal vertex separators with multiplicity one. So, a simplicial vertex of G is adjacent to exactly one minimal vertex separator B of G and B is not a minimal vertex separator of P. By Theorem 8 the body P of a k-tree that is an interval graph has $\eta - 2$ minimal vertex separators. So G has exactly two simplicial vertices, i.e, G is a k-path graph.

(⇐) By definition a k-path graph is a k-tree and Pereira *et al.* proved that k-path graphs are interval graphs. Let $\langle B_0, C_1, B_1, \ldots, C_p, B_p \rangle$ be a maximal k-path of G. Observe that $B_1, B_2, \ldots, B_{p-1}$ are the $\eta = n - k - 1$ minimal vertex separators of G. By Theorem 6 G is an *ur*-chordal graph.

□

Figure 1 shows all results covered in this paper, showing the hierarchy of subfamilies. Note that an arrow indicates that a family is subfamily of its parent. If more than one arrow arrives at a node, the family is the intersection of the parent families.

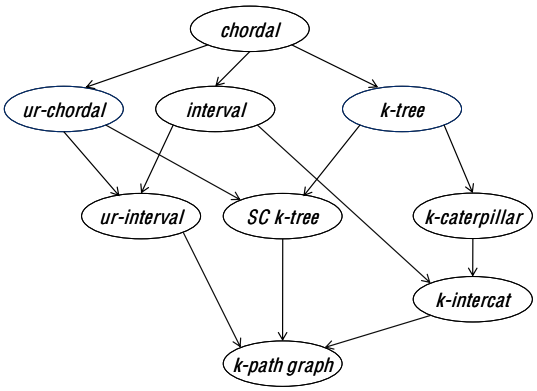


Figure 1: Relationship among k-trees, *ur*-chordal and interval graphs.

5. ACKNOWLEDGMENT

This work is supported by grant 305372/2009-2, CNPq, Brazil.

6. REFERENCES

[1] L.S. Chandran and F. Grandoni, "A linear time algorithm to list the minimal separators of chordal graphs", *Discrete Math.*, vol.306, pp. 351-358, 2006.

[2] P.S. Kumar and C.E.V. Madhavan, "Minimal vertex separators of chordal graphs", *Discrete Appl. Math.*, vol. 89, pp. 155-168, 1998.

[3] L. Markenzon and P.R.C. Pereira, "One-phase algorithm for the determination of minimal vertex separators of chordal graphs", *Internat. Trans. in Oper. Res.*, vol. 17, pp. 683-690, 2010.

[4] J.R.S. Blair and B. Peyton, "An introduction to chordal graphs and clique trees", in *Graph Theory and Sparse Matrix Computation*, IMA vol. 56, 1993, pp. 1-29.

[5] M.C. Golumbic, *Algorithmic Graph Theory and Perfect Graphs*, 2^{nd} edition, Academic Press, New York, 2004.

[6] A. Brandstädt, V.B. Le, and J. Spinrad, *Graph Classes - a Survey*, SIAM Monographs in Discrete Mathematics and Applications, 1999.

[7] A. Proskurowski, "Separating subgraphs in *k*-trees: cables and caterpillars", *Discrete Math.*, vol.49, pp. 275-285, 1984.

[8] P.S. Kumar and C.E.V. Madhavan, "Clique tree generalization and new subclasses of chordal graphs", *Discrete Appl. Math.*, vol.117, pp. 109-131, 2002.

[9] L. Markenzon, C.M. Justel, and N. Paciornik, "Subclasses of *k*-trees: characterization and recognition", *Discrete Appl. Math.*, vol.154, pp. 818-825, 2006.

[10] L.W. Beineke and R.E Pippert, "Properties and characterizations of *k*-trees", *Mathematika*, vol.18, pp. 141-151, 1971.

[11] D.J. Rose, "On simple characterizations of *k*-trees", *Discrete Math.*, vol. 7, pp. 317-322, 1974.

[12] P.R.C. Pereira, L. Markenzon, and O. Vernet, "A clique-difference encoding scheme for labelled *k*-path graphs", *Discrete Appl. Math.*, vol.156, pp. 3216-3222, 2008.

Efficient Algorithms for Regionalization: an Approach Based on Graph Partition

Gustavo Silva Semaan * José André de Moura Brito † Luiz Satoru Ochi *

* Instituto de Computação - Universidade Federal Fluminense, IC-UFF
Rua Passo da Pátria 156 - Bloco E - 3º andar, São Domingos, CEP: 24210-240, Niterói, RJ, Brasil
{gsemaan, satoru}@ic.uff.br

† Escola Nacional de Ciências Estatísticas - Instituto Brasileiro de Geografia e Estatística, ENCE-IBGE
Rua André Cavalcanti 106, sala 403, CEP: 20231-50, Rio de Janeiro, RJ, Brasil
jose.m.brito@ibge.gov.br

ABSTRACT

This paper proposes new approaches based on the GRASP and Evolutionary algorithms for the resolution of a specific regionalization problem. This problem can be mapped on a capacity and connectivity graph partition problem. A review of literature showing that the algorithms work only with the edges of the Minimum Spanning Tree is presented. In this case, the algorithms act on the original graph, in order to increase the possibilities of vertex migration. Results obtained from the application of such algorithms over a set of real data suggested that the use of original graphs through them is a new efficient way to solve this problem.

Keywords: Graph Partition Problem, Clustering, Regionalization, Metaheuristics

1. INTRODUCTION

According to [1, 2], regionalization is a clustering procedure applied to spatial objects with a geographic representation, which groups them into homogeneous contiguous regions and Cluster Analysis is a multivariate technique used to group objects together based on a selected similarity measure, in such way that objects in the same cluster are very similar and objects in different clusters are quite distinct [3].

Considering a given set with n objects $X = \{x_1, .., x_n\}$, it must extract partitions from the set X in k different clusters C_i, respecting the following three conditions:

$$\bigcup_{i=1}^{k} C_i = X$$
$$C_i \neq \emptyset, 1 \leq i \leq k$$
$$C_i \cap C_j = \emptyset, 1 \leq i, j \leq k, i \neq j$$

The cluster analysis is a fundamental technique to experimental sciences in which the classification of elements into groups is desirable . As examples of these fields it is possible to cite: biology, medicine, economy, psychology, marketing, statistic among others [4].

2. GRAPH PARTITION PROBLEM

Several clustering problems can be mapped on graph partition problems.This consists in grouping the vertexes of the graphs in different subsets (clusters), according to their similarities, by using a fitness function [1, 5, 6]. Moreover, this regionalization problem considers the following restrictions:

- *Connectivity*: the vertexes grouped in each cluster must be connected.
- *Minimum Capacity*: associated total to one of the variables must be higher than minimum capacity submitted as parameter.

The high combinatorial possibilities of the clustering problems suggests the use of metaheuristic algorithms [7]. This algorithm can reach a typical optimal solution which is very close to global solution, in some cases the global optimal, in a reasonable amount of time. So, papers about clustering problems, including graph partition problem that consider additional restrictions such as connectivity and capacity had been widely reported in literature.

Some Groups [8, 9] had proposed heuristics algorithms for the capacity clustering problem, while others [1, 2, 10] had suggested algorithms for the regionalization problem, in which the connectivity restriction was considered (Automatic Zoning Procedure - AZP and the Spatial 'K'luster Analysis by Tree Edge Removal - SKATER).

The problem presented in this paper considers both connectivity and capacity restrictions into partition graph problem. It is important to underline that, excepting the AZP, the other works referenced that considered the connectivity restriction were based on Minimum Spanning Tree (MST) Partition Method. This method is composed by two steps:

1. Construction of a MST from the graph which represents the problem.
2. Formation of sets of clusters through of partitioning of MST.

According to the connectivity restriction, a natural solution for the problem will consist of building a MST T from G, respecting the smaller values of d_{ij} (1).

$$d_{ij} = \sqrt{\sum_{s=1}^{p} (x_i^s - x_j^s)^2} \quad (1)$$

In this way, these areas are geographically immediate neighbors, and homogeneity, regarding a set of p variables associated to populational and environmental known characteristics. These variables, which will be represented by x^s, $s = \{1,..,p\}$, are also called indicators (associated variables to each vertex).

Considering these indicators and using the distances d_{ij} between i and j neighbors vertexes are calculated. The distances d_{ij} represent the homogeneity degree, i.e., the proximity among values from p variables associated to all vertexes to be aggregated.

Once provided one tree T and a number k of partitions (cluster to be generated), it is possible to extract $(k - 1)$ edges from T,

defining, this way, a set of K subtrees T_j, $j=\{1,..,\ k\}$. Each one of these subtrees will be associated to one cluster.

The connectivity property can be observed in each of the subtrees (clusters). Thus, the solution for the problem will consist of partitioning T in k subtrees T_j, $j=\{1,..,\ k\}$ associated to cluster what satisfies the capacity restriction and results in the lower possible value for a fitness function(2).

$$f(T) = \sum_{j=1}^{p} \sum_{i=1}^{n} (x_{ij} - \overline{x_j})^2 \quad (2)$$

The case of AZP was based on the spatial object neighbor structure to assure the connectivity restriction and acts, basically, on the migration of the objects in order to minimize a fitness solution.

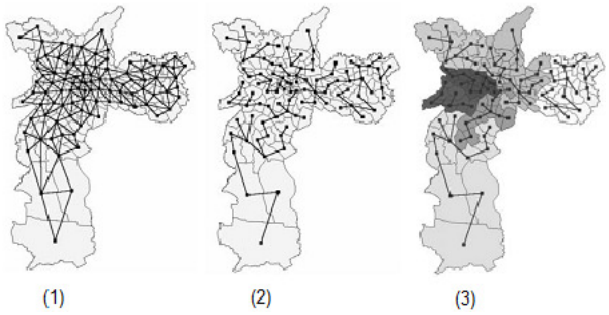


Figure 1: Adjacency relations between objects [1].

According to 1, follows the descriptions of the items: (1) connectivity graph, (2) minimum Spanning Tree and (3) an example of solution.

3. PROPOSED ALGORITHMS

Review of literature showed that the proposed algorithms work only on the edges of MST. In order to increase the possibilities of vertex migration this work presents new heuristic algorithms that act with the original submitted graph of the problem. This proposal enables and facilitates the formation of not only feasible, which the restriction of capacity is respected, but also better quality solutions.

According to [6], a good data structure for the problem is extremely important to the algorithms performance and it can be decisive for a fast convergence and quality of the obtained solutions. The *group-number* structure was used to representation of the solution, where the index of vector represents the vertex of the graph and its content represents the cluster to which the vertex belongs (also used by [5, 6, 11]).

The proposed approach consists in creating solutions using the MST Partition Method through the constructive heuristics, and so, refining its using local search procedures. It was used versions of local search that consider the original graph, and not only the MST built.

3.1. Constructive Heuristics

Two versions of constructive heuristics were proposed, assuring the connectivity restriction through MST Partition Method, both considering the concepts of GRASP Metaheuristic (*Greedy Randomized Adaptive Search Procedures* [12]).

While a first version worked aiming to build feasible solutions, which the restriction of capacity is respected, the second version acted in order to minimize the fitness solution, independently of the restriction of capacity.

Both versions act to generate k partitions, removing (k − 1) edges from T, since the hierarchical division strategy was used and, initially, all the vertexes belong to the same cluster.

The Constructive Heuristic 1 (CH1) was proposed by [11] and consists in, after the selection of the cluster (associated with a subtree T_i) that must be partitioned (what have the high fitness function), to evaluate all the possibilities of edge removal in order to minimize the fitness function. This way, must be removed the edge of high value of (3) of the subtree T_i, generation two new subtrees T_i^1 and T_i^2.

$$C_{edge} = f(T_i) - (f(T_i^1) + f(T_i^2)) \quad (3)$$

Although it is a greedy procedure which has an expensive computational cost, it was applied on the building of the initial solution for the proposed algorithm. In order to make this algorithm semi-greedy, it was used a Restricted Candidate List (RCL), which the α high edges (according C_{edge} value) are selected and, one of them is randomly selected, aiming to divide the selected cluster.

The Constructive Heuristic 2 (CH2) was based on the CH1 but, in this version, intending to obtain valid solutions. In this case, the selection of the cluster that must be partitioned occurs by capacity criteria, in which the cluster with higher capacity must be selected. Moreover, the algorithm is also semi-greedy and a RCL was used. In order to build valid solutions, the CH2 acts dividing the selected cluster C_w (subtree T_w) in the clusters C_{w1} and C_{w2} and, afterwards, one of them must have its capacity minimized and the capacity criteria respected.

3.2. Local Search Procedures

Six versions of Local Search (LS) were used considering:

- *MST*: only the edges of the MST built.
- *Original Graph*: all edges from the original submitted graph.
- *Feasible Solutions*: construction of valid solutions.
- *Better Solutions*: to minimize the fitness solution, independent of the restriction of capacity.

Table 1 ilustrates the distributions of the Local Search versions among the considering properties.

Property	LS1	LS2	LS3	LS4	LS5	LS6
MST	x			x	x	
Original Graph		x	x			x
Feasible Solutions	x	x			x	
Better Solutions			x	x		x

Table 1: Properties by Local Search versions.

Descriptions of the Local Search versions:

- *LS1*: uses the edges that were selected during the cluster partition. Basically, the procedure verifies if one and only one cluster associated to vertexes of the edge is penalized (if it has capacity less than the minimum capacity). In this case, the vertex is migrated to this cluster, aiming to regenerate the solution.
- *LS2*: realizes migrations of vertexes based on the original submitted graph of the problem, aiming to regenerate the infeasible solutions.
- *LS3*: realizes migrations of vertex based on the original submitted graph of the problem aiming to minimize the fitness' solution.

- *LS4 and LS5*: work joining adjacent clusters in which exists an edge connecting vertexes of this clusters, and after, dividing this cluster using, respectively, the CH1 and CH2 procedures.
- *LS6*: was based on the known clustering algorithm of the literature, the K-Means [13, 4] but, in this case, the restrictions of this problem were considered.

3.3. Additional Comments about the Implementation

This paper proposes Evolutionary Algorithms (EA)[12] that bring together the construtives and local search procedures. It follows the other implemented techniques:

- *Crossover*: the vertexes migration occur by the 1-point type crossover operator. It was necessary to verify if the new solutions have k clusters and if the clusters are connected.
- *Mutation*: it was used random vertex migration, aiming to perturb the solution.
- *Elitism*: The best found solutions are saved and inserted to the next population in order to improve quality by using the others procedures.
- *Minimum Capacity*: Total associated to one of the variables. This value can be either submitted as parameter or calculated at the begin of the algorithm, which β is the fit factor, k a number of clusters, n a number of vertexes and x_i^s the variable s associate with the vertex i (4).

$$Cap_{Min} = (\beta/k).\sum_{i=1}^{n} x_i^s \tag{4}$$

In the experiments, were considered only two versions of EA:

- *EAOG*: Evolutionary Algorithm that consider the original submitted graph. It was used: LS2, LS3, LS6, Elitism, CH1 or CH2.
- *EAMST*: Evolutionary Algorithm that consider only the edges of the MST. It was used: LS1, LS4, LS5, Crossover, Mutation, Elitism, CH1 or CH2.

4. COMPUTATIONAL RESULTS

A real set of twenty six instances from Brazilian Demographic Census (data for public use) was used for the experiments. Moreover, the algorithms presented were coded in Ansi C, running on a Intel Centrino II 2,4 GHz processor and 4GB RAM.

Table 2 presents properties of the used instances, where each vertex is a weigthed area. A weighted area is a small geographical area formed by a mutually exclusive enumeration areas (cluster of census segments), which comprise, each one of them, a set of records of households and people. And the associated variables are: total of houses, total of domiciles, total of person, sum of salaries, sum of time of instruction or study, sum of salary per-capita, average time of instruction or study of the responsible.

Aiming to calibrate the parameters, several preliminary experiments were run based on the selected set of instances. The obtained parameters were: k=3 (clusters), PopulationSize=10 solutions, StopCriteria=100 generations, Crossover =80%, Mutation=5% and α=5. The crossover and mutation have a high probability since its execution is evaluate in order to form only feasible solutions.

Although real applications can define the Minimum Capacity for each instance, in this experiment was fixed $\beta = 30\%$.

Id	|Vertex|	|Edge|	Id	|Vertex|	|Edge|
1	21	58	14	178	791
2	61	286	15	121	567
3	409	2020	16	75	359
4	73	350	17	114	502
5	14	46	18	133	620
6	18	59	19	195	868
7	89	363	20	68	307
8	16	60	21	181	843
9	57	236	22	151	560
10	375	1769	23	86	388
11	179	882	24	155	722
12	74	357	25	461	2385
13	231	1172	26	285	1451

Table 2: Real instances of Brazilian Demographic Census.

In the experiment, each algorithm was executed over the same instance twenty times. The elapsed time and the gap associated with the best known result of the each instance were obtained.

The tables 3 and 4 present, respectively, the best of this results by EA version for each instance and some statistics about this experiment. The EAOG obtained best results for all the instances, however, its average of elapsed time was higher then EAMST versions.

$$Gap(AEGO, EAMST) = 100 * \frac{|f_{AEGO} - f_{AEMST}|}{f_{AEGO}} \tag{5}$$

Id	Gap	Id	Gap	Id	Gap
1	26.97	10	43.33	19	54.08
2	7.1	11	61.85	20	16.44
3	5.82	12	40.31	21	41.86
4	20.3	13	51.23	22	39.05
5	11.71	14	91.09	23	60.6
6	3.97	15	65.76	24	48.96
7	78.84	16	35.38	25	26.46
8	17.44	17	56.49	26	56.25
9	59.59	18	84.02		

Table 3: Gap between EAOG and EAMST.

Average Time	EAOG	269 seconds
	EAMST	133 seconds
Gap (EAOG, EAMST)	Min	3.97%
	Max	91.09%
	Mean	42.49%
	Median	42.59%
Gap [Best Known reference]	EAOG	4.00%
	EAMST	51.00%

Table 4: Statistics.

In order to analyze the results, three categories were created according to the Gap values of the best solution known: Best (Gap = 0%), Interesting (Gap $\leq$ 5%) and Bad ($Gap > 70\%$).

The table 5 presents the results by categories.

Since the AEOG reached best results but its elapsed time was higher than of AEMST, both algorithms were submitted to a new experiment. They were run one hundred times, over three among the bigger selected instances and, in this experiment, the StopCriteria was a maximum time (300 seconds) or the solution reach the target value, submitted as parameters.

Categories	EAOG	EAMST
Best	40%	12%
Interesting	60%	17%
Bad	0%	29%

Table 5: Results by categories.

In this experiment all the AEOG executions reached the target, while the AEMST had probabilities of 52%, 55% and 38% for the instances 4, 13 and 22, respectively.

Despite the algorithm had obeyed the stipulated processing time, the EAMST continued limited in best local solutions, while the EAOG obtained new different solutions, that could not be formed through the only MST method. Moreover, the AEOG reached the target of the instances 4, 13 and 22 at 40, 10 and 10 seconds, respectively.

5. CONCLUSIONS

In this paper two versions of constructive heuristics were proposed, both considering the concepts of GRASP Metaheuristic. Afterwards, six local search procedures were used aiming to refine the solutions, in order to increase de solutions' quality or regenerate infeasible solutions.

Two Evolutionary Algorithms were presented, bring together the construtives and local search procedures: The EAOG (based on the Original Graphs) and EAMST (based only on edges of MST).

It was possible to confirm that the procedures that acted with the original submitted graph increase the possibilities of vertex migration and thus facilitated the formation of both valid as better quality solutions.

The computational results showed that the use of Constructive Heuristics that consider only edges of MST together a local search procedures and the use of Original Graphs are an interesting alternative to solve this problem, improving both the solution's quality as the quantity of formation of valid solutions.

These results indicate that the proposed heuristics are an efficient way to solve this problem. Besides, as another ways to solve it we can cite: the use of Pathrelinking in order to integrate intensification and diversification in search for new best solutions [12]; to develope and analyze the use of other metaheuristics, such as: Iterated Local Search (ILS), Variable Neighborhood Search (VNS), Tabu Search or a hybrid heuristic version [12].

6. ACKNOWLEDGMENTS

To all the teachers and students of the Computer Institute at UFF (`http://www.ic.uff.br`) and CAPES (`http://www.capes.gov.br`) for the financial support.

7. REFERENCES

[1] R. M. Assunção, M. C. Neves , G. Câmara, C. Freitas, "Efficient regionalization techniques for socio-economic geographical units using minimum spanning trees," *International Journal of Geographical Information Science*, vol. 20, no. 7, pp. 797–811, 2006.

[2] M.J. Smith, M. F. Goodchild, P. A. Longley, *Geospatial Analysis : a Comprehensive Guide to Principles, Techniques and Software Tools*. Troubadour Publishing Limited, 2009.

[3] J. Han and M. Kamber, *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 2006.

[4] H. C. Romesburg, *Cluster Analysis for Researchers*. Lulu Press, 2004.

[5] C. R. Dias, L. S. Ochi, "Efficient evolutionary algorithms for the clustering problems in directed graphs," in *Proc. of the IEEE Congress on Evolutionary Computation (IEEE-CEC)*, Canberra, Austrália, 2003, pp. 983–988.

[6] D. Doval, S. Mancoridis, B. S. Mitchell, "Automatic clustering of software systems using a genetic algorithm," in *Proc. of the Int. Conf. on Software Tools and Engineering Practice*, Pittsburgh, USA, 1999, pp. 73–81.

[7] P. Hansen, B. Jaumard, "Cluster analysis and mathematical programming," *Mathematical Programming*, vol. 79, pp. 191–215, 1997.

[8] S. W. Scheuerer, "A scatter search heuristic for the capacitated clustering problem," *European Journal of Operational Research*, vol. 169, 2006.

[9] H. M. Shieh, M. D. May, "Solving the capacitated clustering problem with genetic algorithms," *Journal of the Chinese Institute of Industrial Engineers*, vol. 18, 2001.

[10] R. M. Assuncao, J. P. Lage, A. E. Reis, "Analise de conglomerados espaciais via arvore geradora minima," *Revista Brasileira de Estatística*, 2002.

[11] G. S. Semaan, L. S. Ochi, J. A. M. Brito, "An efficient evolutionary algorithm for the aggregated weighting areas problem," in *International Conference on Engineering Optimization*, 2008.

[12] F. Glover, *Handbook of Metaheuristics*. Kluwer Academic Publishers, 2003.

[13] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability*, 1967.

Lagrangean based algorithms for the Weight-Constrained Minimum Spanning Tree Problem

Cristina Requejo * Eulália Santos * †

* Department of Mathematics, University of Aveiro
3810-193 Aveiro, Portugal
crequejo@ua.pt

† School of Technology and Management, Polytechnic Institute of Leiria
2411-901 Leiria, Portugal
eulalia.santos@ipleiria.pt

ABSTRACT

The Weight-Constrained Minimum Spanning Tree problem (WMST) is a NP-hard combinatorial optimization problem having important applications in the telecommunication networks design and communication networks. We use simple but effective Lagrangean based algorithms to compute lower and upper bounds. Computational results show that the algorithms are fast and present small gap values.

Keywords: Weight-constraints, Constrained minimum spanning tree, Lagrangean relaxation, Heuristics

1. INTRODUCTION

In this work we discuss Lagrangean based algorithms for the Weight-Constrained Minimum Spanning Tree problem (WMST).

Consider an undirected complete graph $G = (V,E)$, with node set $V = \{0,1,\ldots,n-1\}$ and edge set $E = \{\{i,j\},\ i,j \in V, i \neq j\}$. Associated with each edge $e = \{i,j\} \in E$ consider nonnegative integer costs c_e and nonnegative integer weights w_e. The Weight Minimum Spanning Tree problem (WMST) is to find a spanning tree $T = (V_T, E_T)$ in G ($V_T \subseteq V$ and $E_T \subseteq E$) of minimum cost $C(T) = \sum_{e \in E_T} c_e$ and with total weight $W(T) = \sum_{e \in E_T} w_e$ not exceeding a given limit W. This combinatorial optimization problem is NP-hard [1, 2].

The WMST is known under several different names. It was first mentioned in Aggarwal, Aneja and Nair [1], under another name, the *MST problem subject to a side constraint*. In this paper the authors propose an exact algorithm to solve the problem that uses a Lagrangian relaxation to approximate a solution combined with a branch and bound strategy. This kind of solution approach can also be found in the work of Shogan [3]. The paper of Ravi and Goemans [4] describes an approximate scheme. In [5] Xue presents a simple but efficient primal-dual algorithm to find approximate solutions. Another approach to solve the problem is given in Hong, Chung and Park [6] where the authors propose a fully polynomial bicriteria approximation scheme. Hassin and Levin [7] adopt the ideas in [4] and add to them an application of a matroid intersection algorithm. Yamada, Watanabe and Kataoka [2] consider a weight-constrained maximum spanning tree problem. They prove the problem is NP-hard, use a local search heuristic to obtain upper bounds, a Lagrangian relaxation to obtain lower bounds, use a branch-and-bound algorithm to solve the problem and propose a method to accelerate the computation. The authors refer that the results can be easily applied to the minimization case. Henn [8] presents a compilation of results and existing algorithms to solve the problem.

A related approach is to include the weight of the tree as a second objective instead of a hard constraint. The resulting problem is the bi-objective spanning tree problem ([9, 10, 11, 12, 13, 14, 15], among many others).

The WMST appears in several real applications and the weight restrictions are mainly concerned with a limited budget on installation/upgrading costs. A general application is related with the upgrade and design of physical systems, somehow connected through a minimum spanning tree, when there is a budget restriction. One such application arises in the areas of communication networks and network design, in which information is broadcast over a minimum spanning tree. There are several problems that consider the design of the enhancement of the performance of an underlying network by carrying out upgrades at certain nodes and/or edges of the network. Upgrading a node corresponds to installing faster switching equipment at that node. Such upgrade reduces the communication delay along each edge emanating from the node. Similarly, upgrading an edge corresponds to replacing an existing link with a new type of link. Moreover, costs/profits is not the only meaning for the weights. Edge weights may represent the delay of an edge or the logarithm of the reciprocal of the reliability of an edge [5]. Another example (see [8, 16]) arising in communication networks problems, is the minimum cost reliability constrained spanning tree. In this application we are given a set of nodes in the plane that can communicate with each other. The objective is to connect the nodes. The cost of a connection might be modeled by the distance of the nodes and the reliability of a connection by its fault probability. We now want to compute a minimum cost connection (spanning tree) such that its total fault probability is beyond a given limit. The interest from the telecommunications community arises from the great deal of emphasis on the need to design communication protocols that deliver certain performance guarantees. This need is the result of an explosive growth in high bandwidth real time applications that require demanding QoS (Quality of Service) guarantees. It is for this reason that the WMST has assumed great importance in telecommunications network applications.

There are several studies of Lagrangean based approximation algorithms either to general constrained combinatorial optimization problems, cf. [17], or to weight/resource constrained shortest path problems, cf. [18, 19]. The WMST has received only brief references and computational results are almost non existing. We will describe Lagrangean based algorithms to the WMST and obtain computational results. To present the Lagrangean relaxation to the WMST in Section 4, we describe a general formulation to the problem in Section 2. We discuss some properties of the prob-

lem in Section 3 and a solution procedure in Section 5. We present existing settings and propose a different setting to obtain approximate trees in the solution procedure. Computational results to assess the quality of the discussed procedures will be shown in Section 6.

2. A FORMULATION FOR THE WMST

Several formulations are well known for the MST (see Magnanti and Wolsey [20]). In [21] natural and extended formulations for the WMST are discussed. To obtain formulations to the WMST one can easily adapt a MST formulation.

It is well known (see Magnanti and Wolsey [20]) that oriented formulations (based on the underlying directed graph) leads, in general, to tighter formulations (formulations whose lower bounds provided by the linear relaxations are closer to the optimum values). Thus, henceforward we consider the corresponding directed graph, with root node 0, where each edge $e = \{0, j\} \in E$ is replaced with arc $(0, j)$ and each edge $e = \{i, j\} \in E, i \neq 0$, is replaced with two arcs, arc (i, j) and arc (j,i), yielding arc set $A = \{(i, j),\ i \in V \setminus \{0\}, j \in V, i \neq j\}$. These arcs inherit the cost and weight of the ancestor edge.

Henceforward, let P_L be the linear programming relaxation of formulation P and let $\vartheta(P)$ be the optimal value of P.

Consider the original variables, the binary variables x_{ij} (for all $(i, j) \in A$) indicating whether arc (i, j) is in the MST solution [20]. Two classical formulations on the space of the original variables for the MST can be considered. In order to ensure the connectivity of the feasible solutions and to prevent the existence of circuits in the feasible solutions, one formulation uses the cut-set inequalities and the other formulation uses circuit elimination inequalities. The linear relaxation of both models provide the same bound [20]. However the number of inequalities in both sets increase exponentially with the size of the model. It is well known that in order to ensure connectivity/prevent circuits, instead of using one of those families with an exponential number of inequalities, one can use compact extended formulations. The well-known Multicommodity Flow formulation (MF) using the additional flow variables can be considered. In this formulation the connectivity of the solution is ensured through the flow conservation constraints together with the connecting constraints [20]. These three formulations for the MST are easily adapted for the WMST through the inclusion of a weight constraint. Therefore a formulation to the WMST is as follows.

$$
\begin{aligned}
(WMST) \qquad & min \quad \sum_{(i,j)\in A} c_{ij}x_{ij} \\
& s.t. \quad x \in (MST) && (1)\\
& \qquad \sum_{(i,j)\in A} w_{ij}x_{ij} \leq W. && (2)
\end{aligned}
$$

Where $x = (x_{ij}) \in \mathbb{R}^{|A|}$ and (MST) represents a set of inequalities describing the convex hull of the (integer) solutions of the MST and can use one of the sets of inequalities referred previously (the circuit elimination inequalities, the cut-set inequalities, the flow conservation constraints together with the connecting constraints) plus the following constraints

$$
\begin{aligned}
& \sum_{i\in V} x_{ij} = 1 \qquad j \in V && (3)\\
& x_{ij} \in \{0,1\} \qquad (i,j) \in A. && (4)
\end{aligned}
$$

Constraint (2) is the weight constraint and we emphasize that the above formulation without the weight constraint is a formulation for the MST [20].

If the incidence vector $x = (x_{ij}) \in \mathbb{R}^{|A|}$ represents an (integer) MST solution, and subgraph $T = (V, A_T)$, $A_T \subseteq A$, of $G = (V, A)$ the corresponding tree, then $C(T) = \sum_{(i,j)\in A} c_{ij}x_{ij} = \sum_{(i,j)\in A_T} c_{ij}$ and $W(T) = \sum_{(i,j)\in A} w_{ij}x_{ij} = \sum_{(i,j)\in A_T} w_{ij}$. Furthermore, if we define a matrix of non-negative profits p_{ij} associated to each arc $(i, j) \in A$, then we use $P(T) = \sum_{(i,j)\in A} p_{ij}x_{ij} = \sum_{(i,j)\in A_T} p_{ij}$.

3. SOME PROPERTIES OF THE WMST

The well know Minimum Spanning Tree problem (MST) is to find a spanning tree $T_c = (V, A_{T_c})$, $A_{T_c} \subseteq A$, on $G = (V, A)$ of minimum cost $C(T_c) = \sum_{(i,j)\in A_{T_c}} c_{ij}$ and for this combinatorial optimization problem there are several polynomial algorithms such as Sollin's, Kruskal's and Prim's algorithm (see [22] for descriptions of these algorithms). An additional constraint to the MST such as the one we use (the total tree weight $W(T_c) = \sum_{(i,j)\in A_{T_c}} w_{ij}$ must not exceed a given limit W) turns the MST into a NP-hard problem [1]. Consider a companion problem to the WMST, the Minimum-weight Spanning Tree problem that is to find a spanning tree $T_w = (V, A_{T_w})$, $A_{T_w} \subseteq A$, on $G = (V, A)$ of minimum weight $W(T_w) = \sum_{(i,j)\in A_{T_w}} w_{ij}$.

T_c and T_w are two spanning trees of G, T_c of minimum cost and T_w of minimum weight. Moreover, these trees give us upper and lower bounds on the optimal value of the problem

$$C(T_c) \leq \vartheta(WMST) \leq C(T_w)$$

and we can assume the following proposition.

Proposition 1. *There exists an optimal solution for the WMST if and only if*

$$W(T_w) \leq W \leq W(T_c).$$

Clearly, if $W(T_w) > W$, then the WMST has no solution. Furthermore, we have the following.

Proposition 2. *If $W(T_c) \leq W$, then T_c is an optimal solution for the WMST.*

Consider another companion problem to the WMST. Define some non-negative profits p_{ij} associated to each arc $(i, j) \in A$ which are linear combination of the cost and weight associated to each arc, $p_{ij} = aw_{ij} + bc_{ij}$ with real scalars a, b. The Minimum-profit Spanning Tree problem that is to find a spanning tree $T_p = (V, A_{T_p})$, $A_{T_p} \subseteq A$, on G of minimum profit $P(T_p) = \sum_{(i,j)\in A_{T_p}} p_{ij}$. If $a = 0$ and $b = 1$ then we have $T_p \equiv T_c$. If $a = 1$ and $b = 0$ then we have $T_p \equiv T_w$.

4. LAGRANGEAN RELAXATION

In order to derive a Lagrangean relaxation attach the Lagrangean multiplier λ to the weight constraint (2) and dualize the constraint in the usual Lagrangean way. This leads to the following relaxed problem.

$$
\begin{aligned}
(WMST_\lambda) \quad -\lambda W \quad + \quad & min \quad \sum_{(i,j)\in A} (c_{ij} + \lambda w_{ij})x_{ij} \\
& s.t. \quad x = \in (MST)
\end{aligned}
$$

For every non-negative multiplier λ, the tree solutions to this relaxed problem give us lower bounds on the optimum value, i.e.

$$\vartheta(WMST_\lambda) \leq \vartheta(WMST).$$

For a given non-negative value of the Lagrangean multiplier λ, the relaxed problem $WMST_\lambda$ can be solved using any well known polynomial algorithm to solve the MST [22]. Moreover, if for each multiplier λ we define the profits $p^\lambda_{ij} = c_{ij} + \lambda w_{ij}$, then

$$\vartheta(WMST_\lambda) = -\lambda W + P(T_{p^\lambda}).$$

Classically a Lagrangean relaxation is solved using a subgradient optimization procedure [23]. The subgradient optimization procedure starts by initializing the Lagrangean multipliers. After, iteratively, solves the relaxed problem $WMST_{\lambda_k}$, then actualizes the Lagrangean multiplier λ_k by setting, at each iteration k , $\lambda_{k+1} = \max\{0, \lambda_k + s_k d_k\}$ using a direction d_k and a step-size s_k, and finally verifies some stopping criteria.

An appropriate choice for the step size s_k produces a convergent method. We can use [23]

$$s_k = \rho \frac{C(T_w) - \vartheta(WMST_{\lambda_k})}{(\sum_{(i,j)\in A} w_{ij} x^k_{ij} - W) d_k} = \rho \frac{C(T_w) - P(T_{p^{\lambda_k}}) + \lambda_k W}{(W(T_{p^{\lambda_k}}) - W) d_k}$$

with $0 < \rho < 2$ and using the upper bound $C(T_w)$ to approximate the optimum value of the problem. Observe that for the tree solution $x^k = (x^k_{ij})$ of the Lagrangean relaxed problem $WMST_{\lambda_k}$, corresponding to $T_{p^{\lambda_k}}$, we have $\vartheta(WMST_{\lambda_k}) = -\lambda_k W + P(T_{p^{\lambda_k}})$ and $W(T_{p^{\lambda_k}}) = \sum_{(i,j)\in A} w_{ij} x^k_{ij}$.

5. SOLUTION PROCEDURE

In order to obtain an approximate solution to the WMST we propose the following general algorithm.

Algorithm

Step 1 *Obtain an upper bound.*
Find a spanning tree $T_w = (V, A_{T_w})$, $A_{T_w} \subseteq A$, on G of minimum weight $W(T_w) = \sum_{(i,j)\in A_{T_w}} w_{ij}$.
If $W(T_w) > W$, then there is no solution. STOP. Otherwise, set $T_\alpha = T_w$.

Step 2 *Obtain a lower bound.*
Find a spanning tree $T_c = (V, A_{T_c})$, $A_{T_c} \subseteq A$, on G of minimum cost $C(T_c) = \sum_{(i,j)\in A_{T_c}} c_{ij}$.
If $W(T_c) \leq W$, then T_c is an optimal solution. STOP. Otherwise, set $T_\beta = T_c$.

Step 3 *Compute an approximate tree.*
Compute profits p_{ij} for every $(i, j) \in A$.
Find a spanning tree $T_p = (V, A_{T_p})$, $A_{T_p} \subseteq A$, on G of minimum value $P(T_p) = \sum_{(i,j)\in A_{T_p}} p_{ij}$.
Compute $P(T_p)$, $W(T_p)$ and $C(T_p)$.

Step 4 *Stopping criteria.*
If $W(T_p) \leq W$ then update upper bound, i.e. if $C(T_p) < C(T_\alpha)$ replace T_α by T_p;
otherwise update lower bound, i.e. if $C(T_p) > C(T_\beta)$ replace T_β by T_p.
If $|P(T_\alpha) - P(T_p)| \leq tol$, then
T_α is the approximate solution, STOP.
Go To **Step 3**.

The subgradient optimization scheme perfectly fits this algorithm layout. Now we will discuss settings for the non-negative profits $p_{ij} = a w_{ij} + b c_{ij}$, with real scalars a, b, associated to each arc $(i, j) \in A$ and their update at each iteration. We will consider settings for the profits p_{ij} characterized by associating a parameter, the Lagrangean multiplier, to the weights, $a = \lambda_k$, and a parameter with value equal to one to the costs, $b = 1$. Two examples of such settings will be given next.

Jüttner et al. [19] built up the Lagrangian Relaxation Based Aggregated Cost (LARAC) algorithm which solves the Lagrangian relaxation of the constrained shortest path (CSP) problem. In [24] the equivalence of the LARAC algorithm and other algorithms in [17, 18, 19] is shown. Using the ideas of these algorithms, the first setting is $a = \lambda_k = \dfrac{C(T_\alpha) - C(T_\beta)}{W(T_\beta) - W(T_\alpha)}$.

If the Held, Wolfe and Crowder [25] direction is to be considered $d_k = \sum_{(i,j)\in A} w_{ij} x^k_{ij} - W = W(T_{p^{\lambda_k}}) - W$, leading to the second setting

$$a = \lambda_k = \max\{0, \lambda_{k-1} + \rho \frac{C(T_w) - P(T_{p^{\lambda_{k-1}}}) + \lambda_{k-1} W}{W(T_{p^{\lambda_{k-1}}}) - W}\}$$

and initializing $\lambda_0 = \frac{C(T_w) - C(T_c)}{W(T_c) - W}$.

6. COMPUTATIONAL RESULTS

Computational results will assess the quality of the approximate solutions obtained with each setting of the profits.

At the moment we present some computational results of the approximation algorithms for instances to the weight-constrained minimum spanning tree problem on complete graphs and between 150 and 300 nodes. Costs and weights are generated based on Euclidean distances combined with Pisinger's [26] instances and $W = \frac{W(T_c) + W(T_w)}{2}$.

$\|V\|$	$W(T_w)$	W	$W(T_c)$	$C(T_c)$	$C(T_w)$	$C(T_p)$
150	824	4197	7570	781	7529	1114
200	866	5890	10914	890	10557	1154
250	958	6921	12884	1004	12925	1361
300	1080	8281	15481	1082	14588	1470

Table 1: Computational results.

Preliminary computational results show that the algorithms are fast and present small gap values. For the instances in Table 1 the bound obtained is equal for both profits settings and its value is shown in the last column.

An extensive computational experience is performed to complete this section.

7. ACKNOWLEDGEMENTS

The research of the authors was supported by Center for Research and Development in Mathematics and Applications (CIDMA) both from the Portuguese Foundation for Science and Technology (FCT), cofinanced by the European Community Fund FEDER/POCI 2010.

8. REFERENCES

[1] V. Aggarwal, Y. P. Aneja, and K. P. K. Nair, "Minimal spanning tree subject to a side constraint," *Computers and Operations Research*, vol. 9, pp. 287–296, 1982.

[2] T. Yamada, K. Watanabe, and S. Kataoka, "Algorithms to solve the knapsack constrained maximum spanning tree problem," *International Journal of Computer Mathematics*, vol. 82, pp. 23–34, 2005.

[3] A. Shogan, "Constructing a minimal-cost spanning tree subject to resource constraints and flow requirements," *Networks*, vol. 13, pp. 169–190, 1983.

[4] R. Ravi and M. Goemans, "The constrained minimum spanning tree problem," in *Proceedings of the Scandinavian Workshop on Algorithmic Theory*, ser. Lecture Notes in Computer Science, vol. 1097, 1996, pp. 66–75.

[5] G. Xue, "Primal-dual algorithms for computing weight-constrained shortest paths and weight-constrained minimum spanning trees," in *Performance, Computing, and Communications Conference, 2000. IPCCC '00. Conference Proceeding of the IEEE International*, 2000, pp. 271 –277.

[6] S.-P. Hong, S.-J. Chung, and B. H. Park, "A fully polynomial bicriteria approximation scheme for the constrained spanning tree problem," *Operations Research Letters*, vol. 32, pp. 233–239, 2004.

[7] R. Hassin and A. Levin, "An efficient polynomial time approximation scheme for the constrained minimum spanning tree problem using matroid intersection," *SIAM Journal on Computing*, vol. 33, no. 2, pp. 261–268, 2004.

[8] S. Henn, "Weight-constrained minimum spanning tree problem," Master's thesis, Department of Mathematics, University of Kaiserslautern, Kaiserslautern, Germany, 2007.

[9] K. A. Andersen, K. Jörnsten, and M. Lind, "On bicriterion minimal spanning trees: an approximation," *Computers and Operations Research*, vol. 23, pp. 1171–1182, 1996.

[10] G. Zhou and M. Gen, "Genetic algorithm approach on multi-criteria minimum spanning tree problem," *European Journal of Operational Researc*, vol. 114, pp. 141–152, 1999.

[11] G. Chen, S. Chen, W. Guo, and H. Chen, "The multi-criteria minimum spanning tree problem based genetic algorithm," *Information Sciences*, vol. 177, pp. 5050–5063, 2007.

[12] F. Sourd and O. Spanjaard, "A multiobjective branch-and-bound: application to the bi-objective spanning tree problem," *INFORMS Journal on Computing*, vol. 20, pp. 472–484, 2008.

[13] D. Rocha, E. Goldbarg, and M. Goldbarg, "A new evolutionary algorithm for the biobjective minimum spanning tree problem," in *Proceedings of the ISDA 07, International Conference on Intelligent Systems Design and Applications*, 2007, pp. 735 –740.

[14] M. Davis-Moradkhan, W. Browne, and P. Grindrod, "Extending evolutionary algorithms to discover tri-criterion and non-supported solutions for the minimum spanning tree problem," in *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, ser. GECCO '09, 2009, pp. 1829–1830.

[15] S. Monteiro, E. Goldbarg, and M. Goldbarg, "A new transgenetic approach for the biobjective spanning tree problem," in *2010 IEEE Congress on Evolutionary Computation (CEC)*, 2010, pp. 1 –5.

[16] K. Mehlhorn and M. Ziegelmann, "CNOP - a package for constrained network optimization," in *Algorithm Engineering and Experimentation*, ser. Lecture Notes in Computer Science, 2001, vol. 2153, pp. 17–31.

[17] D. Blokh and G. Gutin, "An approximation algorithm for combinatorial optimization problems with two parameters," *Australasian Journal of Combinatorics*, vol. 14, pp. 157–164, 1996.

[18] G. Handler and I. Zang, "A dual algorithm for the constrained shortest path problem," *Networks*, vol. 10, pp. 293–310, 1980.

[19] A. Jüttner, B. Szviatovszki, I. Mécs, and Z. Rajkó, "Lagrange relaxation based method for the QoS routing problem," in *Proceedings. IEEE INFOCOM*, 2001, pp. 859–868.

[20] T. Magnanti and L. Wolsey, "Optimal trees," in *Network Models*, ser. Handbooks in Operations Research and Management Science, Vol. 7, M. Ball, T. Magnanti, C. Monma, and G. Nemhauser, Eds. North-Holland: Elsevier Science Publishers, 1995, pp. 503–615.

[21] C. Requejo, A. Agra, A. Cerveira, and E. Santos, "Formulations for the weight-constrained minimum spanning tree problem," in *Proceedings of the International Conference on Numerical Analysis and Applied Mathematics*, ser. AIP Conference Proceedings, vol. 1281, 2010, pp. 2166–2169.

[22] R. Ahuja, T. Magnanti, and J. Orlin, *Network Flows: Theory, Algorithms and Applications*. Prentice-Hall, 1993.

[23] N. Shor, *Minimization Methods for Non-Differentiable Functions*. Springer-Verlag, 1985, english translation.

[24] Y. Xiao, K. Thulasiraman, G. Xue, and A. Jüttner, "The constrained shortest path problem: Algorithmic approaches and an algebraic study with generalization," *AKCE International Journal of Graphs and Combinatorics*, no. 2, pp. 63–86, 2005.

[25] M. Held, P. Wolfe, and H. Crowder, "Validation of subgradient optimization," *Mathematical Programming*, vol. 6, pp. 62–88, 1974.

[26] D. Pisinger, "Where are the hard knapsack problems?" DIKU, University of Copenhagen, Denmark, Technical Report 2003/08, 2003.

A Heuristic and an Exact Method for Pattern Sequencing Problems

Luigi De Giovanni [*] Gionata Massi [†] Ferdinando Pezzella [†] Marc E. Pfetsch [‡]
Giovanni Rinaldi [§] Paolo Ventura [§]

[*] Dipartimento di Matematica Pura e Applicata, Università degli Studi di Padova
via Trieste 63, 35121 Padova (Italy)
luigi@math.unipd.it

[†] Dipartimento di Ingegneria Informatica Gestionale e dell'Automazione
Università Politecnica delle Marche – via Brecce Bianche 12, Ancona (Italy)
{massi,pezzella}@diiga.univpm.it

[‡] Institute for Mathematical Optimization, Technische Universität Braunschweig
Pockelsstraße 14, 38106 Braunschweig (Germany)
m.pfetsch@tu-bs.de

[§] Istituto di Analisi dei Sistemi e Informatica - Antonio Ruberti, CNR
viale Manzoni 30, 00185 Roma (Italy)
{rinaldi,ventura}@iasi.cnr.it

ABSTRACT

In many applications, a suitable permutation of patterns (electronic circuit nodes, cutting patterns, product orders etc.) has to be found in order to optimize over some given objective function, so giving rise to the so-called Open Stack Problems. We focus on the Gate Matrix Layout Problem, where electronic circuits are obtained by connecting gates and one seeks a gate layout permutation that minimizes connection costs under restrictions on the circuit area. In the literature, the connection costs and the circuit area are also know as Time of Open Stacks and Maximum Number of Open Stacks, respectively. We propose a genetic algorithm providing heuristic solutions, and a branch-and-cut algorithm, based on a new linear integer programming formulation and representing, at our best knowledge, the first exact approach in the literature. The algorithms are under extensive test, and preliminary results on real instances are presented here.

Keywords: Time of Open Stacks, Maximum Number of Open Stacks, Genetic Algorithms, Integer Linear Programming, Branch-and-Cut

1. INTRODUCTION

The Gate Matrix Layout Problem is related to programmable logic array folding in Very Large Scale Integration (VLSI) electronic circuit design [1]. Roughly speaking, *gates* correspond to circuit nodes and different connections are required. Each connection involves a subset of nodes and is called *net*. Figure 1(a) shows an example where 7 gates (vertical lines) have to be connected according to 5 different nets, described by dots of the same row: net *A* connects gates 1, 3 and 5, net *B* connects gates 1, 4, 5 and 6 etc. *Wires* are used to create connections, one for each net, as shown in Figure 1(b). Note that, to connect the gates of a net, it may be necessary to cross other gates not included in the net, depending on the gate layout sequence. Also, a single *connection track* can be used to place non-overlapping net wires, as shown in Figure 1(c) for nets *D* and *E*. The total wire length determines the connection cost, while the number of tracks determines the total circuit area, which may be limited by design constraints or efficiency issues.

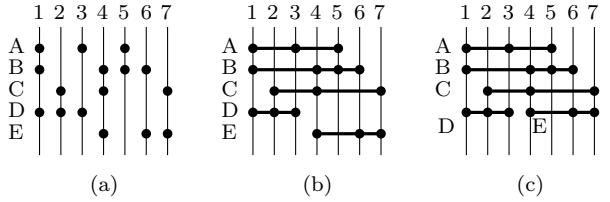


Figure 1: *Sample gate matrix: connection requests (a), wired nets (b) and connection tracks (c).*

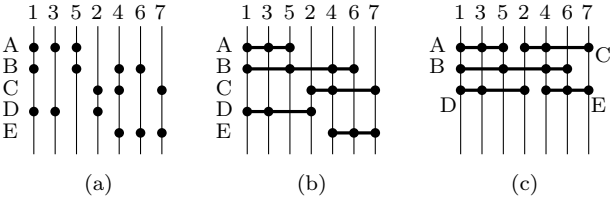


Figure 2: *Sample gate matrix: an improved gate sequence.*

Both indicators give an estimate of the circuit layout efficiency and depend on how gates are sequenced. The gate layout of Figure 1 requires 19 wire units and 4 tracks, corresponding to the maximum number of overlapping net wires. A better layout is shown in Figure 2, using 15 wire units and 3 tracks.

We define the Gate Matrix Layout Problem (GMLP) as the problem of finding a gate permutation such that the connection cost is minimized and the number of required tracks is limited. The problem is NP-Hard and has several applications in different fields [2]. For example, in production planning, gates correspond to articles, nets to client orders and wires represent the occupation of dedicated order stacks (and related loading facilities) over all the order processing time, depending on the article production sequence. The same stack can be used for non-overlapping orders and one wants to find a production sequence that minimizes the total stack occupation time, under the restriction that the maximum number of overlapping orders, that is the maximum number of simultaneously open stacks during the production process, is at most the

number of available stacks, as determined by plant layouts. Similarly, in cutting stock environments, the items (corresponding to nets in GMLP) obtained from panels sawed according to given cutting patterns (corresponding to gates) are heaped on stacks around the sawing machine. Stacks remain open during all the production time of the related item and, again, the same stack (corresponding to track) can be used for items whose production does not overlap over time. The problem is to find a cutting pattern permutation that minimizes the total stack opening time, provided that the maximum number of simultaneously open stacks during the cutting process must not exceed a given threshold, which is a parameter of the sawing center. In the literature, the total stack occupation time and the maximum number of simultaneously open stacks are known as Time of Open Stacks (TOS) and Maximum number of Open Stacks (MOS), respectively. In GMLP, the wire length corresponds to TOS, and the number of required tracks corresponds to MOS. Note that a given gate sequence may not be feasible because the number of *required* tracks (MOS) exceeds the number of *available* tracks as determined by the restrictions on the circuit area.

We can characterize an instance of GMLP by a *production matrix* $M \in \{0,1\}^{m \times n}$ and a parameter $\lambda \in \mathbb{Z}_+$ representing the number of *available* tracks and, hence, an upper bound for MOS, meaning that all the sequences having MOS greater than λ are not feasible. Rows of M are associated with nets, columns with gates, and $M(i,j) = 1$ if and only if net i includes gate j. A solution of GMLP consists in a sequence $\phi : [1,\ldots,n] \to [1,...,n]$, where $\phi(j)$ indicates the layout position of gate j. Such a solution defines a new matrix M_ϕ obtained from M by permuting its columns according to ϕ. From M_ϕ we obtain a *stack matrix* $\bar{M}_\phi$ by switching to 1 any 0 of M_ϕ between two 1s in the same row. Therefore $\bar{M}_\phi(i,j) = 1$ if and only if, according to ϕ, the wire of net i includes or crosses gate j. Figure 3 reports the production matrix of the

$M =$

	1	2	3	4	5	6	7
1	1	0	1	0	1	0	0
2	1	0	0	1	1	1	0
3	0	1	0	1	0	0	1
4	1	1	1	0	0	0	0
5	0	0	0	1	0	1	1

(a)

$M_\phi =$

	1	3	5	2	4	6	7
1	1	1	1	0	0	0	0
2	1	*1*	1	*1*	1	1	0
3	0	0	0	1	1	*1*	1
4	1	1	*1*	1	0	0	0
5	0	0	0	0	1	1	1

(b)

Figure 3: *Sample Production Matrix M (a), and Stack Matrix M_ϕ ($\phi = [1,3,5,2,4,6,7]$) with switched elements in italics (b).*

sample gate matrix of Figure 2 and the stack matrix of sequence $[1,3,5,2,4,6,7]$. Note that MOS and TOS for a given sequence ϕ can be easily obtained from $\bar{M}_\phi$. The length of the wire required by net i is the distance (in number of gates) between the first and the last gate of i, equal to the number of 1s in the i-th row of $\bar{M}_\phi$, minus 1 (the first gate must not be considered). Therefore, the length of the wire for a single net is the sum of the entries of the related row of $\bar{M}_\phi$ minus 1 and TOS is the sum of all the entries of $\bar{M}_\phi$, minus m. MOS is the maximum number of 1s appearing in any of the columns of $\bar{M}_\phi$. Summarizing, given a $\{0,1\}$-matrix M, GMLP is to find a column permutation having MOS not greater than λ and minimizing *TOS*.

Literature on pattern sequencing problems is rich and related to different application fields and solution techniques. Nevertheless, most works consider MOS minimization ([3, 4, 5, 6], among others), and TOS is sometimes used to heuristically drive the search of good MOS sequences (see for example [7, 8]). Just a few works take TOS optimization explicitly into account. Among the most recent ones, we cite [9], proposing a Constructive Genetic Algorithm, where GMLP is solved by integrating genetic operators, local search and schemata filling heuristics, and [10], where a bi-objective approach is considered for an application in the paper industry, and the set of Pareto-optimal solutions is approximated by a genetic algorithm improved by initial heuristics and local search.

In this paper, we focus on GMLP, i.e. on pattern sequencing problems where TOS has to be minimized under restrictions on MOS, and we propose two algorithms: the first one, described in Section 2, aims at determining both an as low as possible threshold λ for the number of tracks (MOS), and a feasible sequence with a low connection cost (TOS); the second one starts from this sequence and minimizes the wire length (TOS), provided that MOS must not exceed λ (Section 3). The first algorithm is based on a genetic approach with a composite and dynamic definition of the fitness function. The second algorithm exploits the flexibility of a new integer programming formulation based on the properties of consecutive-ones matrices and solved by branch-and-cut. An extensive computational campaign is in progress, and preliminary results on real GMLP instances are presented in Section 4.

2. GENETIC ALGORITHM

The aim of the first algorithm for GMLP is twofold. First, we need to determine an appropriate threshold λ for MOS, which may be not a priori known. For example, in production or cutting stock environments, the limitation on the number of available stacks may be too restrictive, so that no feasible sequence exists and temporary warehousing is necessary. We thus want to take λ as low as possible, to limit temporary warehousing and preserve process efficiency. Second, we seek for a feasible sequence that, beyond minimizing MOS, has also a good TOS, to mimimize connection costs. Note that this may also speed-up the branch-and-cut algorithm for TOS optimization, as a good initial incumbent solution is available. We consider a genetic approach: genetic algorithms iteratively evolve a population of several individuals according to the principle of natural selection. Each individual encodes a particular solution and, at each generation, new individuals are obtained by selecting parents and combining their features. In order to obtain better and better solutions, a fitness value is associated to each individual: the fitter the individuals, the more they are likely to be selected as parents and to transmit their features to new generations. The Genetic Algorithm for GMLP (GAG) is sketched in Figure 4. Individuals are encoded as columns sequences, and the

1. *Determine individuals of the **initial population***
2. Repeat *(for each generation)*
3. Repeat *(for each offspring)*
4. ***Select** two parents*
5. *Generate offspring by **crossover***
6. *Apply **mutation** to offspring*
7. Until *a set of new individuals are generated*
8. ***Replace** old individuals with new ones*
9. ***Refine** the fittest individuals by local search*
10. ***Adapt** fitness criteria*
11. Until *termination conditions are satisfied*
12. Return *the best individual found.*

Figure 4: *Sketch of the Genetic Algorithm for GMLP.*

initial population is obtained in part heuristically, in part by random columns permutations (step 1). The operator to recombine individuals and obtain offspring for the new generation (steps 3 to 7) is the Order Crossover, borrowed from the Traveling Salesman Problem. After selecting two parents, two new individuals are generated: each individual inherits a subsequence from one parent and the remaining elements are filled-in in the relative order of the other parent. To avoid premature convergence, new individuals undergo a mutation, with a given probability: mutation exchanges

the position of two randomly chosen columns. The new generation is obtained by replacing with the new offspring all the individuals, but an elite set of the fittest ones and a steady set chosen at random (step 8). Before starting the next iteration, a refinement operator explores the 2-OPT neighborhood of most promising individuals and replaces them with local optima (step 9). GAG terminates after a fixed number of generations, returning the best individual found so far.

With respect to standard genetic algorithms, GAG introduces some new features, which have experimentally shown to significantly impact on its performance, and are mainly related to the fitness function definition and to the refinement operator. The fitness function is used to guide the selection mechanism and, according to the twofold aim of GAG, both MOS and TOS has to be taken into account. MOS is related to critical subsequences and is very unlikely to change under small sequence perturbations. Further indicators are thus necessary to discriminate fittest individuals and, as discussed in [7], TOS is not enough: in fact, both MOS and TOS measure the whole sequence and may hide good *local* features. We thus propose two new indicators, based on relations between close columns in a given sequence ϕ: *NEW*, which sums up the 1s in one column of M_ϕ not contained in the previous one, and *IOS*, the maximum increment in the number of 1s from one column of $\bar{M}_\phi$ to the following one. Summarizing, the fitness of an individual is a weighted sum of MOS, TOS, NEW and IOS. Further, we propose to dinamically change the weights during the evolution (step 10), and three settings are used to obtain different search phases: during the first generations, emphasis is on MOS optimization, with negligible weights to TOS, NEW and IOS; then GAG switches to a second setting, aiming at obtaining better TOS, while diversifying the population and emphasis is on TOS, NEW and IOS; finally, the search is guided again toward MOS optimization and the related weight is increased, to minimize λ and find a good feasible solution. Concerning the refinement operator, a standard implementation of the 2-OPT local search may be computationally expensive. Several speeding-up tricks has been devised, whose details are beyond the scope of this short paper. We just mention that the refinement is applied with a low frequency to a few individuals, and that an incremental neighbor evaluation has been implemented, based on some invariance properties of the stack matrix (the same incremental evaluation is applied to offspring generated by crossover).

3. EXACT BRANCH-AND-CUT PROCEDURE

Given a matrix $A \in \mathbb{R}^{m\times n}$, the *minor* A_{IJ} is the submatrix of A defined by the ordered subsets I and J of rows and columns, respectively. Let $[A]^{p,q}$ be the set of all minors of A of size $p \times q$. Given two matrices $A, B \in \mathbb{R}^{m\times n}$ in the following we will denote by $\langle A, B\rangle$ the inner product of A and B. A $\{0,1\}$-matrix A has the *consecutive ones property for rows* (or, briefly, A is C1P) if the columns of A can be permuted so to obtain a *strict* C1P matrix, that is a $\{0,1\}$ matrix such that in each row the ones appear consecutively, i.e. in each row they can not appear two 1s separated by one or more 0s. According to this definition we can now state our formulation for GMLP as follows: given $M \in \{0,1\}^{m\times n}$ and $\lambda \in \mathbb{Z}_+$, minimize $\sum_{i\in\{1,\dots,m\}, j\in\{1,\dots,n\}} X(i,j)$ with

$$X \text{ is C1P} \tag{1}$$

$$X(i,j) \geq M(i,j), \forall i \in 1,\dots,m,\ \forall j \in 1,\dots,n \tag{2}$$

$$\lambda \geq \sum_{i=1}^{m} X(i,j), \forall j \in 1,\dots,n \tag{3}$$

$$X \in \{0,1\}^{m\times n}. \tag{4}$$

A feasible solution X of the previous system is then a $\{0,1\}$-matrix (constraint (4)), obtained by turning 0s of M into 1s (constraints (2)), and such that there exists a sequence ϕ of its columns such that $X = \bar{M}_\phi$ (constraint (1)). Constraints (3) ensure that the number of stacks contemporary open by the solution X does not exceed the given value λ and the objective function corresponds to TOS. Still, in order to obtain an integer linear program, we have to translate constraint (1) into linear inequalities. Tucker [11] gave a characterization of the C1P matrices using five special matrices $T^1_k, T^2_k, T^3_k, T^4, T^5$, called *Tucker minor*. In particular, T^4 and T^5 have fixed dimension, while T^1_k, T^2_k, and T^3_k have dimension depending on parameter k (for example, the minor T^1_k for $k = 4$ is shown in Figure 5(a)). Tucker proved that a matrix $A \in \{0,1\}^{m\times n}$

$$\begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \qquad \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & -1 \\ -1 & 1 & 1 & 0 & 0 & 0 \\ -1 & 0 & 1 & 1 & 0 & 0 \\ -1 & 0 & 0 & 1 & 1 & 0 \\ -1 & 0 & 0 & 0 & 1 & 1 \\ 1 & -1 & 0 & 0 & 0 & 1 \end{pmatrix}$$

(a) (b)

Figure 5: *The Tucker minor T^1_4 (a) with the corresponding coefficients of the Oswald-Reinelt matrix F_{1_4} (b) defining the valid inequality $\langle F_{1_4}, X_{IJ}\rangle \leq 11$.*

is C1P if and only if none of its minors is a Tucker minor. More recently, Oswald and Reinelt used the Tucker characterization in order to provide a description of the C1P matrices in terms of linear integer programming. Indeed they first defined the $\{0,1,-1\}$ matrices F_{1_k}, F_{2_k}, F_3, and F_4 (see Figure 5(b) for an example) and proved the following:

Theorem 1 ([12, 13])**.** *A matrix $X \in \{0,1\}^{m\times n}$ is C1P if and only if all the following OR-inequalities are satisfied:*

$$\langle F_{1_k}, X_{IJ}\rangle \leq 2k+3,\ \forall X_{IJ} \in [A]^{k+2,k+2},\ \forall k \geq 1; \tag{5}$$

$$\langle F_{2_k}, X_{IJ}\rangle \leq 2k+3,\ \forall X_{IJ} \in [A]^{k+2,k+3},\ \forall k \geq 1; \tag{6}$$

$$\langle F_3, X_{IJ}\rangle \leq 2k+3,\ \forall X_{IJ} \in [A]^{4,6}; \tag{7}$$

$$\langle F_4, X_{IJ}\rangle \leq 2k+3,\ \forall X_{IJ} \in [A]^{4,5}; \tag{8}$$

We can then use such a characterization to get a linear integer formulation of GMLP by replacing constraint (1) with the set of inequalities (5),...,(8). Observe that here, differently from the formulation proposed by Baptiste in [6], one does not need to take explicitly into account the order of the columns of X. Therefore, let X^* be the optimal solution of such a linear integer optimization program. Then X^* is a C1P matrix and we can now apply the so-called PQ-tree procedure [14] that, in linear time, returns a columns sequence ϕ^* that turns X^* into a strict C1P matrix.

Observe here that, as it corresponds to the number of minors of the input matrix M, the number of constraints (5) and (6) grows exponentially with the size of M (the number of inequalities of type (7) and (8), even if not exponential, is bounded by a high polynomial in m and n). This implies that the proposed formulation cannot be used explicitly but its linear relaxation must be solved by a cutting planes procedure. Oswald and Reinelt [13] defined a polynomial time algorithm to exactly separate inequalities (5),..., (8), but here we implemented a heuristic separation routine that is similar to the one proposed in [12]. In particular, given a fractional solution $\tilde{X}$, we round its values to the corresponding closest integers so to obtain the matrix $\bar{X}$ and then, using the PQ-tree algorithm [14], we check if $\bar{X}$ is C1P. In case $\bar{X}$ is not C1P, the PQ-tree algorithm produces as output a Tucker minor of $\bar{X}$ and we use the corresponding Oswald and Reinelt inequality as a cutting plane. Although, because of the rounding procedure, the separation routine we implemented is not exact, all the integer solution that do not correspond to C1P matrices are cut off. This implies that the solution provided

by the branch-and-cut algorithm described above is the optimal solution of the GMLP instance given as input.

4. COMPUTATIONAL RESULTS

The proposed approach for GMLP has been implemented in C++ and run on a 2.1 GHz Intel Core2 processor. For the branch-and-cut procedure, we have used the SCIP 1.00.7 framework [15] and Cplex 11.0 as linear programming solver. The algorithm is currently under extensive test: in this abstract we present preliminary results on a benchmark of real instances from VLSI industry proposed in [5]. Concerning GAG, we have experimentally set the number of generations to $\min\{20n, 500\}$, the number of individuals to $\min\{10n, 500\}$ and, besides other parameters, the fitness function weights shown in Table 1. The results are reported in

Up to iteration	MOS	TOS	NEW	IOS
35%	0.70	0.16	0.07	0.07
50%	0.10	0.50	0.20	0.20
100%	0.95	0.05	0.00	0.00

Table 1: GAG fitness function weight settings.

Table 2 and compare GAG with the Constructive Genetic Algorithm [9] (CGA). Instance name and size are shown in the first column. Column λ is the threshold on MOS, corresponding to the minimum MOS found by GAG. The same MOS is also found by CGA and, for all the instances, it corresponds to proven optimal or best known (instance W4) MOS. Following columns summarize the results of 10 trials of CGA and GAG. SR_λ is the success rate, that is, the percentage of trials obtaining a MOS $= \lambda$. TOS, Avg and Dev are, respectively, the best found TOS, the average TOS and the standard deviation over the feasible sequences having MOS $= \lambda$. Note that Avg and Dev refer to the top five trials, as just this information is available from [9]. T(s) is the average computational time, in seconds, over all the 10 trials. The branch-and-cut procedure has been run, with a time limit of 1 hour, with the aim of improving over the TOS provided by GAG, or prove its optimality under the constraint MOS $\leq \lambda$: the last two columns of Table 2 report the obtained TOS (proven optima in bold) and the time to prove optimality or to find the improved solution (in italics). First, we observe that, for two instances, CGA provides non-feasible TOS (in italics), as they are below the optimal solution. For all the remaining instances but one, GAG provides better TOS. GAG shows also more reliable: it finds the best MOS more frequently than CGA and it has lower average TOS (except W4). Running times are comparable, taking into account that CGA ran on a 266 MHz processor. We remark that the TOS shown in Table 2 come from feasible sequences, that is, sequences whose MOS does not exceed λ. In fact, minimizing TOS and MOS is not equivalent, as shown in [2], and GAG was able to find non-feasible solutions with better TOS: for example, one trial on W4 obtained TOS $= 1633$ with MOS $= 28$ and one trial on v4000 obtained TOS $= 52$ with MOS $= 6$. Concerning B&C, it proves the optimality of four instances, and improves over the TOS provided by GAG in two cases (MOS is always equal to λ).

5. CONCLUSIONS

We have presented a genetic approach (GAG) and a branch-and-cut procedure (B&C) for GMLP, a pattern sequencing problem dealing with TOS minimization under restrictions on MOS. GAG introduces a dynamic weighted sum of TOS, MOS and other new performance indicators as fitness function, to take into account both global and local features of the pattern sequences. B&C is, to our best knowledge, the first algorithm designed to find proven optimal TOS under constraints on MOS: it is based on the properties of C1P matrices and it is flexible enough to accommodate different objectives or performance constraints. Preliminary results on real instances show that GAG normally outperforms previous literature results, and that, in some cases, B&C is able to prove the optimality of the proposed GMLP solutions. Ongoing research includes a better calibration of GAG parameters, extensive tests to better assess the performance of the approach, more sophisticated fitness function weights setting (cycling between settings, choosing settings based on landscape analysis etc.), and the improvement of B&C efficiency on large instances.

6. REFERENCES

[1] R. Möhring, "Graph problems related to gate matrix layout and PLA folding," *Computing*, vol. 7, pp. 17–51, 1990.

[2] A. Linhares and H. H. Yanasse, "Connections between cutting-pattern sequencing, VLSI design, and flexible machines," *Computers & Operations Research*, vol. 29, pp. 1759–1772, 2002.

[3] J. C. Becceneri, H. H. Yanasse, and N. Y. Soma, "A method for solving the minimization of the maximum number of open stacks problem within a cutting process," *Computers and Operations Reasearch*, vol. 31, pp. 2315–2332, 2004.

[4] G. Chu and P. J. Stuckey, "Minimizing the maximum number of open stacks by customer search," *Lecture Notes in Computer Science*, vol. 5732, pp. 242–257, 2009.

[5] Y. H. Hu and S. J. Chen, "GM_Plan: A Gate Matrix Layout Algorithm based on Artificial Intelligence Planning Techniques," *IEEE Transactions on Computer-Aided Design*, vol. 9, pp. 836–845, 1990.

[6] B. M. Smith and I. P. Gent, Eds., *Proceedings of IJCAI'05 – Constraint Modelling Challenge 2005*, Edimburgh, Jul. 31, 2005.

[7] L. De Giovanni, G. Massi, and F. Pezzella, "An adaptive genetic algorithm for large-size open stack problems," DMPA, Università di Padova, Tech. Rep., 2010.

[8] A. C. M. d. Oliveira and L. A. N. Lorena, "Pattern Sequencing Problems by Clustering Search," *Lecture Notes in Computer Science*, vol. 4140, pp. 218–227, 2006.

[9] ——, "A Constructive Genetic Algorithm for Gate Matrix Layout Problems," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 21, no. 8, pp. 969–974, 2002.

[10] A. Respício and M. E. Captivo, *Metaheuristics: Progress as Real Problem Solvers*. Ibaraki T., Nonobe K. and Yagiura M. (Eds.), Eds. Swets & Zeitlinger, 2005, ch. Bi-objective Sequencing of Cutting Patterns – An Application for the Paper Industry, pp. 227–241.

[11] A. Tucker, "A structure theorem for the consecutive 1's property," *J. Combinatorial Theory Ser. B*, vol. 12, pp. 153–162, 1972.

[12] M. Oswald and G. Reinelt, "Constructing new facets of the consecutive ones polytope," in *Combinatorial Optimization – Eureka, You Shrink! Papers Dedicated to Jack Edmonds, 5th International Workshop, Aussois, 2001*, ser. LNCS, M. Jünger, G. Reinelt, and G. Rinaldi, Eds. Springer-Verlag, 2003, vol. 2570, pp. 147–157.

[13] ——, "Computing optimal consecutive ones matrices," in *The Sharpest Cut, The Impact of Manfred Padberg and His Work*, ser. Optimization, M. Grötschel, Ed. MPS/SIAM, 2004, pp. 173–184.

		CGA					GAG					B&C	
Inst. ($m \times n$)	λ	SR_λ	TOS	Avg	Dev	T(s)	SR_λ	TOS	Avg	Dev	T(s)	TOS	T(s)
Wli (11×10)	4	100%	*18*	18.0	0.0%	0.5	100%	**24**	24.0	0.0%	0.0	**24**	5
Wsn (17×25)	8	100%	104	106.6	3.6%	1.5	100%	97	97.6	0.6%	0.3	96	*48*
v4000(10×17)	5	100%	*53*	53.3	1.7%	0.5	40%	58	58.3	5.0%	0.1	**56**	42
v4050(13×16)	5	100%	41	41.4	1.3%	0.5	100%	**38**	38.8	1.2%	0.1	**38**	23
v4090(23×27)	10	90%	95	96.8	1.7%	2.0	100%	109	109.0	0.0%	0.4	–	–
V4470(37×47)	9	100%	246	262.4	5.6%	66.5	100%	237	242.6	1.3%	4.0	–	–
X0 (40×48)	11	80%	303	305.2	0.6%	75.6	100%	298	298.8	0.1%	5.6	–	–
W1 (18×21)	4	100%	**39**	39.8	4.6%	1.0	100%	**39**	39.8	2.8%	0.2	**39**	4
W2 (48×33)	14	100%	235	257.2	8.5%	18.5	100%	233	233.0	0.0%	1.9	–	–
W3 (84×70)	18	50%	677	751.6	11.9%	306.3	100%	675	677.6	0.3%	82.2	–	–
W4 (202×141)	27	30%	1730	1805.0	3.3%	5224.7	70%	1701	2000.0	12.0%	94.6	–	–

– no optimal solution nor improvement after 1 hour computation

Table 2: Results on VLSI instances.

[14] K. S. Booth and G. S. Lueker, "Testing for the consecutive ones property, interval graphs, and graph planarity using pq-tree algorithms," *J. Comput. Syst. Sci.*, vol. 13, pp. 335–379, 1976.

[15] T. Achterberg, "Scip: Solving constraint integer programs," *Mathematical Programming Computation*, vol. 1, no. 1, July 2009.

An integer programming framework for sequencing cutting patterns based on interval graph completion

Isabel Cristina Lopes * † J.M. Valerio de Carvalho †

* ESEIG, Polytechnic Institute of Porto
Rua D.Sancho I, 981, Vila do Conde
cristinalopes@eu.ipp.pt

† Department of Production and Systems, University of Minho
Campus de Gualtar, Braga
vc@dps.uminho.pt

ABSTRACT

We derived a framework in integer programming, based on the properties of a linear ordering of the vertices in interval graphs, that acts as an edge completion model for obtaining interval graphs. This model can be applied to problems of sequencing cutting patterns, namely the minimization of open stacks problem (MOSP). By making small modifications in the objective function and using only some of the inequalities, the MOSP model is applied to another pattern sequencing problem that aims to minimize, not only the number of stacks, but also the order spread (the minimization of the stack occupation problem), and the model is tested.

Keywords: Integer programming, Interval graphs, Sequencing cutting patterns

1. INTRODUCTION

Cutting stock operations require advanced planning. The classic cutting stock problem consists in defining the cutting patterns with a cost minimization criterion that usually depends on the waste of the cutting process. But even after the cutting patterns are defined, there is more optimization that can be done in order to reduce the cost of the operations. The sequence in which the cutting patterns will be processed on the cutting equipment can be a relevant factor for the efficiency of the operations, for the organization of the work area space, for the fulfillment of the customers' orders on time, or for the fastness of the deliveries to customers. These concerns gave rise to several pattern sequencing problems, such as the minimization of open stacks and the minimization of the order spread.

In literature, pattern sequencing problems have been studied both alone and integrated with the determination of the cutting patterns. The most used approach is to solve the problem combining two stages, a first stage where the cutting patterns are defined and a second stage where the sequence of the implementation of the cutting patterns is decided. This work is devoted to the second stage, when the cutting patterns are already determined but the sequence in which they will be processed is still an open issue. The main problem addressed is the minimization of the maximum number of open stacks, also called MOSP.

This problem has been widely studied in literature, but there are several other pattern sequencing problems, such as the minimization of the order spread (MORP) and the minimization of discontinuities (MDP).

The Minimization of Open Stacks Problem (MOSP) comes from the flat glass cutting industry, but it also has many applications in other cutting industries (wooden panels, steel tubes, paper,...) as well as in other fields such as production planning, VLSI circuit design and in classic problems from graph theory. The MOSP problem is based on the premise that the different items obtained from cutting patterns are piled in stacks in the work area until all items of the same size have been cut. Usually, machines process one cutting pattern at a time and the sequence in which preset cutting patterns are processed can affect the number of stacks that remain around the machine.

Due to space limitations and danger of damages on the stacked items, it is advantageous to find a sequence for the patterns that minimizes the number of different items that are being cut and therefore the number of open stacks.

The minimization of open stacks problem is known to have tight relations with problems in graph theory such as treewidth, vertex separation and the profile of a matrix. In studying these problems, we found a type of graphs called interval graphs that can play an important role in this work.

An *interval graph* is an undirected graph G such as its vertices can be put into a one-to-one correspondence with a set of intervals I of a linearly ordered set (like the real line) such that two vertices are connected by an edge of G if and only if their corresponding intervals have nonempty intersection. I is called an *interval representation* for G. [1]

These graphs can be used to describe a solution of the pattern sequencing problems, by modeling the duration of the intervals in time in which the same piece type is being cut. Using several properties of this type of graphs we will see that it is possible to derive a general framework that can be used to model the minimization of open stacks problem and to model many related problems.

MOSP is modeled as an interval graph completion problem. An initial integer programming model was derived, using the addition of arcs to the graph and the properties of interval graphs to achieve a solution, and based on the following characterization of interval graphs by Olariu:

A graph $G = (V,E)$ is an interval graph if and only if there exists a linear ordering $\varphi : V \to \{1,...,N\}$ such that $\forall i,j,k \in V : \varphi(i) < \varphi(j) < \varphi(k)$ we have $[ik] \in E \Rightarrow [ij] \in E$. [2]

The model is strengthened with inequalities derived from the relationship between the chromatic number of a graph and the number of intersecting intervals.

The MOSP model is applied to different problems. By making small modifications in the objective function and using only some of the inequalities, the MOSP model is applied to the minimum interval graph completion problem. Another pattern sequencing

problem that aims to minimize, not only the number of stacks, but also the order spread (the minimization of the stack occupation problem) is considered, and the model is tested.

There is also another pattern sequencing problem called the Minimization of Tool Switches (MTSP) which is addressed with this framework, using the similarities between this problem and the MOSP, but for this problem the model has a limited use.

With the choice being integer programming, the formulation developed in this work can later be integrated in other integer programming models for cutting stock problems, namely to create a combined model of the stages one and two where the cutting stock patterns are defined and sequenced.

2. MODELING THE MINIMIZATION OF OPEN STACKS

Consider a cutting machine that processes just one cutting pattern at a time. The items already cut that are equal are piled in stacks by the machine. The stack of an item type remains near the machine if there are more items of that type to be cut in a forthcoming pattern. A stack is closed and removed from the work area only after all items of that size have been cut, and immediately before starting to process the next cutting pattern. After a pattern is completely cut and before any stack is removed the number of open stacks is counted. The maximum number of open stacks for that sequence of patterns is called the *MOSP number*.

There are often space limitations around the cutting machines, there is danger of damages on the stacked items, difficulty in distinguishing similar items, and in some cases there are handling costs of removing the stack temporarily to the warehouse. It is advantageous to minimize the number of open stacks, and that can be done simply by finding an optimal sequence to process the cutting patterns.

MOSP has been proved to be a NP-hard problem [3].

As suggested in [4], an instance of the MOSP can be associated with a graph having a vertex for each item that is cut and an edge between two vertices if the corresponding items are present in the same cutting pattern.

To optimize the number of stacks, it is convenient to find the best sequence to process the cutting patterns. Considering that the patterns do not appear explicitly in the MOSP graph constructed in this way, how will we find that sequence for the cutting patterns? We will focus on finding a sequence to open the stacks, rather than on sequencing the cutting patterns. That is not a problem, because it is possible to take a solution for the ordering of the vertices of the graph and construct a sequence for the corresponding cutting patterns [5].

Given an instance of the problem, we first build a graph $G = (V,E)$, associating each item cut from the patterns to a vertex and creating an arc joining vertex i and j if and only if items i and j are cut from the same pattern. This graph may not be an interval graph at the start, but we will add some arcs to it in such a way that it will become one. We need this graph to become an interval graph because, if we associate each item to the interval of time in which the stack of that item is open, we can use the graph to model what intervals should occur simultaneously and what intervals should precede others. According to the sequence in which the cutting patterns are processed, there may be more or less open stacks simultaneously. Each arc of the future interval graph means that, for a period of time, the two stacks (the respective vertices of the arc) will remain both open. The initial graph contains only the arcs that must be there, in any possible sequence in which the patterns can be processed. The rest of the arcs that are added later to the graph will differ according to the sequence of the patterns. It is the choice of these arcs that defines which are the other simultaneously open stacks. Our model for this problem consists in finding out which edges should be added to the original MOSP graph $G = (V,E)$ in order to get an interval graph $H = (V, E \cup F)$ that minimizes the maximum number of simultaneously open stacks.

2.1. The variables

We set an ordering for opening the stacks by assigning a number to each item cut, with a bijective function $\varphi : V \to \{1,...,N\}$. This linear ordering of the vertices is set by the decision variables x_{ij}:

$$x_{ij} = \begin{cases} 1 & \text{if } \varphi(i) < \varphi(j) \\ 0 & \text{otherwise} \end{cases} \quad \forall i,j \in V$$

Notice that $x_{ii} = 0$ for any $i \in V$ and also that we have

$$x_{ij} = 1 \Leftrightarrow x_{ji} = 0$$

These variables are setting an orientation into the arcs, for us to keep track of the sequence of the items in the current instance. If $x_{ij} = 1$ then item i starts being cut before the item j is, even though the corresponding stacks may overlap or not, i.e., in spite of having an arc between the two vertices or not.

The other decision variables that will be used are concerned to the arcs that are necessary to add to the original graph $G = (V,E)$ to get an interval graph $H = (V, E \cup F)$ and, together with variables x, determine which intervals will overlap in the desired interval graph. To decide which of these additional arcs are to be added, we define a variable y_{ij} for each arc $[ij]$ that did not exist before in the graph:

$$y_{ij} = \begin{cases} 1 & \text{if } [ij] \notin F \text{ and } \varphi(i) < \varphi(j) \\ 0 & \text{if } [ij] \in F \text{ or } \varphi(i) \geq \varphi(j) \end{cases} \quad {\scriptstyle \forall i,j \in V : [ij] \notin E}$$

Notice that y_{ij} is 1 when the arc $[ij]$ is NOT added, because the variable y_{ij} works like an "eraser"variable. To get an interval graph, if we decided to add to the original graph all the arcs that were missing, and then remove some of them - the ones that we do not need to have an interval graph, then variable y is 1 for these additional arcs which are to be removed.

Variables y depend on the linear ordering of vertices, so it follows that there is an anti-reflexive relation:

$$y_{ij} = 1 \Rightarrow y_{ji} = 0$$

When $y_{ij} = 1$, the arc $[ij]$ is not needed in the interval graph, so, by definition of interval graph, if there is not an arc $[ij]$, then the intervals i and j do not intersect. Consequently, one of the intervals should finish before the other one starts. As $i \prec j$, the interval i opens and finishes before the interval j starts. It means that the stacks for items i and j will never be open at the same time, so they can share the same stack space.

To explain the relations between the intervals horizontally, we will add an extra set of variables z, based on the asymmetric representatives formulation for the vertex coloring problem by Campêlo et al. [6]. The value of the optimum of the MOSP is equal to the size of the biggest clique in the solution graph $\omega(H)$ and, because interval graphs are perfect graphs, it is equal to the chromatic number of the graph $\chi(H)$, which is the number of colors needed to assign to the vertices of the graph such that there are no two adjacent vertices of the same color.

If we assign colors to the vertices of the desired interval graph, such that no two adjacent vertices have the same color, we can count the maximum number of simultaneously open stacks by counting the minimum number of different colors needed, because simultaneously open stacks will get different colors, and stacks that do not overlap can have the same color.

The variables that we will use are:

$$z_{ij} = \begin{cases} 1 & \text{if vertex } i \text{ represents vertex } j \\ 0 & \text{otherwise} \end{cases} \qquad \forall i, j \in V : [ij] \notin E$$

Note that if $i \in V$ is a representative vertex then $z_{ii} = 1$.

We will use the variable $K \in \mathbb{N}$ to denote the maximum number of simultaneously open stacks.

2.2. The main model

Using this variables we present the following integer programming model for the MOSP:

Minimize K

Subject to:

$$0 \leq x_{ij} + x_{jk} - x_{ik} \leq 1 \qquad \forall i,j,k = 1,...,N, i < j < k \qquad (1)$$

$$y_{ij} - x_{ij} \leq 0 \qquad \forall i,j = 1,...,N, i < j, [ij] \notin E \qquad (2)$$

$$y_{ij} + x_{ji} \leq 1 \qquad \forall i,j = 1,...,N, j < i, [ij] \notin E \qquad (3)$$

$$y_{ij} - x_{kj} \leq 0 \qquad \forall i,j,k = 1,...,N, k < j, [ij] \notin E, [ik] \in E \qquad (4)$$

$$y_{ij} + x_{jk} \leq 1 \qquad \forall i,j,k = 1,...,N, j < k, [ij] \notin E, [ik] \in E \qquad (5)$$

$$0 \leq y_{ik} - y_{ij} + x_{kj} \leq 1 \qquad \forall i,j,k = 1,...,N, k < j, [ij], [ik] \notin E \qquad (6)$$

$$0 \leq y_{ij} - y_{ik} + x_{jk} \leq 1 \qquad \forall i,j,k = 1,...,N, j < k, [ij], [ik] \notin E \qquad (7)$$

$$\sum_{i=1}^{j-1} x_{ij} + \sum_{i=j+1}^{N} (1 - x_{ji}) - \sum_{\substack{i=1 \\ [ij] \notin E}}^{N} y_{ij} + 1 \leq K \qquad \forall j = 1,...,N \qquad (8)$$

$$y_{ij} + y_{ki} \leq 1 \qquad \forall i,j,k = 1,...,N \text{ with } [ij], [ik] \notin E, [jk] \in E \qquad (9)$$

$$y_{ij} + y_{jk} \leq 1 \qquad \forall i,j,k = 1,...,N \text{ with } [ij], [jk] \notin E, [ik] \in E \qquad (10)$$

$$y_{ij} + y_{lk} \leq 1 \qquad \forall i,j,k,l = 1,...,N \text{ with } [ij], [kl] \notin E, [jl], [ik] \in E \qquad (11)$$

$$y_{ij} + y_{jk} - y_{ik} \leq 1 \qquad \forall i,j,k = 1,...,N \text{ with } [ij], [jk], [ik] \notin E \qquad (12)$$

$$y_{ik} + y_{ki} + y_{jl} + y_{lj} \leq 1 \qquad \forall i,j,k,l = 1,...,N \text{ with } i \neq j \neq k \neq l, \; [ik], [jl] \notin E, [ij], [jk], [kl], [li] \in E \qquad (13)$$

$$y_{il} + y_{li} + y_{ik} + y_{ki} + y_{jl} + y_{lj} + y_{jm} + y_{mj} + y_{mk} + y_{km} \leq 3 \qquad \forall i,j,k,l,m = 1,...,N \text{ with } i \neq j \neq k \neq l \neq m, \; [ik], [il], [jl], [jm], [km] \notin E, [ij], [jk], [kl], [lm], [mi] \in E \qquad (14)$$

$$\sum_{i=1}^{N} z_{ii} = K \qquad (15)$$

$$\sum_{\substack{i=1 \\ [ij] \notin E}}^{N} \sum_{\substack{j=1 \\ [ij] \notin E}}^{N} z_{ij} = N \qquad (16)$$

$$\sum_{\substack{i=1 \\ [ij] \notin E}}^{N} z_{ij} = 1 \qquad \forall j = 1,...,N \qquad (17)$$

$$z_{ij} \leq y_{ij} \qquad \forall i,j = 1,...,N \text{ with } [ij] \notin E \qquad (18)$$

$$z_{ij} + z_{ik} - y_{jk} - y_{kj} \leq 1 \qquad \forall i,j,k = 1,...,N \text{ with } [ij], [ik], [jk] \notin E \qquad (19)$$

$$z_{ij} \leq z_{ii} \qquad \forall i,j = 1,...,N \text{ with } [ij] \notin E \qquad (20)$$

$$z_{ij} + z_{ik} \leq z_{ii} \qquad \forall i,j,k = 1,...,N \text{ with } j < k, [ij], [ik] \notin E, [jk] \in E \qquad (21)$$

$$z_{ij} + z_{ik} + z_{il} \leq z_{ii} \qquad \forall i,j,k,l = 1,...,N \text{ with } j < k < l, \; [ij], [ik], [il] \notin E, [jk], [kl], [lj] \in E \qquad (22)$$

$$z_{ij} + z_{ik} + z_{il} + z_{im} \leq z_{ii} \qquad \forall i,j,k,l,m = 1,...,N \text{ with } j < k, j < l, k < m, \; [ij], [ik], [il], [im] \notin E, [jk], [jl], [jm], [kl], [km], [lm] \in E \qquad (23)$$

$$z_{il} + z_{li} + z_{ik} + z_{ki} + z_{jl} + z_{lj} + z_{jm} + z_{mj} + z_{mk} + z_{km} \leq 2 \qquad \forall i,j,k,l,m = 1,...,N \text{ with } i \neq j \neq k \neq l \neq m, \; [ik], [il], [jl], [jm], [km] \notin E, [ij], [jk], [kl], [lm], [mi] \in E \qquad (24)$$

$$x_{ij} \in \{0,1\} \qquad \forall i,j = 1,...,N \text{ with } i < j \qquad (25)$$

$$y_{ij} \in \{0,1\} \qquad \forall i,j = 1,...,N \text{ with } i \neq j, [ij] \notin E \qquad (26)$$

$$z_{ij} \in \{0,1\} \qquad \forall i,j = 1,...,N \text{ with } [ij] \notin E \qquad (27)$$

$$K \in \mathbb{N} \qquad (28)$$

Having developed a fully functional integer programming model for the minimization of open stacks problem, we then explore some variants of this model.

3. MINIMUM INTERVAL GRAPH COMPLETION

The main idea behind the integer programming model presented is the completion of the MOSP graph with suitable fill edges, with the purpose of constructing an interval graph. There are several edge completion problems documented in literature [7]. Here we address the Minimum Interval Graph Completion, which searches for the minimum number of fill edges that should be added to a graph to obtain an interval graph. With small changes in the objective function and using some of the previous constraints, we can build an integer programming model for this problem in Graph Theory.

We will not need the variables z_{ij} because the number of stacks is irrelevant in the minimum interval graph completion problem. Therefore, inequalities (8), (15) to (24), (27) and (28) are droped for this case.

The objective is simply completing the graph with the smallest number of edges to obtain an interval graph. The sum of all variables y gives the number of edges that are not added to the graph G when completing it to an interval graph H. By maximizing this sum, we get a minimum number of added edges.

More formally, the objective function for the minimum interval graph completion problem is

$$\max \sum_{[ij] \notin E} y_{ij} \qquad (29)$$

4. MINIMIZING THE STACK OCCUPATION

The model we have developed for the minimization of open stacks can be used in another pattern sequencing problem, where the objective is to find an optimal sequence to process the cutting patterns in order to minimize the occupation of the stacks.

The problem we address now is similar to minimizing the flow time of the orders: besides having the minimum number of open stacks, we also want to minimize the sum of the time that the stacks remain open within the system.

The sequence in which preset cutting patterns are processed can affect the flow and total completion time, so it is desirable to optimize the occupation of the stacks to eliminate unnecessary dispersion.

When considering the MOSP, it is usual to find more than one optimal solution, in the sense that there is more than one sequence of the cutting patterns that achieves the same maximum number of open stacks. We may be interested in choosing between these optimal solutions of the MOSP according to a different criterion. A natural choice is the minimization of the order spread.

Noticing that in most instances there are alternative optimal solutions for the MOSP, we tried to take the problem further and added a second step with a new objective function: the minimization of the order spread. This pattern sequencing problem similar to the MOSP is also related with the minimum interval graph completion problem.

Our model consists in finding out which arcs should be added to the original MOSP graph $G = (V,E)$ in order to get an interval graph $H = (V, E \cup F)$ that minimizes the stack occupation while keeping the minimum number of simultaneously open stacks.

The model we present is divided in two steps. In a first step, the

minimum number of open stacks is determined, and then in a second step, we search for a new sequence of the patterns that improves the total stack spread while using the optimal number of open stacks.

In the first step the formulation is the same as before, with the objective to minimize the maximum number of open stacks. Then, in the second step, the objective becomes the minimization of the stack spread. To minimize the average order spread is equivalent to minimizing the total stack spread. This is also equivalent to minimizing the number of fill-in zeros obtained in the matrix of the description of the cutting patterns after the columns have been rearranged to match the sequence in which the patterns will be processed.

This is done by minimizing the number of arcs that are added to the MOSP graph in order to obtain an interval graph. As the variables y_{ij} are 1 when an arc is not added to the graph, we can minimize the number of added arcs by maximizing the sum of the variables y_{ij}. Therefore the objective function in step 2 is expression (29).

To guarantee that the optimal number of open stacks does not increase from step 1 to step 2, some of the inequalities have to be modified accordingly. Let us denote the optimal number of open stacks found in step 1 by $MOSP^*$. For step 2, in the inequalities (8) and (15), the variable K is replaced by $MOSP^*$.

5. COMPUTATIONAL RESULTS

The integer programming models were tested on the instances of the Constraint Modeling Challenge 2005, available at: http://www.cs.st-andrews.ac.uk/ ipg/challenge/instances.html

The instances were provided by the participants in the challenge and present different kinds of difficulty, such as size, sparseness and symmetry. Computational tests were performed with ILOG OPL Development Studio 5.5 on an Intel®Core2 Duo T7200 @2.00GHz 0.99GB RAM. For each instance, the best objective value found by the model, the best lower bound, the gap, the number of nodes of the search tree and the runtime were recorded.

In small instances we found the optimal solution for MOSP in just a few seconds. In larger instances we found the optimal solution in a few seconds as well, but it takes too long to prove that it is optimal, specially in instances with many symmetries. In really large instances the models could not be started because there was not enough memory to handle so many variables and inequalities.

For the problem of minimizing the stack occupation, in the second step we were able to obtain the optimal solution in every instances tested. This second step allowed to reduce the order spread in almost every instance, while maintaining the same optimal number of open stacks. This reduction was very significant in many cases, decreasing around 75% of the number of added edges.

For the Minimum Interval Graph Completion Problem, in all of the instances tested, the optimal solution was reached and proved optimal.

6. ACKNOWLEDGEMENTS

This work was financially supported by the Portuguese Foundation for Science and Technology (FCT) and supported by ESEIG - Superior School of Industrial Studies and Management - Polytechnic Institute of Porto.

7. REFERENCES

[1] M. C. Golumbic, *Algorithmic graph theory and perfect graphs*. New York: Academic Press, 1980.

[2] D. G. Corneil, S. Olariu, and L. Stewart, "The ultimate interval graph recognition algorithm? (Extended Abstract)," in *Symposium on Discrete Algorithms*, 1998, pp. 175–180.

[3] A. Linhares and H. H. Yanasse, "Connections between cutting-pattern sequencing, VLSI design, and flexible machines," *Computers & Operations Research*, vol. 29, no. 12, pp. 1759–1772, 2002.

[4] H. H. Yanasse, "Minimization of open orders - polynomial algorithms for some special cases," *Pesquisa Operacional*, vol. 16, no. 1, pp. 1–26, June 1996.

[5] ——, "A transformation for solving a pattern sequencing problem in the wood cut industry," *Pesquisa Operacional*, vol. 17, no. 1, pp. 57–70, 1997.

[6] M. Campêlo, V. A. Campos, and R. C. Corrêa, "On the asymmetric representatives formulation for the vertex coloring problem," *Discrete Applied Mathematics*, vol. 156, no. 7, pp. 1097 – 1111, 2008, GRACO 2005 - 2nd Brazilian Symposium on Graphs, Algorithms and Combinatorics.

[7] M. C. Golumbic, H. Kaplan, and R. Shamir, "On the complexity of DNA physical mapping," *Advances in Applied Mathematics*, 1994.

OPTFRAME: A Computational Framework for Combinatorial Optimization Problems

Igor Machado Coelho [*] Pablo Luiz Araujo Munhoz [*] Matheus Nohra Haddad [†]
Vitor Nazario Coelho [†] Marcos de Melo Silva [*] Marcone Jamilson Freitas Souza [†]
Luiz Satoru Ochi [*]

[*] Fluminense Federal University, UFF
Niteroi, RJ, Brazil
{imcoelho, pmunhoz, mmsilva, satoru}@ic.uff.br

[†] Federal University of Ouro Preto
Ouro Preto, MG, Brazil
{mathaddad, vncoelho}@gmail.com, marcone@iceb.ufop.br

ABSTRACT

This work presents OptFrame, a computational framework for the development of efficient heuristic based algorithms. The objective is to provide a simple C++ interface for common components of trajectory and population based metaheuristics, in order to solve combinatorial optimization problems. Since many methods are very common in literature, we provide efficient implementations for simple versions of these methods but the user can develop "smarter" versions of the methods considering problem-specific characteristics. Moreover, parallel support for both shared-memory and distributed-memory computers is provided. OptFrame has been successfully applied to model and solve some combinatorial problems, showing a good balance between flexibility and efficiency.

Keywords: Framework, Metaheuristics, General Variable Neighborhood Search, TSP, Eternity II

1. INTRODUCTION

In the development of optimization systems it is common to face up with combinatorial NP-Hard problems. To produce algorithms that solve such problems is often a hard and long task, since the algorithm must solve the problem with low gaps in short computational time. That is, the heuristic algorithm must find good solutions at each execution. The solutions should be good enough for the application that uses the method and the elapsed time to generate them must be acceptable in terms of the application. One way of speeding up the development of such algorithms is by using tools that provide classic algorithms for combinatorial problems, both in practical and theoretical cases. This fact often motivates the use of a framework.

The architecture of a framework, that typically follows the object-oriented paradigm, defines a model for code reuse [1]. This fact justifies the development of frameworks that seek to find good solutions for optimization problems by means of heuristics and metaheuristics. Mainly because metaheuristics are essentially independent of the addressed problem structure. In the context of metaheuristics development, the developers that do not use any framework or library in general expend much effort by writing and rewriting code. Thus, the focus that should be at the problem and its efficient resolution is often directed to many programming aspects.

This work presents OptFrame[1], a white-box object oriented framework in C++ for the development of efficient heuristic based algorithms. Our objective is to provide a simple interface for common components of trajectory and population based metaheuristics. Since many methods are very used in literature we provide efficient implementations for simple versions of these methods but the user can develop *smarter* versions of the methods considering problem-specific characteristics.

The present work is organized as follows. Section 2 describes some optimization frameworks in literature. Section 3 defines important optimization concepts about metaheuristics that are behind OptFrame architecture. In Section 4 we present OptFrame architecture in details. Section 5 concludes the work with some applications and benchmarks on the framework.

2. FRAMEWORKS IN OPTIMIZATION

Many authors have already proposed frameworks for optimization problems, among which we cite: *TabOO Builder* [2], *NP-Opt* [3], *HotFrame* [1], *EasyLocal++* [4], ParadisEO [5], *iOpt* [6] and jMetal [7]. Now, we present some of them in details.

In [3] it is presented NP-Opt, a computational framework for NP class problems. The framework proposes to minimize code rewriting when the focused problem is changed. NP-Opt supports five distinct problems: Single Machine Scheduling, Parallel Machine Scheduling, Flowshop Scheduling with job families, Grid Matrix Layout (VLSI design) and non-linear continuous function optimization. The built-in heuristic methods are based on Memetic and Genetic Algorithms, so as Multiple Start. The authors of NP-Opt points to a code reuse of 75% when dealing with a new problem. The framework is programmed in Java language.

[1] present the C++ computational framework HotFrame, that shares some similarities with OptFrame, proposed in this work. HotFrame, so as OptFrame, was firstly designed for Iterated Local Search, Simulated Annealing and Tabu Search metaheuristics. And also in this sense HotFrame is very complete, since the authors show many implementation details and many variations of these metaheuristics. According to the authors a framework provides adaptable software components, which encapsulate common domain abstractions. To develop a framework requires solid knowledge in the considered domain.

[1] OptFrame website: http://sourceforge.net/projects/optframe/

[4] point that local search is a common interest theme of scientific community, at the same time that there isn't a standard software in this sense. So, the authors propose EasyLocal++, a computational object-oriented framework for the design and analysis of local search algorithms. According to the authors the architecture of EasyLocal++ allows code modularization and the combination of basic techniques and neighborhood structures. Some successful applications of EasyLocal++ are showed and according to the authors EasyLocal++ provides flexibility enough for the implementation of many scheduling problems.

ParadisEO [5] is a white-box object-oriented framework written in C++ and dedicated to the reusable design of parallel and distributed metaheuristics. This framework is based on a conceptual separation of the solution methods from the problems they are intended to solve. According to the authors, this separation gives the users maximum code and design reuse. ParadisEO provides some modules that deals with population based metaheuristics, multiobjective optimization, single-solution based metaheuristics, and it also provides tools for the design of parallel and distributed metaheuristics. ParadisEO, as the OptFrame, is one of the rare frameworks that provide parallel and distributed models. Their implementation is portable on distributed-memory machines as well as on shared-memory multiprocessors, as it uses standard libraries such as MPI, PVM and PThreads.

The Intelligent Optimization Toolkit (iOpt), proposed by [6] can be seen as an IDE for the rapid construction of combinatorial problems. The iOpt takes as input problems modeled in one-way constraints and uses metaheuristics to solve them. The authors show how to model the Vehicle Routing Problem with iOpt and good results are reported. Finally, the authors conclude that a better understanding of the problem can be achieved by a fairer comparison between heuristic methods.

jMetal [7] is an object-oriented Java-based framework aimed at facilitating the development of metaheuristics for solving multiobjective optimization problems (MOPs). According to the authors, this framework provides a rich set of classes which can be used as the building blocks of multi-objective metaheuristics; thus, taking advantage of code-reusing, the algorithms share the same base components, such as implementations of genetic operators and density estimators, so making the fair comparison of different metaheuristics for MOPs possible.

In general, frameworks are based on previous experience with the implementation of many methods for different problems. In this work we also review some important concepts of combinatorial problems and metaheuristics, in order to propose an architecture that is both problem and heuristic independent. The following section shows the theoretical modeling of combinatorial problems behind OptFrame architecture.

3. METAHEURISTICS

We present now some important concepts of metaheuristics and combinatorial optimization problems.

Let S be a set of discrete variables s (called *solutions*) and $f : S \to \mathbb{R}$ an objective function that associates each solution $s \in S$ to a real value $f(s)$. We seek any $s^* \in S$ such that $f(s^*) \leqslant f(s), \forall s \in S$ for minimization problems, or $f(s^*) \geqslant f(s), \forall s \in S$ for maximization problems. The solution s^* is called a *global optimum*.

A function N associates a solution $s \in S$ to a set $N(s) \subseteq S$ (called *neighborhood* of s). This is also an important concept in the subject of heuristic based algorithms. This way, a *neighbor* s' of s is such that $s' = s \oplus m$, where m is called a *move* operation. The cost of a move m is defined as $\widehat{f} = f(s') - f(s)$, which means that $s' = s \oplus m \implies f(s') = f(s) + \widehat{f}$. So, a *local optimum* (in terms of a neighborhood N) is a solution s' such that $f(s') \leqslant f(s), \forall s \in N(s')$ for minimization problems, or $f(s') \geqslant f(s), \forall s \in N(s')$ for maximization problems.

Many combinatorial optimization problems are classified as NP-Hard and it is common to use heuristic algorithms to find good solutions for these problems. These methods have the capability of finding good local optimums in short computational times. Classical local search heuristics stop on the first local optimum found. However, metaheuristics can go beyond the local optimum and thus these methods are able to produce final solutions of better quality.

4. OPTFRAME

OptFrame is a white-box object oriented framework in C++. In the following sections its implementation and design aspects are presented and discussed.

4.1. Representation and Memory

The OptFrame framework is mainly based on two important structures: the solution representation and the memory.

The Representation is the data structure used to represent a valid solution for a specific problem. For example, for the Traveling Salesman Problem (TSP) [8] a user may wish to represent the solution as an array of integers. In this case, the representation in this heuristic approach for TSP is $vector < int >$ (in C++ language).

On the other hand, the Memory is a set of auxiliary data structures needed for a *smarter* version of the method.

4.2. Solution and Evaluation

There are two important *container* classes[2] in OptFrame: Solution and Evaluation. Solution carries a reference to a Representation of the problem, while a Evaluation carries a reference to a Memory structure. To develop a *smarter* version of the method, the information in the Memory structure along with an earlier evaluation can be used to reevaluate a Solution in a faster way, for example.

4.3. Evaluators

The Evaluator concept is very important in OptFrame. It encapsulates the function $f : S \to \mathbb{R}$ (defined in Section 3) as an specific case of its function $f : S \to E$, where $E = (\mathbb{R}, \mathbb{R}, M)$. The tuple E can be seen as the Evaluation class defined in Subsection 4.2.

The first value of the tuple E is the *objective function value* itself and the second one is an *infeasibility measure value*. By evaluating a solution this way you can implement heuristic methods that *are able to see* unfeasible solutions, by giving a high penalty value to the *infeasibility measure value*. When the *infeasibility measure value* is zero the solution is considered *feasible*. So, the *evaluation function value* over a solution consists in the sum of $objective\_function\_value + infeasibility\_measure\_value$.

The third value M of the tuple E is called *memory* defined in Subsection 4.1. In this context the *memory* can record some steps of the evaluation algorithm, so they won't be repeated in future evaluations. This way, some future computational effort can be avoided.

[2] What we name here as a *container* class is in some ways related to with Proxy Pattern [9] since the idea is to carry a reference to an object (representation or memory) and to delete it when the container itself is destroyed. But in this case a *container* is also used to provide some extra operations over the carried object like *printing*, *reference counting* and *cloning*.

There is also a more general definition for the *evaluation* method where the function f is defined by $f : (S,E) \rightarrow E$. This way it is possible to develop *smarter* versions of a Evaluator by using informations of a previous evaluation E.

4.4. Moves

A move operation defines a neighborhood structure. In OptFrame the Move class has two most important methods: *canBeApplied* and *apply*.

The *canBeApplied* method of a Move object m returns *true* if the application of m to a solution s will produce a valid solution. Otherwise it returns false. This is method is often used before the *apply* method.

The *apply* method of a Move m to a solution s transforms s into a neighbor s' and returns another Move $\overline{m}$ that can undo the changes made by m. Since complete copies of solutions are expensive operations it is possible to avoid them by developing efficient implementations of the reverse Move $\overline{m}$.

4.5. Neighborhood Structures

There are three types of neighborhood structure in OptFrame: NS, NSSeq and NSEnum.

NS is the simplest definition of a neighborhood structure. It only requires the user to define a $move(s)$ method, that returns a random move operation of the neighborhood type. Although not in focus of this paper, it is possible to define neighborhood structures for continuous problems optimization using this kind of structure.

NSSeq is a more elaborated version of NS. It also requires the user to define a $getIterator(s)$ method, that returns an object capable of generating moves of the neighborhood structure in a sequential way. The returned object must implement the NSIterator interface, that itself implements the Iterator Pattern [9].

NSEnum is the most complete definition of a neighborhood structure in OptFrame. It provides an enumerable set of move operations for a given combinatorial problem. Although it only requires the user to define the $move(int)$ and $size()$ methods, with these methods it is possible to define default implementations for the $move(s)$ and $getIterator(s)$ methods of NS and NSSeq.

4.6. Heuristic based methods

Heuristic methods are mainly divided in two classes: *trajectory based* and *population based* methods [10].

In order to maximize the code reuse and to favor testing of Hybrid Metaheuristics [11], all heuristic methods should be implemented using the Heuristic class abstraction. With this abstraction we have already been able to implement the following methods: First Improvement, Best Improvement, Hill Climbing and other classical heuristic strategies [12]; Iterated Local Search, Simulated Annealing, Tabu Search, Variable Neighborhood Search and other basic versions of many famous trajectory based metaheuristics [13]; and, finally, the basic versions of population based metaheuristics Genetic Algorithm and Memetic Algorithm [13].

So, there are four definitions of the method *exec* and the user must implement at least two of them. For trajectory based heuristics, the user must implement:

```
void exec(Solution){ ... }
void exec(Solution, Evaluation){ ... }
```

For population based heuristics:

```
void exec(Population){ ... }
void exec(Population, FitnessValues){ ... }
```

where: `Population` is a list of `Solutions` and `FitnessValues` is a list of `Evaluations`.

The first one is the simplest version of the method while the second is a more elaborated version. But if the user wish to implement only one of them it is possible to implement one and the other one only calls the first.

4.7. Other structures

Some metaheuristics may require specific structures, but they can also be defined in specific files, e.g., Perturbation for Iterated Local Search; Mutation and Crossover operators for Genetic and Memetic Algorithms.

5. COMPUTATIONAL EXPERIMENTS AND CONCLUDING REMARKS

This work presents OptFrame, a white-box object oriented framework in C++ for the development of efficient heuristic based algorithms. Our objective is to provide a simple interface for common components of trajectory and population based metaheuristics.

OptFrame's architecture is intended to minimize the differences among code and theoretical concepts of combinatorial optimization. Thus, this paper describes a C++ modeling of the framework, but this model can also be applied to other programming languages, since generic programming features are available.

As a benchmark for the framework, we propose to implement a heuristic algorithm based on General Variable Neighborhood Search [14] for two different optimization problems.

The first problem is the classical Traveling Salesman Problem, and the second is the Eternity II Puzzle optimization problem (more details on [15]). We also want to show the flexibility of the developed interface by implementing the proposed heuristic in two different programming languages: C++ and Java[3].

To guarantee that the algorithms will follow the same paths (even on different languages), we have implemented the Mersenne Twister [16] random number generator, using the same seeds for both tests.

Table 1 shows the average time (in seconds) of 10 executions of the proposed algorithm. "Java GCJ" is a compiled version of the Java code (using the most optimized flags); "Java JRE" is an interpreted version of the Java code; and "C++" is a compiled version of the code using GCC compiler (with the most optimized flags).

Table 1: Computational experiments

	Java GCJ (s)	**Java JRE (s)**	**C++ (s)**
Eternity II	121.60	33.08	8.35
TSP	115.52	33.45	7.32

As expected, in both problems C++ got the lowest computational times, while the compiled Java version got the highest times. The interpreted version of Java was faster than the compiled one, but slower than C++ version.

This way, OptFrame showed to be a good tool for a fair comparison between heuristic methods for different optimization problems and even with different programming languages.

[3]The Java version of OptFrame is called JOptFrame and it is also available on `http://sourceforge.net/projects/joptframe/`

OptFrame is a free software licensed under LGPLv3. The development version and newer stable version of OptFrame are available at `http://sourceforge.net/projects/optframe/`. It has been successfully applied to model many realistic optimization problems.

Users are invited to visit our homepage and collaborate with the project. Code reuse must be maximized, with clear abstractions based on optimization concepts, but always keeping in mind that the target user should use only simple C++ on his/her code.

6. ACKNOWLEDGMENTS

The authors are grateful to CNPq (CT-INFO and UNIVERSAL), CAPES (PROCAD and PRO-ENG), FAPERJ and FAPEMIG that partially funded this research.

7. REFERENCES

[1] A. Fink and S. Voß, "HotFrame: a heuristic optimization framework," in *Optimization Software Class Libraries*, S. Voß and D. L. Woodruff, Eds. Boston: Kluwer Academic Publishers, 2002, pp. 81–154.

[2] M. Graccho and S. C. S. Porto, "TabOOBuilder: An object-oriented framework for building tabu search applications." in *Proceedings of the Third Metaheuristics International Conference*, Angra dos Reis, Rio de Janeiro, 1999, pp. 247–251.

[3] A. Mendes, P. França, and P. Moscato, "NP-Opt: an optimization framework for np problems," in *Proceedings of the IV SIMPOI/POMS 2001*, Guarujá, São Paulo, 2001, pp. 11–14.

[4] L. D. Gaspero and A. Schaerf, "EasyLocal++: an object-oriented framework for the flexible design of local-search algorithms," *Softw. Pract. Exper.*, vol. 8, no. 33, pp. 733–765, 2003.

[5] S. Cahon, N. Melab, and E.-G. Talbi, "Paradiseo: A framework for the reusable design of parallel and distributed metaheuristics," *Journal of Heuristics*, vol. 10, no. 3, pp. 357–380, 2004.

[6] R. Dorne, P. Mills, and C. Voudouris, "Solving vehicle routing using iOpt," in *Proceedings of MIC 2005 - The 6th Metaheuristics International Conference*, Viena, Áustria, 2005.

[7] J. J. Durillo, A. J. Nebro, F. Luna, B. Dorronsoro, and E. Alba, "jMetal: A java framework for developing multi-objective optimization metaheuristics," Departamento de Lenguajes y Ciencias de la Computación, University of Málaga, E.T.S.I. Informática, Campus de Teatinos, Tech. Rep. ITI-2006-10, 2006.

[8] D. L. Applegate, R. E. Bixby, V. Chvatal, and W. J. Cook, *The Traveling Salesman Problem: A Computational Study*. United Kingdom: Princeton University Press, 2006.

[9] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1995.

[10] C. Ribeiro and M. Resende, "Path-relinking intensification methods for stochastic local search algorithms," AT&T Labs Research, Tech. Rep. NJ 07932, 2010.

[11] C. Blum and A. Roli, *Hybrid Metaheuristics*. Springer, 2008.

[12] P. Hansen and N. Mladenović, "First vs. best improvement: an empirical study," *Discrete Appl. Math.*, vol. 154, no. 5, pp. 802–817, 2006.

[13] F. W. Glover and G. A. Kochenberger, *Handbook of Metaheuristics*. Springer, 2003.

[14] Hansen, Mladenovic, and Perez, "Variable neighborhood search: methods and applications," *4OR: Quarterly journal of the Belgian, French and Italian operations research societies*, vol. 6, pp. 319–360, 2008.

[15] I. M. Coelho, B. N. Coelho, V. N. Coelho, M. N. Haddad, M. J. F. Souza, and L. S. Ochi, "A general variable neighborhood search approach for the resolution of the eternity ii puzzle," in *International Conference on Metaheuristics and Nature Inspired Computing*, Tunisia, Djerba Island, 2010, p. 3.

[16] M. Matsumoto and T. Nishimura, "Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator," *ACM Trans. Model. Comput. Simul.*, vol. 8, pp. 3–30, January 1998.

RAMP: An Overview of Recent Advances and Applications

Dorabela Gamboa * César Rego †

* Escola Superior de Tecnologia e Gestão de Felgueiras, CIICESI, GECAD, Instituto Politécnico do Porto
Apt. 205, 4610-156 Felgueiras, Portugal
dgamboa@estgf.ipp.pt

† School of Business Administration, University of Mississippi
MS 38677, USA
crego@bus.olemiss.edu

ABSTRACT

The Relaxation Adaptive Memory Programming (RAMP) metaheuristic approach has been applied to several complex combinatorial optimization problems, exhibiting an extraordinary performance by producing state-of-the art algorithms. We describe some of these applications and consider modeling techniques and implementation details that proved effective in enhancing RAMP algorithms.

Keywords: RAMP, Scatter Search, Cross-Parametric Relaxation, Adaptive Memory, Metaheuristics

1. INTRODUCTION

In recent years, innovations in metaheuristic search methods have expanded our ability to solve hard problems, and have increased the size of problems that can be considered computationally tractable. Advances have notably come from designs of variable-depth neighborhood constructions [1, 2] and the application of adaptive memory search methods originated by the framework of Tabu Search [3, 4], and from recent developments in the area of evolutionary methods represented by the frameworks of Genetic Algorithms [5], Evolutionary Programming [6] and Scatter Search [7].

Some of the most significant advances derive from a marriage of the adaptive memory Tabu Search approaches with the evolutionary method of Scatter Search (SS). Scatter Search embodies many of the principles of Tabu Search, and the union of these methods is typically implicit in SS applications.

A new advance has occurred with the emergence of Relaxation Adaptive Memory Programming (RAMP), a method that integrates AMP with mathematical relaxation procedures to produce a unified framework for the design of dual and primal-dual metaheuristics that take full advantage of adaptive memory programming [8].

The RAMP metaheuristic has been applied to several complex combinatorial optimization problems, exhibiting an extraordinary performance by producing state-of-the art algorithms. We describe some of these applications and consider modeling techniques and implementation details that proved effective in enhancing RAMP algorithms.

2. RAMP

The Relaxation Adaptive Memory Programming framework is embodied in two approaches, its basic form (Simple RAMP or just RAMP) and its primal-dual extension (PD-RAMP). The RAMP method, at the first level, operates by combining fundamental principles of mathematical relaxation with those of adaptive memory programming, as expressed in tabu search. The extended PD-RAMP method, at the second level, integrates the RAMP approach with other more advanced strategies. We identify specific combinations of such strategies at both levels, based on Lagrangean and surrogate constraint relaxation on the dual side and on scatter search and path relinking on the primal side, in each instance joined with appropriate guidance from adaptive memory processes. The framework invites the use of alternative procedures for both its primal and dual components, including other forms of relaxations and evolutionary approaches such as genetic algorithms and other procedures based on metaphors of nature.

The implementation model of a RAMP algorithm can be seen as an incremental process, starting with one of the simplest forms of the method and successively applying more complex forms, adjusting the design of the algorithm based on the analysis of the results obtained in previous levels of implementation in the quest for attaining the best results possible.

An instance of such an incremental approach may be illustrated by the application of the RAMP method to the Capacitated Minimum Spanning Tree (CMST) [9]. In this application, the development of the RAMP algorithm involved the following incremental steps: (1) the design of a basic surrogate constraints relaxation coupled with a projection method based on a constructive heuristic; (2) the design of an enhanced surrogate relaxation using cutting planes; (3) the development of tabu search procedure used as an improvement method; (4) the implementation of a subgradient-based procedure to appropriately connect primal with dual components of the algorithm; (4) the development of a scatter search solution combination method to create compound memory structures.

Recent applications featuring the design and implementation of effective RAMP algorithms in a variety of settings ranging from facility location to assignment and resource allocation demonstrate the effectiveness of this approach. These problems are classical in combinatorial optimization and arise in numerous applications. The results obtained for these recognizably difficult problems clearly demonstrate the superiority of the RAMP method comparatively to the current state of the art algorithms for the solution of these problems.

3. CONCLUSIONS

In spite of its freshness, the RAMP framework has already shown great potential by obtaining excellent results with every applica-

tion of the method developed so far. In fact, in all these applications, the method revealed impressive effectiveness, frequently attaining optimal solutions for the problems tested, and in many cases, where the optimal solutions are unknown, the method finds solutions with better quality than the previously best known.

4. ACKNOWLEDGEMENTS

The authors would like to acknowledge FCT, FEDER, POCTI, POSI, POCI, POSC, and COMPETE for their support to R&D Projects.

5. REFERENCES

[1] R. K. Ahuja, O. Ergun, J. B. Orlin, and A. P. Punnen, "A survey of very large-scale neighborhood search techniques," *Discrete Applied Mathematics*, vol. 123, pp. 75–102, 2002.

[2] C. Rego and F. Glover, "Ejection chain and filter-and-fan methods in combinatorial optimization," *Annals of Operations Research*, vol. 175, pp. 77–105, 2010.

[3] F. Glover, "Tabu search - Part I," *ORSA Journal on Computing*, vol. 1, pp. 190–206, 1989.

[4] ——, "Tabu search - Part II," *ORSA Journal on Computing*, vol. 2, pp. 4–32, 1990.

[5] C. Reeves, *Modern Heuristic Techniques for Combinatorial Problems*. Blackwell Scientific Publishing, 1993.

[6] D. B. Fogel, "Evolutionary programming: An introduction and some current directions," *Statistics and Computing*, vol. 4, pp. 113–130, 1994.

[7] F. Glover, *Scatter Search and Path Relinking*. McGraw Hill, 1999, pp. 297–316.

[8] C. Rego, *RAMP: A New Metaheuristic Framework for Combinatorial Optimization*. Kluwer Academic Publishers, 2005, pp. 441–460.

[9] C. Rego, F. Mathew, and F. Glover, "Ramp for the capacitated minimum spanning tree problem," *Annals of Operations Research*, vol. 181, pp. 661–681, 2010.

A Polyhedral Study of Mixed 0-1 Sets

Agostinho Agra * Mahdi Doostmohammadi *

* Department of Mathematics and CIDMA
University of Aveiro
{aagra, mahdi}@ua.pt

ABSTRACT

We consider a variant of the well-known single node fixed charge network flow set with constant capacities. This set arises from the relaxation of more general mixed integer sets such as lot-sizing problems with multiple suppliers. We provide a complete polyhedral characterization of the convex hull of the given set.

Keywords: Mixed Integer Set, Polyhedral Description, Valid Inequality, Convex Hull

1. INTRODUCTION

We consider mixed integer sets of the form

$$X = \{(w,z,y) \in \mathbb{R}^n_+ \times \mathbb{B}^n \times \mathbb{B} \mid \sum_{j\in N} w_j \le Dy, \quad (1)$$

$$w_j \le Cz_j,\ j \in N \quad (2)\}$$

where $N = \{1,\dots,n\}$.

These sets are much related with the well-known single node fixed charge network flow set

$$W = \{(w,z) \in \mathbb{R}^n_+ \times \mathbb{B}^n \mid \sum_{j\in N} w_j \le D, w_j \le Cz_j,\ j \in N\}$$

while binary variables z_j are associated with the arcs incident to the node (see Figure 1), indicating whether the arc is open or not, binary variable y in associated with the node itself. These binary variables allow us to model cases where there are fixed costs associated to the use of each arc and node, respectively.

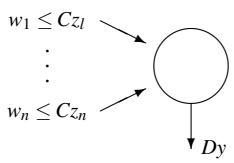


Figure 1: Single node fixed charge set.

Here we investigate the polyhedral description of the convex hull of X, denoted by P. This study is motivated by the interest in tightening more general mixed integer sets, and, in particular, the single-item Lot-sizing with Supplier Selection (LSS) problem. In the LSS problem a set of suppliers is given, and in each time period one needs to decide a subset of suppliers to select and the lot-sizes. Let T be the set of production periods and N be the set of suppliers. We assume that $d_t > 0$ is the demand in period $t \in T$, h_t is unit holding cost, fp_t and p_t represent the production set-up cost and variable production cost in period t, respectively, and c_{jt} and fs_{jt} are variable and fixed sourcing set-up cost for supplier j in period t. D and C are production and supplying capacities. In addition, several types of decision variables are defined. We let x_t be the quantity produced in period t; s_t be the stock level at the end of period $t \in T$; w_{jt} be the quantity sourced from supplier $j \in N$ in period $t \in T$. We define also the binary variables y_t indicating whether there is a setup for production in period t or not, and z_{jt} taking value 1 if and only if supplier j is selected in period t. The LSS problem can be formulated as follows (see [5]):

$$Min \quad \sum_{t\in T} h_t s_t + \sum_{t\in T}\sum_{j\in N}(p_t + c_{jt})w_{jt} + \sum_{t\in T} fp_t y_t + \sum_{t\in T}\sum_{j\in N} fs_{jt} z_{jt}$$

$$s.t. \quad s_{t-1} + x_t = d_t + s_t \quad : \quad \forall t \in T, \quad (3)$$

$$x_t \le Dy_t \quad : \quad \forall t \in T, \quad (4)$$

$$x_t = \sum_{j\in N} w_{jt} \quad : \quad \forall t \in T, \quad (5)$$

$$w_{jt} \le Cz_{jt} \quad : \quad \forall j \in N, \forall t \in T, \quad (6)$$

$$s_0 = s_{|T|} = 0, \quad (7)$$

$$x_t, s_t \ge 0 \quad : \quad \forall t \in T, \quad (8)$$

$$w_{jt} \ge 0 \quad : \quad \forall j \in N, \forall t \in T, \quad (9)$$

$$y_t \in \{0,1\} \quad : \quad \forall t \in T, \quad (10)$$

$$z_{jt} \in \{0,1\} \quad : \quad \forall j \in N, \forall t \in T. \quad (11)$$

For a fixed t, set X arises from (4)-(6), (9)-(11). Valid inequalities for W can be converted into valid inequalities for X.

The polyhedral description of the convex hull of W, denoted by Q is given [4]. In [2] is studied the polyhedral characterization a similar set where lower bounds are imposed on the flow on the arcs. Valid inequalities for SNFC sets with multiple upper and lower bounds also in [3].

We study the polyhedral characterization of P. Although X is very close related to W, and valid inequalities for X can be easily converted into valid inequalities for W and vice-versa, we show that P has, in general, many more facet-defining inequalities than Q. Our main contribution is the full polyhedral description of P.

2. POLYHEDRAL RESULTS

In this section we provide a polyhedral characterization of P and establish the main differences between polyhedra P and Q. We assume $D > C > 0$ and assume that C does not divide D.

We start by an intuitive result.

Proposition 2.1. *P and Q are full dimensional polyhedra.*

It is well-known, see [4], that in addition to inequalities defining W, the following set of facet-defining inequalities is enough to de-

scribe Q.

$$\sum_{j\in S}(w_j - rz_j) \le D - \lceil \frac{D}{C} \rceil r, \;\; S \subseteq N, |S| \ge \lceil \frac{D}{C} \rceil, \qquad (12)$$

where $r = D - \lfloor \frac{D}{C} \rfloor C$.

Polyhedral description of P is somewhat more complex. It is not difficult to verify the following property relating valid inequalities for the two sets.

Proposition 2.2. *The inequality*

$$\sum_{j\in N} \alpha_j w_j + \sum_{j\in N} \beta_j z_j \le \alpha$$

is valid for W, if and only if

$$\sum_{j\in N} \alpha_j w_j + \sum_{j\in N} \beta_j z_j \le \alpha y$$

is valid for X.

One can also check that facet-defining inequalities for Q are converted into facet-defining inequalities for P. However, the converse does not hold in general.

Next we introduce two families of valid inequalities for P.

Proposition 2.3. *Let $D > C > 0$ and assume D is not a multiple of C. The inequality*

$$w_j \le Cy, \;\; j \in N, \qquad (13)$$

is valid for X.

Proof: Validity of (13) follows from (2) and $z_j \le 1$.

□

Proposition 2.4. *Let $D > C > 0$ and assume D is not a multiple of C. Define $r = D - \lfloor \frac{D}{C} \rfloor C$. Let $S_1, S_2 \subseteq N$ such that $S_1 \cap S_2 = \emptyset$ and $0 \le |S_1| < \lceil \frac{D}{C} \rceil$, $\lceil \frac{D}{C} \rceil \le |S_1| + |S_2| \le n$. Then the following inequality is valid for X.*

$$\sum_{j\in S_1} w_j + \sum_{j\in S_2} (w_j - rz_j) \le (D - kr)y \qquad (14)$$

where $k = \lceil \frac{D}{C} \rceil - |S_1|$.

Proof: We prove the validity as follows. If $y = 0$, then constraint (1) implies that $w_j = 0, \forall j \in N$. Since $w_j = 0$, $z_j \ge 0, \forall j \in N$, and $r > 0$, so the inequality (14) is satisfied.
If $y = 1$, then we take a $k = \lceil \frac{D}{C} \rceil - |S_1|$. Inequality (14) can be represented in the following way.

$$\sum_{j\in S_1\cup S_2} w_j \le D + r(\sum_{j\in S_2} z_j - k) \qquad (15)$$

We consider the following two cases.

(i) If $\sum_{j\in S_2} z_j \ge k$, then $r(\sum_{j\in S_2} z_j - k) \ge 0$. So,

$$\sum_{j\in S_1\cup S_2} w_j \le D \le D + r(\sum_{j\in S_2} z_j - k)$$

which shows that (15) is satisfied.

(ii) If $\sum_{j\in S_2} z_j = k - a$ with $a \ge 1$, then we must prove that $\sum_{j\in S_1\cup S_2} w_j \le D - ar$. So by the assumption, definitions of k and r, and the fact that $C > r$,

$$\begin{aligned}
\sum_{j\in S_1\cup S_2} w_j &= \sum_{j\in S_1} w_j + \sum_{j\in S_2} w_j \le C\,|S_1| + \sum_{j\in S_2} Cz_j \\
&= C(|S_1| + \sum_{j\in S_2} z_j) = C(|S_1| + k - a) \\
&= C(\lceil \frac{D}{C} \rceil - a) = C(\lfloor \frac{D}{C} \rfloor + 1 - a) \\
&= C\lfloor \frac{D}{C} \rfloor - C(a-1) \le C\lfloor \frac{D}{C} \rfloor - r(a-1) \\
&= D - r - r(a-1) = D - ar
\end{aligned}$$

Therefore (14) is valid for X.

□

A key point not shown here is that (13) and (14) define facets of P. From Proposition 2.2, valid inequalities for X are valid for W. However, considering for instance (14) with $S_1 \neq \emptyset$, the corresponding valid inequality for W

$$\sum_{j\in S_1} w_j + \sum_{j\in S_2} (w_j - rz_j) \le D - kr,$$

do not define a facet of Q since every point lying in the face defined by the inequality must satisfy $z_j = 1, j \in S_1$.

Example 2.5. *Consider an instance with $n = 4$, $D = 11$, and $C = 4$. Using the software PORTA we obtain 57 facet-defining inequalities for P and 18 facet-defining inequalities for Q. Considering the case with y, we have the following examples of facet-defining inequalities for $k = 1, 2,$ and 3.*

$$\begin{aligned}
&w_1 + w_2 + w_4 - 3z_2 \le 8y \quad : \quad k = 1, \\
&w_1 + w_2 + w_3 - 3z_2 - 3z_3 \le 5y \quad : \quad k = 2, \\
&w_1 + w_2 + w_3 - 3z_1 - 3z_2 - 3z_3 \le 2y \quad : \quad k = 3.
\end{aligned}$$

Note that for $k = 3$, there exist 5 facet-defining inequalities for P and these inequalities appear in Q as a facet-defining inequalities by setting $y = 1$. However for $k = 1$ and $k = 2$ the corresponding inequalities for Q, obtained by setting $y = 1$ are not facet-defining.

Next we establish the main result.

Theorem 2.6. *The defining inequalities of X with the inequalities* (13) *and* (14) *suffice to describe the convex hull of P.*

3. CONCLUSION AND FUTURE RESEARCH

We provide a polyhedral description of a mixed 0-1 set which can be regarded as a variant of the single node fixed charge network flow set where setups are associated to the node and to each arc. We consider the constant capacitated case. Although this set is much related to the well-known set W (where there is not binary variable associated to the node) we have shown that many new facets appear that can not be obtained from facets of the convex hull of W. Some results established here can easily be generalized for the case with different capacities on the arcs. Currently we are following this direction of research as well as investigating the new facet-defining inequalities that might occur for the set with constant lower bounds whose polyhedral description was studied by Constantino [2] and the set with constant lower and upper bounds whose polyhedral description was given by Agra and Constantino[1].

4. REFERENCES

[1] A. Agra and M. Constantino, "Polyhedral description of the integer single node flow set with constant bounds", *Mathematical Programming*, vol. 105, no. 2-3, pp. 345-364, 2006.

[2] M. Constantino, "Lower Bounds in Lot-sizing Models: a Polyhedral Study", *Mathematics of Operations Research*, vol. 23, no. 1, pp. 101-118, 1998.

[3] M. X. Goemans, "Valid Inequalities and Separation for Mixed 0-1 Constraints with Variable Upper Bounds", *Operations Research Letters*, vol. 8, pp. 315-322, 1989.

[4] M. W. Padberg and T. J. Van Roy and L. A. Wolsey, "Valid Linear Inequalities for Fixed Charge Problems", *Operations Research*, vol. 22, no. 4, pp. 842-861, 1985.

[5] Y. Zhao and D. Klabjan, "A Polyhedral Study of Lot-sizing with Supplier Selection", to appear in *Discrete Optimization*.

Multi-Objective Economic Lot-Sizing Models

Wilco van den Heuvel [*] H. Edwin Romeijn [†] Dolores Romero Morales [‡]
Albert P.M. Wagelmans [*]

[*] Econometric Institute, Erasmus University Rotterdam
P.O. Box 1738, 3000 DR Rotterdam, The Netherlands
wvandenheuvel@ese.eur.nl,wagelmans@few.eur.nl

[†] Department of Industrial and Operations Engineering, University of Michigan
1205 Beal Avenue, Ann Arbor, Michigan 48109-2117, USA
romeijn@umich.edu

[‡] Saïd Business School, University of Oxford
Park End Street, Oxford OX1 1HP, United Kingdom
dolores.romero-morales@sbs.ox.ac.uk

ABSTRACT

Nowadays, companies are forced to think about their environmental impact and their levels of pollution. In the production setting, pollution stems from the setup of the machinery, the functioning of the machinery during production as well as from holding inventory. Bearing in mind this environmental awareness, the choice of a production plan can be modeled as a Multi-Objective Economic Lot-Sizing problem, in which we aim at minimizing the total lot-sizing costs including production and inventory holding costs, as well as minimizing the total production and inventory emission costs. Different multi-objective optimization models can be obtained depending on time horizon in which the emissions are minimized. We can minimize the emission costs for the whole planning horizon, yielding a bi-objective model (BOLS), or we can minimize the emission costs in each period of the planning horizon yielding a truly multi-objective optimization model (MOLS). In this talk, we aim at describing Pareto efficient solutions for both (BOLS) and (MOLS). We first show that, in general, this task is *NP*-complete. We then present classes of problem instances for which these Pareto efficient solutions can be found in polynomial time.

Keywords: Lot-sizing, Pollution, Pareto efficient solutions

1. INTRODUCTION

Nowadays, companies are forced to think about their environmental impact and their levels of pollution. In the production setting, pollution stems from the setup of the machinery, the functioning of the machinery during production as well as from holding inventory. Bearing in mind this environmental awareness, the choice of a production plan can be modeled as a Multi-Objective Economic Lot-Sizing problem. This is a generalization of the *Economic Lot-Sizing Problem (ELSP)* in which we aim at minimizing the total lot-sizing costs including production and inventory holding costs, as well as minimizing the total production and inventory emission costs.

Consider a planning horizon of length T. For period t, let f_t be the setup lot-sizing cost, c_t the unit production lot-sizing cost, h_t the unit inventory holding lot-sizing cost, and d_t the demand. Similarly, for period t, let $\hat{f}_t$ be the setup emission cost, $\hat{c}_t$ the unit production emission cost and $\hat{h}_t$ the unit inventory emission holding cost. Let M be a constant such that $M \geq \sum_{t=1}^{T} d_t$.

Let us consider the following biobjective economic lot-sizing model (BOLS):

$$\text{minimize } \Big(\sum_{t=1}^{T} [f_t y_t + c_t x_t + h_t I_t], \sum_{t=1}^{T} [\hat{f}_t y_t + \hat{c}_t x_t + \hat{h}_t I_t] \Big)$$

subject to (BOLS)

$$\begin{aligned}
x_t + I_{t-1} &= d_t + I_t && t = 1,\ldots,T && (1)\\
x_t &\leq M y_t && t = 1,\ldots,T && (2)\\
I_0 &= 0 && && (3)\\
y_t &\in \{0,1\} && t = 1,\ldots,T &&\\
x_t &\geq 0 && t = 1,\ldots,T &&\\
I_t &\geq 0 && t = 1,\ldots,T &&
\end{aligned}$$

where y_t indicates whether a setup has been placed in period t, x_t denotes the quantity produced in period t, and I_t denotes the inventory level at the end of period t. In the following, we will refer to a production period as a period in which production occurs, i.e., $x_t > 0$. The first objective in (BOLS) models the usual *lot-sizing* costs, i.e., the production and inventory holding costs over the whole planning horizon. The second objective models the total emission of pollution across the whole planning horizon. Constraints (1) model the balance between production, storage and demand in period t. Constraints (2) impose that production level is equal to zero if no setup is placed in period t. Constraints (3) impose that the inventory level is equal to zero at the beginning of the planning horizon. The last three constraints define the range in which the variables are defined.

Alternatively, we can define a (truly) multi-objective economic lot-sizing model, where the emission of pollution is minimized in each period of the planning horizon. The model reads as follows:

$$\text{minimize } \Big(\sum_{t=1}^{T} [f_t y_t + c_t x_t + h_t I_t],$$

$$\hat{f}_1 y_1 + \hat{c}_1 x_1 + \hat{h}_1 I_1, \ldots, \hat{f}_T y_T + \hat{c}_T x_T + \hat{h}_T I_T \Big)$$

subject to (MOLS)

$$\begin{array}{rcll} x_t + I_{t-1} & = & d_t + I_t & t = 1,\ldots,T \\ x_t & \leq & My_t & t = 1,\ldots,T \\ I_0 & = & 0 & \\ y_t & \in & \{0,1\} & t = 1,\ldots,T \\ x_t & \geq & 0 & t = 1,\ldots,T \\ I_t & \geq & 0 & t = 1,\ldots,T. \end{array}$$

When the lot-sizing cost function is concave, the classical ELSP is solvable in polynomial time in T, see [12]. More efficient algorithms for special cases have been developed in [1, 4, 11]. In this paper, we aim at describing Pareto efficient solutions for both (BOLS) and (MOLS). In Section 2, we show that, in general, this task is *NP*-complete. Therefore, in Sections 3 and 4, we propose classes of problem instances for which this task can be performed in polynomial time. We conclude the paper with Section 5.

2. PARETO OPTIMAL SOLUTIONS

When more than one objective function is optimized, Pareto efficient solutions are sought. These can be found by minimizing one of the objective functions, for instance, the lot-sizing costs, while constraining the rest of objectives.

Given $\hat{B} \in \mathbb{R}_+$, the following problem defines a Pareto efficient solution for (BOLS):

$$\text{minimize} \sum_{t=1}^{T} [f_t y_t + c_t x_t + h_t I_t]$$

subject to $(\mathscr{P}(\hat{B}))$

$$\begin{array}{rcll} x_t + I_{t-1} & = & d_t + I_t & t = 1,\ldots,T \\ x_t & \leq & My_t & t = 1,\ldots,T \\ I_0 & = & 0 & \\ y_t & \in & \{0,1\} & t = 1,\ldots,T \\ x_t & \geq & 0 & t = 1,\ldots,T \\ I_t & \geq & 0 & t = 1,\ldots,T \\ \sum_{t=1}^{T} [\hat{f}_t y_t + \hat{c}_t x_t + \hat{h}_t I_t] & \leq & \hat{B}. & (4) \end{array}$$

Given $(\hat{b}_t) \in \mathbb{R}_+^T$, the following problem defines a Pareto efficient solution for (MOLS):

$$\text{minimize} \sum_{t=1}^{T} [f_t y_t + c_t x_t + h_t I_t]$$

subject to $(\mathscr{P}((\hat{b}_t)))$

$$\begin{array}{rcll} x_t + I_{t-1} & = & d_t + I_t & t = 1,\ldots,T \\ x_t & \leq & My_t & t = 1,\ldots,T \\ I_0 & = & 0 & \\ y_t & \in & \{0,1\} & t = 1,\ldots,T \\ x_t & \geq & 0 & t = 1,\ldots,T \\ I_t & \geq & 0 & t = 1,\ldots,T \\ \hat{f}_t y_t + \hat{c}_t x_t + \hat{h}_t I_t & \leq & \hat{b}_t & t = 1,\ldots,T. \quad (5) \end{array}$$

Both models, $(\mathscr{P}(\hat{B}))$ and $(\mathscr{P}((\hat{b}_t)))$, can be found in [2]. We may observe that if the emission constraints are not binding, both $(\mathscr{P}(\hat{B}))$ and $(\mathscr{P}((\hat{b}_t)))$ reduce to an ELSP and, therefore, are polynomially solvable. Also, it is not difficult to see that the Capacitated Lot-Sizing problem (CLSP) is a particular case of Problem $(\mathscr{P}((\hat{b}_t)))$. Propositions 1 and 2 show that, in general, both $(\mathscr{P}(\hat{B}))$ and $(\mathscr{P}((\hat{b}_t)))$ are $\mathscr{NP}$-complete.

Proposition 1. *Problem $(\mathscr{P}(\hat{B}))$ is $\mathscr{NP}$-complete.*

Proposition 2. *Problem $(\mathscr{P}((\hat{b}_t)))$ is $\mathscr{NP}$-complete.*

3. POLYNOMIALLY SOLVABLE SCENARIOS FOR $(\mathscr{P}(\hat{B}))$

In the following we discuss several scenarios for which $(\mathscr{P}(\hat{B}))$ can be solved in polynomial time.

Recall that, for a given $\hat{B}$, $(\mathscr{P}(\hat{B}))$ yields a Pareto efficient solution of (BOLS). When possible we also discuss the running time of a procedure that describes the whole efficient frontier, i.e., the running time of solving the parametric problem $(\mathscr{P}(\hat{B}))$, for all $\hat{B} \geq 0$.

3.1. Setup emissions

If $\hat{h}_t = 0$, for all t, and $\hat{f}_t$ and $\hat{c}_t$ are stationary, then $(\mathscr{P}(\hat{B}))$ is polynomially solvable. First note that $\sum_{t=1}^{T} x_t = \sum_{t=1}^{T} d_t$. Therefore, if the production emissions are stationary, then $\sum_{t=1}^{T} \hat{c}_t x_t = \hat{c} \sum_{t=1}^{T} d_t$. Now (4) can be written as

$$\sum_{t=1}^{T} y_t \leq \lfloor \tilde{B} \rfloor,$$

where $\tilde{B} = \frac{1}{\hat{f}}(\hat{B} - \hat{c} \sum_{t=1}^{T} d_t)$. Thus, the problem can be written as an ELSP with a bound on the number of production periods. Let $F_n(t)$ be the optimal cost of the subproblem consisting of periods $1,\ldots,t$ with n production periods. Clearly, we can solve the lot-sizing problem with a bound on the number of production periods if we have at hand the values $F_n(T)$ for $n = 1,\ldots,T$.

The values $F_n(t)$ can be found by the following dynamic programming recursion

$$F_n(t) = \min_{i=n,\ldots,t} \{F_{n-1}(i-1) + C(i,t)\},$$

where $C(i,t)$ is the total lot-sizing cost incurred for satisfying the demand in interval $[i,t]$ by production in period i. Note that there are $n-1$ production periods in the interval $[1,i-1]$ and there is 1 production period in the interval $[i,t]$. This recursion is initialized by $F_0(0) = 0$ and $F_0(t) = \infty$ for $t = 1,\ldots,T$. Clearly, this Dynamic Programming (DP) algorithm runs in $\mathscr{O}(T^3)$ time. A similar recursion can be found in [7]. In [9], it is shown that all values $F_n(t)$ can be found in $\mathscr{O}(T^2)$ time when the lot-sizing costs are such that there are no speculative motives to hold inventory. The same running time is shown in [3] in case of stationary setup costs.

Back to $(\mathscr{P}(\hat{B}))$, its optimal solution value is equal to

$$\min_{n \leq \lfloor \tilde{B} \rfloor} F_n(T),$$

which can be found in $\mathscr{O}(T^3)$ time. (Savings can be achieved by noting that the maximum number of production periods is $\tilde{B}$, yielding an algorithm that runs in $\mathscr{O}(T^2\tilde{B})$ time.) If the lot-sizing costs are such that there are no speculative motives to hold inventory, $(\mathscr{P}(\hat{B}))$ can be solved in $\mathscr{O}(T^2)$ time.

The following proposition shows that if $\hat{f}_t$ are general, $(\mathscr{P}(\hat{B}))$ is $\mathscr{NP}$-complete.

Proposition 3. *If $\hat{c}_t = \hat{h}_t = 0$, for all t, Problem $(\mathscr{P}(\hat{B}))$ is $\mathscr{NP}$-complete.*

For the class of problem instances in this section, the efficient frontier of (BOLS) can de described in polynomial time too, since we only need to solve $(\mathscr{P}(\hat{B}))$ for T possible values to $\hat{B}$, namely $\hat{B} = n\hat{f}$, where $n = 1,\ldots,T$. Thus, the efficient frontier can be found in $\mathscr{O}(T^3)$ time, while for the special cases mentioned above, it can be found in $\mathscr{O}(T^2)$ time.

3.2. Production emissions

If $\hat{f}_t = \hat{h}_t = 0$, for all t, and $\hat{c}_t$ are stationary, it is trivial to see that $(\mathscr{P}(\hat{B}))$ is polynomially solvable. This can be easily seen by noticing that the problem is feasible if

$$\sum_{t=1}^{T} d_t \leq \tilde{B}.$$

where $\tilde{B} = \frac{\hat{B}}{\hat{c}}$. If the problem is feasible, then constraint (4) is redundant and the problem reduces to an ELSP.

If $\hat{c}_t$ are general, the complexity of problem $(\mathscr{P}(\hat{B}))$ is unknown. In this case, constraint (4) reads

$$\sum_{t=1}^{T} \hat{c}_t x_t \leq \hat{B},$$

i.e., it imposes an upper bound on the weighted productions.

For the class of problem instances in this section, the efficient frontier of (BOLS) can clearly be described in polynomial time for this scenario.

3.3. Inventory emissions

Suppose that for the lot-sizing costs we have $f_t = f$ and $c_t = c$, while for the emissions we have $\hat{f}_t = \hat{f}$, $\hat{c}_t = \hat{c}$ and $\hat{h}_t = \alpha h_t$. We will show that in this case $(\mathscr{P}(\hat{B}))$ is solvable in polynomial time by fixing the number of production periods. Note that such a problem instance satisfies the Zero Inventory Order property, i.e., $x_t I_{t-1} = 0$ for all t, because of the non-speculative motives assumption (both in the emission and lot-sizing cost).

Two observations are given before we present the procedure to find the optimal solution. First, for a production plan with n production periods, constraint (4) can be written as

$$\alpha \sum_{t=1}^{T} h_t I_t \leq \hat{B} - \hat{f}n - \hat{c}\sum_{t=1}^{T} d_t. \tag{6}$$

Second, because both the setup and the unit production lot-sizing costs are stationary, the objective function of $(\mathscr{P}(\hat{B}))$ boils down to

$$\sum_{t=1}^{T}(f_t y_t + c_t x_t + h_t I_t) = fn + c\sum_{t=1}^{T} d_t + \sum_{t=1}^{T} h_t I_t.$$

Thus, when the number of production periods is fixed, minimizing the total lot-sizing costs is equivalent to minimizing the total inventory cost. Moreover, the objective function also minimizes the left hand side of (6).

The following procedure solves the problem to optimality. Given $n = 1,\ldots,T$, solve the ELSP with n production periods, this can be done in polynomial time, as already shown in Section 3.1. If the inventory levels of the optimal solution satisfy (6), this solution is kept. After evaluating all possible values of n, we will have at most T solutions, from which we choose the solution having the lowest lot-sizing costs.

Notice that if $\hat{h}_t$ are general, the complexity of problem $(\mathscr{P}(\hat{B}))$ is unknown. In this case, constraint (4) can be rewritten as

$$\sum_{t=1}^{T} \bar{c}_t x_t \leq \hat{B} - \sum_{t=1}^{T} \hat{h}_t \sum_{\tau=1}^{t} d_t,$$

where $\bar{c}_t = \sum_{\tau=t}^{T} \hat{h}_t$. Therefore, this reduces to a problem of the form given in Section 3.2, and thus its complexity is unknown.

Again, for the class of problem instances in this section, we can describe the whole efficient frontier in polynomial time. From above, it is clear that the only possible Pareto efficient solutions will be the ones returned by the ELSP with n production periods, $n = 1,\ldots,T$. Also, it is clear that the total inventory levels of these solutions will become the breakpoints of $\tilde{B}$ in the Pareto efficient frontier.

4. POLYNOMIALLY SOLVABLE SCENARIOS FOR $(\mathscr{P}((\hat{b}_T)))$

In the following, we discuss several scenarios for which $(\mathscr{P}((\hat{b}_t)))$ can be solved in polynomial time.

4.1. Setup emissions

In this section, we show that $(\mathscr{P}((\hat{b}_t)))$ is polynomially solvable if $\hat{c}_t = \hat{h}_t = 0$. In this case, constraint (5) implies $y_t = 0$ if $\hat{f}_t > \hat{b}_t$, and otherwise it is redundant. This can be easily incorporated into the dynamic programming approach that solves the ELSP in polynomial time, without increasing the running time, and therefore remaining polynomial.

4.2. Production emissions

In this section, we show that $(\mathscr{P}((\hat{b}_t)))$ is polynomially solvable if $\hat{f}_t = \hat{h}_t = 0$. In this case, constraint (5) can be written as a constraint on x_t. The new capacity constraints are stationary and therefore the problem can be solved in polynomial time, [5] and [8].

4.3. Inventory emissions

In this section, we show that $(\mathscr{P}((\hat{b}_t)))$ is polynomially solvable if $\hat{f}_t = \hat{c}_t = 0$. In this case, constraint (5) can be written as a constraint on I_t. This problem was shown to be polynomially solvable in [10].

4.4. Setup, production and inventory emissions

In this section, we show that $(\mathscr{P}((\hat{b}_t)))$ is polynomially solvable under the following assumptions. With respect to the lot-sizing costs, we assume that the setup costs are non-increasing and there are no speculative motives to hold inventory. With respect to the emissions, we assume that all parameters are stationary.

Definition 4. *We will say that period t is a tight period if*

$$\hat{f}_t y_t + \hat{c}_t x_t + \hat{h}_t I_t = \hat{b}_t.$$

As usual in the literature, we will refer to a regeneration period as a period in which the inventory level at the end of the period is equal to zero, i.e., $I_t = 0$. We will refer to a subplan as the subproblem defined by two consecutive regeneration points. Without loss of optimality, we can assume that the inventory levels within a subplan must all be positive. We will decompose the problem

into subplans using the regeneration periods, and define a straightforward Dynamic Programming algorithm to solve ($\mathscr{P}((\hat{b}_t))$). In order to show that the problem is polynomially solvable we need to show that the costs of a subplan can be calculated in polynomial time. Let us therefore focus on a given subplan, and its optimal costs.

Proposition 5. *There is at most one non-tight production period in a subplan.*

Proposition 6. *Without loss of optimality, the only possible non-tight production period in a subplan is the first period.*

Proposition 7. *There exists an optimal solution satisfying $I_{t-1} < d_t$ for any production period t.*

Proposition 8. *Consider a subplan $[u,v]$ and a period t ($u < t \leq v$) with outgoing inventory I_t and satisfying the properties:*

- $\bar{x}_t \stackrel{\text{def}}{=} (\hat{b} - \hat{f} - \hat{h}I_t)/\hat{c} > 0$,
- $\bar{I}_{t-1} \stackrel{\text{def}}{=} I_t - \bar{x}_t + d_t > 0$.

Then period t is a tight production period in the subplan with production quantity $\bar{x}_t$.

We can now use Proposition 8 to construct an optimal solution to any non-degenerate subplan $[u,v]$ (i.e., it does not decompose into multiple subplans) in a backward way. Assume that we arrive in some period $t > u$, that I_t is known (note that $I_v = 0$ in the initialization of the procedure) and we want to determine x_t and I_{t-1}. We consider the following cases:

- $\bar{x}_t \leq 0$: The subplan is infeasible, since constraint (5) is violated for period t or some period before t. Note that $\bar{x}_t$ is the maximum production quantity in period t without violating the emission constraint. It follows from the proof of Proposition 8 that any feasible production quantity in period s ($s < t$) is at most equal to $\bar{x}_t$. In other words, any period with a positive production amount before period t will violate the emission constraint.
- $\bar{x}_t > 0$ and $\bar{I}_{t-1} \leq 0$: In this case period t cannot be a tight production period, since production would be too much. Therefore, we set $x_t = 0$ and $I_{t-1} = I_t + d_t$. Note that the subplan would be degenerate in case $\bar{I}_{t-1} = 0$.
- $\bar{x}_t > 0$ and $\bar{I}_{t-1} > 0$: By Proposition 8 period t is tight. Hence, we set $x_t = \bar{x}_t$ and $I_{t-1} = \bar{I}_{t-1}$.

This procedure is applied until we arrive at period u. If $0 < d_u + I_{u+1} \leq \bar{x}_u$, then subplan $[u,v]$ is feasible and non-degenerate with a production quantity equal to $x_u = d_u + I_{u+1}$.

For given periods u and v, the cost of subplan $[u,v]$ can be determined in linear time. Hence, a straightforward implementation would lead to an $\mathscr{O}(T^3)$ time algorithm. However, note that when determining subplan $[1,v]$, we also find subplans $[u,v]$ for $u = 1,\ldots,v$. This means that all subplans can be found in $\mathscr{O}(T^2)$ time, and so the optimal solution to the problem.

5. CONCLUSIONS

In this paper, we have studied lot-sizing models incorporating pollution emissions, and modeled them as multi-objective problems. We have shown that finding Pareto efficient solutions to this problems is, in general, an *NP*-complete task. We have presented classes of problem instances for which these Pareto efficient solutions can be found in polynomial time.

6. REFERENCES

[1] A. Aggarwal and J.K. Park. Improved algorithms for economic lot-size problems. *Operations Research*, 41(3):549–571, 1993.

[2] S. Benjaafar, Y. Li, and M. Daskin. Carbon footprint and the management of supply chains: Insights from simple models. Research report, 2010.

[3] A. Federgruen and J. Meissner. Competition under time-varying demands and dynamic lot sizing costs. *Naval Research Logistics*, 56(1):57–73, 2009.

[4] A. Federgruen and M. Tzur. A simple forward algorithm to solve general dynamic lot sizing models with n periods in $O(n \log n)$ or $O(n)$. *Management Science*, 37:909–925, 1991.

[5] M. Florian and M. Klein. Deterministic production planning with concave costs and capacity constraints. *Management Science*, 18:12–20, 1971.

[6] M.R. Garey and D.S. Johnson. *Computers and intractability: a guide to the theory of NP-completeness*. W.H. Freeman and company, New York, 1979.

[7] S.M. Gilbert. Coordination of pricing and multi-period production for constant priced goods. *European Journal of Operational Research*, 114(2):330–337, 1999.

[8] C.P.M. van Hoesel and A.P.M. Wagelmans. An O(T^3) algorithm for the economic lot-sizing problem with constant capacities. *Management Science*, 42(1):142–150, 1996.

[9] C.P.M. van Hoesel and A.P.M. Wagelmans. Parametric analysis of setup cost in the economic lot-sizing model without speculative motives. *International Journal of Production Economics*, 66:13–22, 2000.

[10] S.F. Love. Bounded production and inventory models with piecewise concave costs. *Management Science*, 20(3):313–318, 1973.

[11] A. Wagelmans, S. van Hoesel, and A. Kolen. Economic lot sizing: An O($n \log n$) algorithm that runs in linear time in the Wagner-Whitin case. *Operations Research*, 40(1):S145–S156, 1992.

[12] H.M. Wagner. A postscript to dynamic problems of the theory of the firm. *Naval Research Logistics Quarterly*, 7:7–12, 1960.

An Optimization Model for the Traveling Salesman Problem with Three-dimensional Loading Constraints

Leonardo Junqueira * José Fernando Oliveira † Maria Antónia Carravilla †
Reinaldo Morabito *

* Departamento de Engenharia de Produção, Universidade Federal de São Carlos
Rodovia Washington Luís, km 235 - SP-310, 13565-905, São Carlos - São Paulo - Brasil
leo_junqueira@yahoo.com, morabito@ufscar.br

† Faculdade de Engenharia, Universidade do Porto
Rua Dr. Roberto Frias s/n, 4200-465, Porto, Portugal
{jfo, mac}@fe.up.pt

ABSTRACT

In this paper, we present a mixed integer linear programming model for the traveling salesman problem that considers three-dimensional loading constraints. Computational tests with the proposed model were performed with randomly generated instances using an optimization solver embedded into a modeling language. The results validate the model and show that it is able to handle only problems of a moderate size. However, the model can be useful to motivate future research to solve larger problems, especially when this problem appears as a sub-problem of another problem, as well as modeling the more general vehicle routing problem with three-dimensional loading constraints.

Keywords: Traveling salesman problem, Three-dimensional loading, Combinatorial optimization, Mathematical modeling

1. PROBLEM DESCRIPTION

The vehicle routing literature has been recently merged with the container loading literature to treat cases where the goods required by the customers are wrapped up in discrete items, such as boxes. This effort arises from the attempt to avoid expressing the demands of the customers simply as their weights or volumes. In other words, if the demand constraints are seen in one-dimensional point of view, it is assumed that each demand fills one certain section of the vehicle or that the cargo shapes up smoothly according to the vehicle shape. However, when dealing with rigid discrete items, their geometry may lead to losses of space or even to infeasible solutions if the vehicle has not enough capacity. If other practical constraints are also considered ([1]), the coupling of the routing and loading structures becomes even more complex. Constraints such as vertical and horizontal stability of the cargo, load bearing strength and fragility of the cargo, grouping or separation of items inside a container, multi-drop situations, complete shipment of certain item groups, container weight limit, weight distribution within a container, among others, are common in the container loading literature and can also be embedded into vehicle routing problems.

One of these combined problems, the 3L-CVRP (e.g., [2], [3], [4]), considers a fleet of identical vehicles that must run minimum cost routes to deliver boxes to a set of customers, departing from and returning to a depot. Besides the non-overlap of the three-dimensional boxes, the constraints that have been usually considered are the vertical stability of the cargo, the load bearing strength of the boxes and the multi-dropping of the boxes. The 2L-CVRP (e.g., [5], [6], [7]) is a particular case of the above problem where the boxes are too heavy for being stacked and only the floor of the vehicle is considered for the boxes' placement. The approaches used to solve these problems have been mainly heuristic.

In this paper, we address another variant of these combined problems, named the 3L-TSP. In this problem, a set of customers make requests of goods, that are packed into boxes, and the objective is to find a minimum cost delivery route for a single vehicle that, departing from a depot, visits all customers only once and returns to the depot, while considering some three-dimensional loading constraints. Apart the constraints that ensure that the boxes do not overlap each other, the vertical stability of the cargo, the load bearing strength of the boxes (including fragility), and the multi-dropping of the boxes are also taken into account. It is assumed that the boxes and the vehicle are of rectangular shape, and that the cargo completely fits inside the vehicle. We present a mixed integer linear programming model for the problem, aiming to show the impact of the loading constraints. We are not aware of other papers that have presented mathematical formulations for the 3L-TSP and which explicitly deal with such constraints.

2. THREE-DIMENSIONAL LOADING CONSTRAINTS

In a recent study, [8] and [9] presented mathematical formulations for the container loading problem with cargo stability, load bearing strength and multi-drop constraints. Cargo stability refers to the support of the bottom faces of boxes, in the case of vertical stability (i.e., the boxes must have their bottom faces supported by other box top faces or the container floor), and the support of the lateral faces of boxes, in the case of horizontal stability. Load bearing strength refers to the maximum number of boxes that can be stacked one above each other, or more generally, to the maximum pressure that can be applied over the top face of a box, so as to avoid damaging the box. We note that fragility is a particular case of load bearing where boxes cannot be placed above a fragile box, since its top face does not bear any kind of pressure. Multi-drop constraints refer to cases where boxes that are delivered to the same customer (destination) must be placed close to each other in the vehicle, and the loading pattern must take into account the delivery route of the vehicle and the sequence in which the boxes are unloaded. The practical importance of incorporating these constraints to the problem is to avoid loading patterns where boxes are "floating in mid-air" inside the vehicle, where products are damaged due to deformation of the boxes that contain them, or where an unnecessary additional handling is incurred when each drop-off point of the route is reached. In the present study, we

have extended these ideas in the context of the traveling salesman problem.

3. CONCLUSIONS

Computational tests with the proposed model were performed with randomly generated instances using an optimization solver embedded into a modeling language. The results validate the model and show that it is able to handle only problems of a moderate size. However, the model can be useful to motivate future research to solve larger problems, especially when the 3L-TSP appears as a sub-problem of another problem, as well as modeling the more general vehicle routing problem with three-dimensional loading constraints.

4. ACKNOWLEDGEMENTS

This research was partially supported by FAPESP (Grant 09/07423-9) and CAPES (Grant BEX 3187/10-1).

5. REFERENCES

[1] E. E. Bischoff and M. S. W. Ratcliff, "Issues in the development of approaches to container loading," *Omega*, vol. 23, no. 4, pp. 377–390, 1995.

[2] M. Gendreau, M. Iori, G. Laporte, and S. Martello, "A tabu search algorithm for a routing and container loading problem," *Transportation Science*, vol. 40, no. 3, pp. 342–350, 2006.

[3] C. D. Tarantilis, E. E. Zachariadis, and C. T. Kiranoudis, "A hybrid metaheuristic algorithm for the integrated vehicle routing and three-dimensional container-loading problem," *IEEE Transactions on Intelligent Transportation Systems*, vol. 10, no. 2, pp. 255–271, 2009.

[4] G. Fuellerer, K. F. Doerner, H. F. Hartl, and M. Iori, "Metaheuristics for vehicle routing problems with three-dimensional loading constraints," *European Journal of Operational Research*, vol. 201, no. 3, pp. 751–759, 2010.

[5] M. Gendreau, M. Iori, G. Laporte, and S. Martello, "A tabu search heuristic for the vehicle routing problem with two-dimensional loading constraints," *Networks*, vol. 51, no. 1, pp. 4–18, 2008.

[6] G. Fuellerer, K. F. Doerner, R. F. Hartl, and M. Iori, "Ant colony optimization for the two-dimensional loading vehicle routing problem," *Computers & Operations Research*, vol. 36, no. 3, pp. 655–673, 2009.

[7] E. E. Zachariadis, C. D. Tarantilis, and C. T. Kiranoudis, "A guided tabu search for the vehicle routing problem with two-dimensional loading constraints," *European Journal of Operational Research*, vol. 195, no. 3, pp. 729–743, 2009.

[8] L. Junqueira, R. Morabito, and D. S. Yamashita, "Three-dimensional container loading models with cargo stability and load bearing constraints," *(to appear in Computers & Operations Research, doi:10.1016/j.cor.2010.07.017).*

[9] ——, "Mip-based approaches for the container loading problem with multi-drop constraints," *(submitted to Annals of Operations Research).*

Rect–TOPOS: A constructive heuristic for the rectilinear packing area minimization problem

Marisa Oliveira [*] Eduarda Pinto Ferreira [*] [†] A. Miguel Gomes [‡]

[*] ISEP – Instituto Superior de Engenharia do Porto
Dr. António Bernardino de Almeida, 431 4200-072 Porto Portugal
{mjo, epf}@isep.ipp.pt

[†] GECAD – Knowledge Engineering and Decision Support Research Center
Dr. António Bernardino de Almeida, 431 4200-072 Porto Portugal

[‡] INESC Porto, Faculdade de Engenharia, Universidade do Porto
Rua Dr. Roberto Frias, s/n 4200-465 Porto Portugal
agomes@fe.up.pt

ABSTRACT

In this paper we propose a constructive heuristic, Rect–TOPOS, to solve the problem of minimizing the enclosing rectangular area that contains, without overlapping, a set of rectilinear pieces (e.g., L and T shaped pieces). This is a NP-hard combinatorial optimization problem, which belongs to the class of cutting and packing problems. To evaluate the Rect–TOPOS heuristic computational tests were performed to validate it for the presented problem. In these tests, instances with different characteristics were used, namely the total number of pieces, and the shaped diversity of the pieces. The results show that this is a heuristic that can quickly and easily to deal with all the rectilinear shaped pieces.

Keywords: Combinatorial optimization, Cutting and packing, Constructive heuristic, Area minimization

1. INTRODUCTION

In the rectilinear packing area minimization problem (RPAMP) one wishes to pack a set of rectilinear shaped pieces (pieces with 90 or 270 interior angles) while minimizing the area of the enclosing rectangle without overlapped pieces (Figure 1). This problem arises in many industrial applications such as VLSI design, facility layout problems, newspaper layout, etc. It is NP-hard combinatorial optimization problem [1] and belongs to the class of cutting and packing problems (C&P), which are combinatorial problems with a strong geometric component. Approaches to solve C&P problems can be based on any of the usual techniques available for solving general combinatorial optimization problems like: mixed integer programming, heuristics, metaheuristics, etc. Given the combinatorial nature of these problems, the exact techniques are not able to deal effectively with instances of large dimension and it becomes necessary

To solve the RPAMP we propose a variant of the constructive heuristic TOPOS. The main differences between the proposed variant, Rect–TOPOS, and TOPOS come from the shapes of the pieces, rectilinear shapes instead of irregular shapes, and the objective function, area minimization instead of layout length minimization. Additionally, the criteria used to select the next piece to place, its orientation and the best placement point needed to be adapted.

This paper is structured as follows: section 2 presents a detailed description of the RPAMP; in section 3, the constructive heuristic proposed, Rect- TOPOS, is presented; in section 4, computational results are shown and, finally, in Section 5 some concluding remarks are presented.

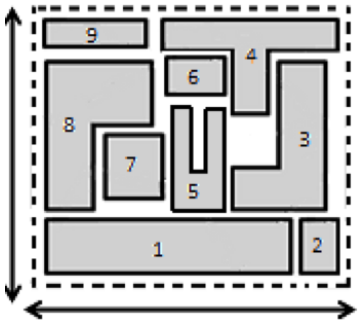


Figure 1: Rectilinear Packing Area Minimization Problem.

2. RECTILINEAR PACKING AREA MINIMIZATION PROBLEM

The objective of the RPAMP is to pack, without overlapping, a set of given rectilinear shaped pieces while minimizing the area of the enclosing rectangle. The dimensions of the pieces are fixed and they must be placed orthogonally (i.e., with sides parallel to the horizontal and vertical axes), though a 90^0, 180^0 or 270^0 rotation of the pieces are allowed. This is a two-dimensional problem and, according to the typology of C&P problems proposed in [2], is classified as an open dimension problem (ODP) since the dimensions of the enclosing rectangle are unknown.

The RPAMP arises in many real word applications such as the placement of modules in Very Large Scale Integration (VLSI) circuits, in the designing of facility, newspaper layouts, etc. For example, in VLSI circuits rectilinear shaped pieces appeared to facilitate the usage of circuit area and improve the connectivity between the pieces, increasing the circuit performance. Early works that appeared in the literature to solve area minimization problems only dealt with rectangles and the main concern was to find efficient data structures to represent layouts. These representations encode solutions as sequences, graphs or trees. Over time, new representations appeared, justified by improvements in the efficiency of solution evaluation, the type of encoding schemes, the amount

of redundancy that exists in the encoding and the total number of configurations. An early work by Wong et *al.* [3] proposed an algorithm for slicing layouts[1] using a tree structure. One important breakthrough is the introduction of the Sequence Pair (SP), by Murata *et al.* [1], for representing non-slicing layouts. This representation is based on a pair of sequences that specifies the relative positions of the rectangles. Many other representations have emerged after the sequence pair. The existing representations for the rectangle packing have been adapted to enable its applicability to problems with rectilinear shaped pieces.

Unlike what happens in most approaches in the literature to the RPAMP the proposed approach does not deal with representations of the layout but it works directly on the layout. The next section provides a description of the proposed heuristic to solve this problem.

3. RECT–TOPOS

To solve the RPAMP we propose a variant of the TOPOS algorithm [4] which was originally developed to solve problems with irregular shapes[2]. The main idea behind it is successively adding a new piece to a partial solution. In the TOPOS algorithm two different levels are used: a first one to choose the best placement point for each piece to place in each admissible orientation (nesting strategies) and a second one to choose, from all the possible candidates from the previous level, the best overall placement (layout evaluation). Three nesting strategies which aim to evaluate the best fit of two irregular shapes (partial solution and the piece chosen) with fixed orientations have been used: minimizing the area of the enclosure of the two pieces, minimizing the length of the enclosure of the two pieces and maximizing the overlap between the rectangular enclosures of the two pieces. To evaluate and compare different layouts three different criteria have been used: the difference between the area of the rectangular enclosure of the partial solution and the area of all pieces already placed (waste), the overlap between the rectangular enclosure of the piece under evaluation and the rectangular enclosure of each piece already placed and, finally, the euclidean distance between the centre of the rectangular enclosure of the piece under evaluation and the centre of the rectangular enclosure of the partial solution.

The overall objective is to minimize the layout length since in these problems the layout width is fixed.

In our variant, Rect–TOPOS, we follow the same general idea, successively adding a new piece to a partial solution while minimizing the enclosing rectangular area. We choose to use a single level to select the next piece to place, its orientation and the best placement point simultaneously. The existence of a single level allows choosing the best piece to place between all possibilities unlike what happens when there are two levels, in which there is an initial selection of the placement point for each piece to place. We used the waste and distance evaluation criteria, taken directly from the criteria used in the second level of the TOPOS, and introduced a new criterion, the perimeter minimization. This new criterion tries to minimize the perimeter between the piece under evaluation and the current partial solution.

The third criterion used in TOPOS, overlap maximization, was removed since it is not appropriate for situations where there are a large number of rectangles to place. In these situations, the enclosing rectangle of a rectangle is the rectangle itself, it makes no sense trying to maximize the overlap of two rectangles because pieces are not allowed to overlap.

[1] A layout is said to be slicing if it can be obtained by successive horizontal and vertical cuts, from one side to another, which divide it into two rectangles.

[2] An irregular shape is a polygon with arbitrary angles.

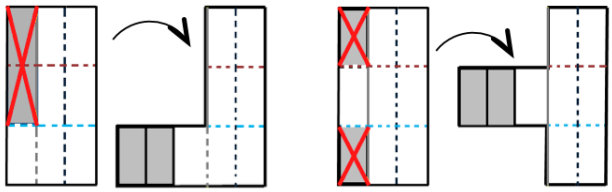


Figure 2: Construction of L and T-shaped pieces from rectangles.

As in TOPOS, the iterative process needs to have an initial non-empty partial solution, so we used another criteria to select the first piece of the partial solution. For this selection we chose to use 3 criteria that favor the selection of the larger pieces: piece with larger area; piece with larger perimeter or piece with larger width.

4. COMPUTATIONAL RESULTS

This section presents the computational results with the heuristic Rect–TOPOS. The tests were performed on a Linux workstation equipped with a Intel XEON Dual Core 5160, 3GHz. Although the workstation has two CPUs, only one thread was used in the tests. The test instances used have different characteristics, particularly in the total number of pieces, number of pieces with different shapes (number of types of pieces) and in the shape of the pieces (rectangular and other shapes with rectangular components). To evaluate the heuristic Rect–TOPOS we used the following four sets of instances:

- instances of the reference set MCNC (`http://vlsicad.eecs.umich.edu/BK/MCNCbench/HARD/`), which is a benchmark set with origins in the project of VLSI circuits, in which all the pieces have a rectangular shape and where the total number of pieces to place does not exceed 50 (APTE, XEROX, HP, AMI33, AMI49);
- instances also composed only by rectangles, however, differ from the previous one by having higher number of pieces, from 100 to 500 (`http://www.simplex.tu-tokyo.ac.jp/imahori/packing/`) (RP100, RP200, PCB146, PCB500);
- instances taken from [5] (NAKATAKE1, NAKATAKE2), [6] (LIN) and [7](AMI49L, AMI49LT) containing a mix of pieces that are rectangles, L-shaped and/or T-shaped pieces and other pieces with rectangular components (U, +, H, etc.);
- instances AMI33LTa and AMI49LTa were generated from instances AMI33 e AMI49 from the MCNC reference set. The rule used to obtain these two intances was to change approximately 10% of the total number of rectangles in L and/or T pieces. Each of the new L or T shaped have integer dimensions and have an area similar to the area of the original rectangle accordingly to the procedure shown in Figura 2.

The instances chosen to test and evaluate the heuristic Rect–TOPOS have very different characteristics, namely in what concerns the total number of pieces, the number of different pieces types, the shape of the pieces (rectangular, L-shaped, T-shaped, etc.). This characteristics are shown in Table 1.

Table 2 summarizes the computational tests performed to test and evaluate the heuristic Rect–TOPOS. We tested the three criteria for choosing the next piece to place, its orientation and placement point previously presented (WASTE, DISTANCE and PERIMETER), and, for each one of them, we considered the three possibilities

	# Pieces		# Rectilinear pieces	
Instance	Total	# Types	# Rect.	# Others
APTE	9	3	9	—
XEROX	10	10	10	—
HP	11	6	11	—
AMI33	33	31	33	—
AMI49	49	46	49	—
RP100	100	99	100	—
PCB146	146	22	200	—
RP200	200	99	146	—
PCB500	500	417	500	—
AMI49L	28	28	7	21
AMI49LT	27	27	6	21
NAKATAKE1	40	35	30	10
NAKATAKE2	19	19	5	14
LIN	29	21	22	7
AMI33LTa	33	31	30	3
AMI49LTa	33	46	41	5

Table 1: Characteristics of the used instances.

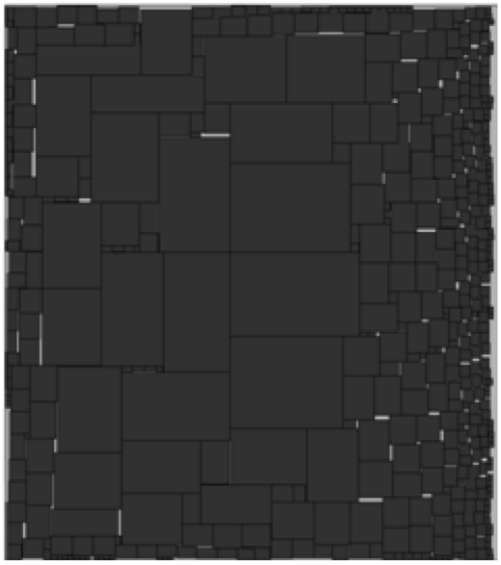


Figure 3: Layout obtained for PCB500 instance.

to choose the piece to start the partial solution (AREA, PERIMETER and DISTANCE). The values shown in the table are the area usage, measured as the ratio between the sum of the area of the placed pieces and the area of the enclosing rectangle obtained. The bold values are the best result for each instance. Table 2 also presents, for each instance, the average computational time, measured in seconds. Note that, for each instance, the computational times does not show great variability. Finally we present also for each instance, the best result found in literature, their area usage, computational time and the publication reference.

From Table 2 we can see that the best results were obtained when using for choosing the next piece to place and the placement point the perimeter criterion, except for instances APTE and XEROX. These two instances are very sensitive to the choice of the first piece to place as they have a small number of pieces to place, 9 and 10 respectively. Regarding the choice of the first piece, the results show balance between the three criteria. When comparing the results obtained with the best published results one should take into account that the Rect–TOPOS is only a constructive heuristic, while the best published results were obtained with approaches based on local search and tree search algorithms. Thus, as expected, the results obtained with the Rect–TOPOS fall short of the published ones, but in return the computational times are much lower. We note that for the PCB500 instance the result obtained by Rect–TOPOS was better than the best result found in the literature [10]. Table 2 also allows to show the great impact that the number of types of pieces have in the Rect–TOPOS heuristic performance. For example, although the PCB146 instance have more 46 pieces in total than the RP100 instance its running time is about 10 times lower because it has only 22 different pieces types while the instance RP100 has 99 different types.

Figure 3 shows the layout obtained for the PCB500 instance.

5. CONCLUSIONS

In this article we presented a constructive heuristic, Rect–TOPOS, to the Rectilinear Packing Area Minimization Problem. Rect–TOPOS is a fast heuristic which is able to easily handle rectilinear shaped pieces. This heuristic uses several criteria to choose the next piece to place, its orientation and the placement point. The quality of solutions proved to be quite satisfactory because it is a simple heuristic with reduced run times. These features suggest, as future developments, the incorporation of Rect–TOPOS heuristic in an approach based on local procedure. In this approach could, at the expense of increased run time, improving the already good results achieved by Rect–TOPOS in situations where this was necessary.

6. ACKNOWLWDGEMENTS

Partially supported by Fundação para a Ciência e Tecnologia (FCT) Project PTDC/EME-GIN/105163/2008 - EaGLeNest, through the "Programa Operacional Temático Factores de Competitividade (COMPETE)" of the "Quadro Comunitário de Apoio III", partially funded by FEDER.

7. REFERENCES

[1] H. Murata, K. Fujiyoshi, S. Nakatake, and Y. Kajitani, "Rectangle-packing-based module placement," in *Proceedings of the 1995 IEEE/ACM international conference on Computer-aided design*, ser. ICCAD '95. Washington, DC, USA: IEEE Computer Society, 1995, pp. 472–479.

[2] G. Wäscher, H. Haußner, and H. Schumann, "An improved typology of cutting and packing problems," *European Journal of Operational Research*, vol. 183, no. 3, pp. 1109–1130, December 2007.

[3] D. F. Wong and C. L. Liu, "A new algorithm for floorplan design," in *Proceedings of the 23rd ACM/IEEE Design Automation Conference*, ser. DAC '86. Piscataway, NJ, USA: IEEE Press, 1986, pp. 101–107.

[4] J. F. Oliveira, A. M. Gomes, and J. S. Ferreira, "TOPOS: A new constructive algorithm for nesting problems," *OR Spectrum*, vol. 22, pp. 263–284, 2000.

[5] S. Nakatake, K. Fujiyoshi, H. Murata, and Y. Kajitani, "Module placement on BSG-structure and ic layout applications," in *Proceedings of the 1996 IEEE/ACM international conference on Computer-aided design*, ser. ICCAD '96. Washington, DC, USA: IEEE Computer Society, 1996, pp. 484–491.

[6] J.-M. Lin, H.-L. Chen, and Y.-W. Chang, "Arbitrarily shaped rectilinear module placement using the transitive closure graph representation." *IEEE Trans. VLSI Syst.*, pp. 886–901, 2002.

	Waste			Distance			Perimeter			Average	Best known result		
Instance	Area	Perim.	Width	Area	Perim.	Width	Area	Perim.	Width	Time (s)	(%)	(s)	
APTE	**0.917**	**0.917**	**0.917**	0.893	0.893	0.893	0.894	0.894	0.894	0.01	0.992	2.38	[8]
XEROX	0.801	0.801	0.801	**0.804**	**0.804**	**0.804**	0.788	0.788	0.788	0.09	0.977	9812	[8]
HP	0.848	0.848	0.695	0.834	0.834	0.695	0.924	0.924	**0.936**	0.03	0.987	891	[8]
AMI33	0.813	0.813	0.875	0.712	0.712	0.745	0.832	0.832	**0.863**	0.84	0.986	2.01	[9]
AMI49	0.807	0.807	0.807	0.792	0.792	0.792	**0.843**	**0.843**	**0.843**	1.97	0.983	56.61	[9]
RP100	0.819	0.819	0.857	0.721	0.721	0.773	**0.924**	**0.924**	0.905	9.35	0.968	200	[10]
PCB146	0.622	0.622	0.622	0.786	0.786	0.786	**0.881**	**0.881**	**0.881**	0.95	0.977	300	[10]
RP200	0.876	0.876	0.878	0.746	0.746	0.754	**0.929**	**0.929**	0.913	13.2	0.963	400	[10]
PCB500	0.865	0.865	0.865	0.781	0.781	0.781	**0.967**	**0.967**	**0.967**	221.0	0.963	1000	[10]
AMI49L	0.625	0.625	0.667	0.761	0.761	0.761	**0.829**	**0.829**	0.792	1.11	0.956	2728	[11]
AMI49LT	0.731	0.731	0.663	0.787	0.787	0.753	0.793	0.793	**0.823**	1.08	0.951	2843	[11]
NAKATAKE1	0.825	0.825	0.763	0.807	0.807	0.784	0.852	0.852	**0.867**	1.35	0.969	10.24	[11]
NAKATAKE2	0.590	0.619	0.590	0.669	0.614	0.669	0.731	**0.770**	0.731	2.14	0.802	93.32	[11]
LIN	0.789	0.740	0.780	0.780	0.780	0.740	0.840	0.840	**0.910**	0.82	1.000	2.72	[11]
AMI33LTa	0.764	0.764	0.711	0.752	0.752	0.731	0.832	0.832	**0.844**	1.05	—	—	—
AMI49LTa	0.875	0.875	0.875	0.761	0.761	0.761	**0.881**	**0.881**	**0.881**	2.41	—	—	—

Table 2: Comparison of results obtained by the Rect–TOPOS vs literature results.

[7] J. Xu, P.-n. Guo, and C.-K. Cheng, "Rectilinear block placement using sequence-pair," in *Proceedings of the 1998 international symposium on Physical design*, ser. ISPD '98. New York, NY, USA: ACM, 1998, pp. 173–178.

[8] H. Chan and I. Markov, "Practical slicing and non-slicing block-packing without simulated annealing," in *ACM/IEEE Great Lakes Symp. on VLSI 2004*, 2004, pp. 282–287.

[9] M. Chen and W. Huang, "A two-level search algorithm for 2D rectangular packing problem," *Comp. & Ind. Eng.*, vol. 53, no. 1, pp. 123 – 136, 2007.

[10] S. Imahori, M. Yagiura, and T. Ibaraki, "Improved local search algorithms for the rectangle packing problem with general spatial costs," *EJOR*, vol. 167, no. 1, pp. 48 – 67, 2005.

[11] D. Chen, J. Liu, Y. Fu, and M. Shang, "An efficient heuristic algorithm for arbitrary shaped rectilinear block packing problem," *Comput. Oper. Res.*, vol. 37, pp. 1068–1074, June 2010.

Local search methods for leather nesting problems

Pedro Brás Cláudio Alves José Valério de Carvalho

Centro ALGORITMI / Departamento de Produção e Sistemas, Universidade do Minho
4710-057 Braga, Portugal
{pedro.bras,claudio,vc}@dps.uminho.pt

ABSTRACT

We describe a set of new local search based algorithms for a real leather nesting problem (LNP) arising in the automotive industry. The problem consists in finding the best layouts for a set of irregular shapes within large natural leather hides with highly irregular contours, and which may have holes and quality zones. Our case study comes from a multinational company that produces car seats. The irregular shapes that must be cut from the hides are pieces of these car seats, and they may contain holes and different quality zones. A relevant characteristic of the problem addressed is that the cutting patterns are not subject to any special constraint that may reduce the set of feasible solutions, and hence simplify the problem. The directionality constraints arising in the shoe industry are an example of such constraints.

Very few solution methods were proposed in the literature for this variant of the LNP. The value of the potential savings contrast with this very small number of contributions. Here, we intend to contribute with new solution methods that embeds a new constructive heuristic that we proposed recently in [1]

Keywords: Leather nesting, Variable neighbourhood search

1. INTRODUCTION

The leather nesting problem (LNP) consists in finding the best layouts for a set of irregular shapes within the boundaries of natural leather hides. The leather hides are natural products with irregular contours and a very inhomogeneous surface with holes and different quality levels. Here, we address the real case of a multinational company that produces car seats. The irregular shapes to be cut from the leather hides are pieces of these car seats. The corresponding LNP is one of the most general 2-dimensional nesting problem. The pieces may have holes, and the surface from which they are cut must satisfy minimum quality requirements defined by the clients. These requirements translate into quality zones within the pieces, which in turn restrict the position of the pieces within the hides. The details of this LNP are introduced in Section 2.

The first algorithm reported in the literature for this LNP is due to Heistermann and Lengauer [2]. These authors developed a greedy heuristic that starts by identifying a limited and empty region of the hide where to place one of the available pieces. The selection of this region can be fixed a priori, or it may vary from one iteration to another. The placement of the pieces in this region is evaluated using different criteria such as the area of the piece and the distance between its contour, the borders of the hide and the current partial layout. To repair the eventually infeasible layouts that are built in this way, the authors resort to compaction. The authors argue that their approach is competitive compared to humans. However, they present their results without distinguishing the type of instances from which these results are obtained although this may have a critical impact on the quality of the layouts. Indeed, in the furniture industry, for example, the pieces tend to be much larger than in the automotive industry, and as a consequence, these instances may lead to better layouts.

More recently, Alves *et al.*[1] analyzed a set of constructive heuristics for this LNP. These heuristics rely on the computation of no-fit and inner-fit polygons to ensure feasible placements on the hides. The authors explored different strategies that use directly the information provided by these polygons to guide the selection of the pieces and their placement. Additionally, they explored different criteria to evaluate the quality of a placement. An extensive set of computational experiments on real instances are reported, which pointed to the efficiency of some of the original heuristics explored.

We extend the work of [1], and propose new local search based heuristics that embed the best strategies described in this paper. We propose three alternative sequence-based neighborhood structures. These structures depend on the values provided by the evaluation function used to assess the quality of the placement points. The different neighborhoods are obtained by varying the size of the sets of pieces in the sequence from which a piece can be removed. The pieces that are removed are replaced by another piece. The number of candidate pieces is another parameter that define our neighborhoods. These neighborhoods are explored using the variable neighborhood search metaheuristic described in [3].

In Section 2, we describe the relevant aspects of our LNP. In Section 3, we introduce the constructive strategies followed in our heuristics. In Section 4, we discuss some of the details of our local search procedures, namely the neighborhood structures.

2. THE LEATHER NESTING PROBLEM

In the LNP, we are given a set of small two-dimensional irregular shapes (the pieces of the car seats) and a larger irregular shape representing the leather hides. The objective is to place the pieces on the hide so as to minimize the total empty space (or equivalently, maximize the yield).

The contour of the leather hides is irregular, and their interior is usually inhomogeneous. It may have holes, defects and regions with different levels of quality (the quality zones). The holes and defects of the hides are treated as any other piece that may be already placed on the surface of the hides. The quality zones are treated differently. A piece can only be placed on a given region of the hide only if the quality of this region is greater or equal than the quality requirements of the piece. In the automotive industry, four quality zones are used (A, B, C and D). A stands for the best quality zone. The quality decreases from A to D. Some parts at the boundaries of the hides are considered as waste because their quality is too low to cut any piece.

The pieces that must be placed on the hides are also irregular. They may have holes and different quality requirements. The quality zone of a piece can never be placed on a region of the hide with a lower quality. The characteristics of the pieces that must be cut

from the hides depend on the application. In the shoe industry, the shapes are small compared to the size of the hides. In the furniture industry, many of the pieces are large. In the automotive industry, there is many different pieces. The area of the pieces ranges from 0.1% to 6% of the area of the hides.

A layout consists in the pieces that are placed on the hide, and their corresponding position and rotation. In our case, a layout is feasible if and only if all the pieces do not overlap, if all the pieces are placed inside the usable area of the hide, and if all the quality constraints are satisfied.

3. PLACEMENT STRATEGIES BASED ON INNER-FIT POLYGONS

The no-fit polygons (NFP) are used to determine wether two pieces overlap or not, while the inner-fit polygons (IFP) are used to determine wether a piece is completely contained within another, or not. As noted in [4], the concepts of NFPs and IFPs allow the definition of new placement approaches. In [1], we defined new constructive heuristics that use the information provided by these polygons to guide the selection of the pieces, and the placement of these pieces into specific regions of the hides.

The heuristics proposed in [1] can be divided in four stages. The pieces are first grouped according to a given criterion (area, irregularity, value, for example). These groups are defined such that the pieces with almost the same attributes are treated with the same priority. Then, a piece is selected to be placed on the hide. One of the criteria that we used for selecting a piece was based on the characteristics of the IFP of this piece with the hide. After a piece has been selected, we choose a region inside the hide where the piece will be placed, and we evaluate the possible placement points inside that region. The point that maximizes a given criterion is selected, and the piece is placed at this point. Note that when a piece is selected according to the characteristics of its IFP, the region of the hide where this piece will be placed is inevitably this IFP.

The sequence of pieces that will be used to define our neighborhood structures are obtained with the constructive procedure that relies on the characteristics of the IFPs. To evaluate a placement position, we used a function based on the relative area between an offset of the piece and the area of the polygon resulting from the intersection of this offset with the current layout and the border of the hide.

4. VARIABLE NEIGHBORHOOD SEARCH

Our algorithms are based on the variable neighbourhood search (VNS) metaheuristic. New neighbourhood structures are proposed based on a representation of the solutions as a sequence of pieces combined with the constructive heuristic alluded above.

The selection process generates a sequence of pieces. Each piece is placed in a given region of the hide, which corresponds in fact to a particular IFP of the piece with the hide. For the smallest pieces, the IFP that is chosen is the smallest IFP associated to the piece, while for the largest pieces, the IFP that is selected is the largest one. The next step of the constructive heuristic consists in determining the placement position where the piece should be placed. The possible placement positions of the hide are evaluated based on the criterion described above. It depends on an offset of the piece, and the intersection of this offset with the current layout and the boundary of the hide. For the sake of clarity, we will designate this value as the fitness of the piece.

Our neighborhood structures depend on the sequence of pieces, on the values given by the evaluation function for each piece and on the value of the yield achieved after placing each one of the pieces of the sequence. Let i_j denote the index of the piece in the sequence with a corresponding yield of $j\%$. We explored three neighborhood structures using the following definitions:

- for the pieces between i_{j_1} and i_{j_2}, let k be the piece with the lowest fitness, and g be the group of this piece. The neighborhood solutions consists in all the solutions obtained by removing k, replacing it by a piece from the group g up to $g-p$ (p is a parameter with $p \leq g$), and completing the sequence by running the constructive heuristic;
- for the pieces between i_{j_1} and i_{j_2}, we select a subsequence of n pieces with the lowest total fitness. We replace the first piece of this set (k of group g) by another piece from the group g up to $g-p$. The remaining $n-1$ pieces of the set are replaced by running the constructive heuristic. The final part of the original sequence remains unchanged. The corresponding pieces are placed on the hide using the criteria used in the constructive heuristic;
- for the pieces between i_{j_1} and i_{j_2}, we select n pieces with the lowest fitness. These pieces are replaced by other pieces from the corresponding group g up to the group $g-p$, while the remaining subsequences of the original sequence remains unchanged.

Note that, in the previous definitions, j_1, j_2, p, and n are all parameters that allow to configure the different neighborhoods that will be explored using VNS.

In our first implementation, we considered the basic version of VNS described in [3]. The preliminary tests realized on a set of real instances yield promising results. Further experiments are being conducted on an extensive set of real instances to analyze the best set of parameters that should be applied, and also to analyze the impact of using different constructive heuristics.

5. CONCLUSIONS

The LNP with no specific constraints remains a challenge that deserves attention given the potential for savings associated to the value of the raw material involved. Recently, the authors proposed new constructive heuristics for this problem. In this extended abstract, we gave some of the details of a local search approach that extends our previous work on that problem. The methods proposed are based on three different neighborhood structures that depends on the sequence of pieces generated by the constructive procedure.

6. ACKNOWLEDGEMENTS

This work was partially supported by the Algoritmi Research Center of the University of Minho for Cláudio Alves and José Valério de Carvalho and by the Portuguese Science and Technology Foundation through the research grant SFRH/ BDE/15650/2007 for Pedro Brás.

7. REFERENCES

[1] C. Alves, P. Brás, J. Valério de Carvalho, and T. Pinto, "New constructive algorithms for leather nesting in the automotive industry," *submitted*, 2011.

[2] J. Heistermann and T. Lengauer, "The nesting problem in the leather manufacturing industry." *Annals of Operations Research*, vol. 57, pp. 147–173.

[3] P. Hansen and N. Mladenovic, "Variable neighborhood search: principles and applications," *European Journal of Operational Research*, vol. 130, pp. 449–467, 2001.

[4] J. Bennell and J. Oliveira, "The geometry of nesting problems: a tutorial," *European Journal of Operational Research*, vol. 184, no. 2, pp. 397–415, 2008.

Nesting Problems: mixed integer formulations and valid inequalities

Antonio Martínez Sykora * Ramón Álvarez-Valdés Olaguíbel *
José Manuel Tamarit Goerlich *

* Universidad de Valencia, Departamento de Estadística e Investigación Operativa
C/Dr. Moliner, 50, 46100, Burjassot, Valencia
{antonio.martinez-sykora, ramon.alvarez, jose.tamarit}@uv.es

ABSTRACT

Cutting and packing problems involving irregular shapes, usually known as Nesting Problems, are common in industries ranging from clothing and footwear to engineering and shipbuilding. The research publications on these problems are relatively scarce, compared with other cutting and packing problems with rectangular shapes, and have been mostly focused on heuristic approaches. In this paper we propose a new mixed integer formulation for the problem and derive some families of valid inequalities, as a first step for developing an exact Branch & Cut Algorithm.

Keywords: Cutting and Packing, Nesting, Integer Programming

1. INTRODUCTION

Nesting problems are two-dimensional cutting and packing problems involving irregular shapes. These problems arise in a wide variety of industries like garment manufacturing, sheet metal cutting, furniture making and shoe manufacturing.

There are several types of nesting problems depending on the rotation of the shapes. We can define three types of problems:

- Without rotation: The pieces have a fixed orientation.
- With specific angles of rotation: The pieces can be placed with any of the specific angles. Usually these angles are 0^o, 90^o and 180^o.
- With rotation: Pieces can be placed with any angle $\theta \in [0, 2\pi[$.

In this work we study the nesting problem as the problem of arranging a set of two-dimensional irregular shapes without overlapping in a rectangular stock sheet with fixed width where the objective is to minimize the require length. We will consider that pieces cannot be rotated. This problem arises, e.g, in the garment manufacturing, because in this industry the pattern of the fabric must be respected. An example of a layout from the garment manufacturing industry is provided in figure 1.

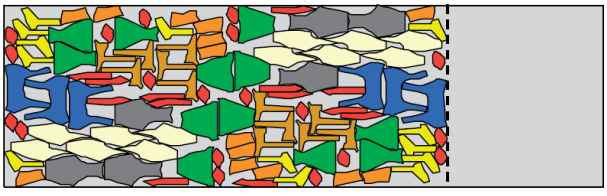


Figure 1: An example layout from garment manufacturing

The main difficult of nesting problems is to ensure that the pieces have a non-overlapping configuration. This question has been studied deeply in recent years and there are several approaches which determine when two polygons overlap. Bennell and Oliveira [2] give a tutorial of the different approaches which study the geometry of nesting problems. The problem is NP-complete and as a result solution methodologies predominantly utilise heuristics.

We consider the pieces approximately described by polygons. The most used tool to check if two polygons overlap is the Non Fit Polygon (NFP). It can be used, along with the vector difference of the position of the two polygons, to determine whether these polygons overlap, touch, or are separated, by conducting a simple test to identify whether the resultant vector is inside the NFP.

The formulation proposed in this paper uses the Non Fit Polygons to create inequalities for separating each pair of pieces. There are two different formulations using the NFPs. The first one is used in the Simulated Annealing Algorithm proposed by Gomes and Oliveira ([1]). In this formulation, they use binary variables and the *big M* constant to activate and inactivate each convex region given by the NFP. Fischetti and Luzzi ([3]) propose a more efficient formulation by defining *slices* to have a partition of the feasible places in which to arrange each pair of pieces without overlap. The *slices* must be disjoint but they do not specify how they build them. Our formulation is similar to the Fischetti and Luzzi formulation (FLF), but we consider horizontal *slices*.

2. MIXED INTEGER FORMULATION FOR *NESTING* PROBLEMS

Let $P = \{p_1, \ldots, p_N\}$ be the set of pieces to arrange in the strip. We consider that the reference point of each piece is the bottom left corner of the enclosing rectangle. We denote by (x_i, y_i) the coordinates of the reference point of piece p_i. Let l_i (w_i) be the length (width) and let L and W be the length and width of the strip. We consider that the bottom left corner of the strip is placed at the origin.

The NFP_{ij} is the region in which the reference point of piece p_j cannot be placed because it would overlap with piece p_i (see figure 2). The feasible zone to place p_j with respect to p_i is a non-convex polygon or it could be unconnected. In the next section we present the *Horizontal Slices*, which consist of dividing this feasible zone into convex polygons and assigning a binary variable to each one of these polygons.

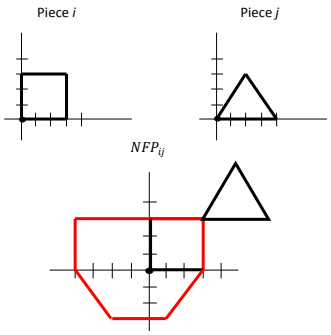


Figure 2: NFP_{ij}. If the reference point of p_j is in the NFP_{ij} then p_j overlaps p_i.

2.1. Horizontal Slices

Let $NFP_{ij} := \{r_1, \ldots, r_n\}$ be the NFP of pieces p_i and p_j such that $r_t, \forall t \in \{1, \ldots, n\}$, represents the vertices of the NFP in anticlockwise order. In order to build the horizontal slices, we require the NFP_{ij} to be convex. There are two possibilities:

- The NFP_{ij} has no concavities. We define one horizontal slice for each edge.
- The NFP_{ij} has concavities. We *close* all the concavities in order to obtain a convex polygon. In this case we build a horizontal slice for each edge of the modified NFP_{ij} and for each created *hole*. If the polygon has k concavities then we build k holes of the NFP_{ij}.

To each slice we associate a binary variable b_k which takes the value 1 if the reference point of piece j is in the slice and 0 otherwise. The set of all binary variables associated with a NFP_{ij} is denoted by $VNFP_{ij}$. In figure 3 we can find the set of variables associated to NFP_{ij}. Variable b_{ij4} corresponds to the concavity of the NFP_{ij}.

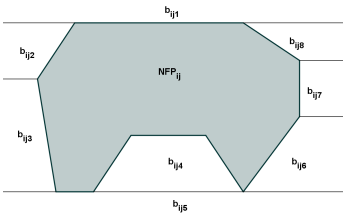


Figure 3: *Horizontal Slices*

2.2. NFP constraints

For each pair of pieces (p_i, p_k), we use the NFP_{ij} to build the necessary constraints to place this pair of pieces without overlap. Let us consider the binary variables $b_{ij} \in VNFP_{ij}$ defined above. Let us denote by m_{ij} the number of binary variables considered in $VNFP_{ij}$. Each slice is described by several inequalities. The slices are limited by L_{sup}, an upper bound for the length of the strip.

We use the constraints proposed by Fischetti and Luzzi (FLF) [3], adapting them to our horizontal slices and closed concavities:

$$\alpha_{ij}^{kf}(x_j - x_i) + \beta_{ij}^{kf}(y_j - y_i) \leq \sum_{h=1}^{m_{ij}} \delta_{ij}^{kfh} b_{ijh} \qquad (1)$$

where the coefficients α_{ij}^{kf} and β_{ij}^{kf} are the coefficients of the *NFP*-inequality f and δ_{ij}^{kfh} are the greatest value the left hand side can take when *slice* defined by b_{ijh} is turned on.

Note that for each NFP_{ij} it is necessary that one binary variable $b_{ijk} \in VNFPij$ takes value one for separating pieces p_i and p_j. Then we need the following equalities in the formulation:

$$\sum_{k=1}^{m_{ij}} b_{ijk} = 1, \quad \forall 1 \leq i \leq j \leq N \qquad (2)$$

2.3. Bounds for the position of the pieces

Each piece must be placed entirely into the strip so the reference point must satisfy some bound constraints. The usual bound constrains are:

$$0 \leq x_i \leq L - l_i, \quad \forall i = 1, ..., N \qquad (3)$$
$$0 \leq y_i \leq W - w_i, \quad \forall i = 1, ..., N \qquad (4)$$

We add to the formulation more bound constraints by lifting these inequalities. Let L_{ij} (R_{ij}) and D_{ij} (U_{ij}) be the subsets of binary variables such that piece i protrude from the left (right) or below (over), respectively, of piece j. Let λ_{ij}^k (μ_{ij}^k) be the minimum quantity such that piece p_j protrude horizontally (vertically) to piece p_i when the slice defined by $b_k \in VNFP_{ij}$ is turned on.

For each one of the inequalities (3) and (4) we build N inequalities by adding binary variables as follows:

$$x_i \leq L - l_i - \sum_{b_{ijk} \in L_{ij}} \lambda_{ij}^k b_{ijk}, \quad \forall i, j \in \{1, ..., N\} \qquad (5)$$
$$y_i \leq W - w_i - \sum_{b_{ijk} \in D_{ij}} \mu_{ij}^k b_{ijk}, \quad \forall i, j \in \{1, ..., N\} \qquad (6)$$

Inequalities (5) indicate that if any binary variable b_{ijk} which forces piece p_j to be placed at the right of piece p_i is turned on then the length of the strip L must be greater than $x_i + l_i + \lambda_{ij}^k$. Inequalities (6) have the same idea in a vertical direction.

We use a similar idea to lift the inequalities on the left of (below) the strip. In (8) and (9) of the formulation we can see these bound constraints.

2.4. Mixed Integer Formulation

The **Horizontal Slices Formulation** (HSF) is the following one:

$$\text{Objective Function:} \quad \min L \qquad (7)$$

s.t.

$$\sum_{b_{ijk} \in R_{ij}} \lambda_{ij}^k b_{ijk} \leq x_i \leq L - l_i - \sum_{b_{ijk} \in L_{ij}} \lambda_{ij}^k b_{ijk}, \qquad (8)$$
$$\forall i, j \in \{1, \ldots, N\}$$
$$\sum_{b_{ijk} \in U_{ij}} \mu_{ij}^k b_{ijk} \leq y_i \leq W - w_i - \sum_{b_{ijk} \in D_{ij}} \mu_{ij}^k b_{ijk}, \qquad (9)$$
$$\forall i, j \in \{1, \ldots, N\}$$
$$\alpha_{ij}^{kf}(x_j - x_i) + \beta_{ij}^{kf}(y_j - y_i) \leq \sum_{h=1}^{m_{ij}} \delta_{ij}^{kfh} b_{ijh}, \qquad (10)$$

$$\forall 1 \leq i \leq j \leq N,\ \forall k = 1,...,m_{ij} \quad (11)$$

$$\sum_{k=1}^{m_{ij}} b_{ijk} = 1, \quad \forall 1 \leq i \leq j \leq N \quad (12)$$

$$b_{ijk} \in \{0,1\}, \quad \forall 1 \leq i \leq j \leq N \quad (13)$$

The objective function minimizes the required strip length (7). Constraints (8) constraints (9) are the bound constraints of the pieces. Inequalities (10) are the corresponding *NFP* inequalities and constraints (12) indicate that one slice of each *NFP* must be turned on (inequalities (2)).

3. VALID INEQUALITIES FOR NESTING PROBLEMS

In this section we present some valid inequalities for the *HSF*. When we relax the integer conditions of the Mixed Integer Formulation we usually obtain a non integer solution. The inequalities presented here can be very useful if we want to cut some non valid solutions. The first kind of inequalities are the *LU covers*. These inequalities ensure that the columns of pieces fit into the strip. The same idea is used in the second inequalities, the *cliques* and *covers*. The third inequalities are the *Transitivity Constraints* in which the idea is to place a set of pieces consistently, and do not turn on variables which are incompatible. Finally, we introduce the impenetrability constraints relating binary variables with the variables associated to the reference points of the pieces.

3.1. *LU-cover* inequalities

Let us consider the polygon of the NFP_{ij} where the reference point of piece p_i is placed at $(0,0)$. Let us denote by $\overline{Y}_{ij}$ ($\underline{Y}_{ij}$) the maximum (minimum) value of the NFP_{ij} on the $Y-axis$ and let $\overline{y}_{ijk}$ ($\underline{y}_{ijk}$) be the maximum (minimum) value of the slice on the $Y-axis$.

Let us consider that variable b_{ijk} is turned on. If we want to know how much the piece p_j protrudes from the piece p_i (or viceversa) in a vertical way we need to calculate $\overline{Y}_{ij} - \underline{y}_{ijk}$ (if $\underline{y}_{ijk} > 0$) or $(-1)\underline{Y}_{ij} - (-1)\overline{y}_{ijk}$ (if $\overline{y}_{ijk} < 0$). This difference can be viewed as the quantity of width that the pieces share. Then we compare this difference with the minimum width of the pieces p_i and p_j ($\min_{i,j}\{w_i, w_j\}$). If the difference is lower than the minimum width, there is a part of piece p_j which protrudes from piece p_i. In case that $\underline{y}_{ijk} < 0$ and $\overline{y}_{ijk} > 0$ the *slice* allows to place the reference point of the two pieces on the same *y*-coordinate, and in this case the pieces do not pile up.

Let p_i y p_j be two pieces. We denote by $\mathbf{U}^*_{\mathbf{ij}}$ ($\mathbf{D}^*_{\mathbf{ij}}$) the subsets of binary variables which define *slices* of the NFP_{ij} such that, when they are turned on, they put p_j above p_i (p_j below p_i):

$$\begin{aligned} \mathbf{U}^*_{\mathbf{ij}} &:= \{b_{ijk} \mid \overline{Y}_{ij} - \underline{y}_{ijk} < w_{ij}\} \\ \mathbf{D}^*_{\mathbf{ij}} &:= \{b_{ijk} \mid (-1)\underline{Y}_{ij} - (-1)\overline{y}_{ijk} < w_{ij}\} \end{aligned}$$

where $w_{ij} := \min\{w_i, w_j\}$.

Let $C = \{p_1, \ldots, p_r\}$, $1 < r \leq N$, be a subset of r pieces, and let $U'_{st} \subseteq U^*_{st}$, $U'_{st} \neq \emptyset$ and $D'_{st} \subseteq D^*_{st}$, $D'_{st} \neq \emptyset$, $\forall 1 \leq s < t \leq r$. We denote by $UD'_{st} := U'_{st} \cup D'_{st}$. Note that $U'_{st} = D'_{ts}\ \forall p_s, p_t \in C$.

Proposition:
Let

$$\delta := \max_{\tau \in \pi\{C\}} \{\sum_{t=1}^{r-1} \sum_{l \in U'_{\tau(t)\tau(t+1)}} q_{\tau(t)\tau(t+1)l} b_{\tau(t)\tau(t+1)l}\}$$

and let $q_{\tau(t)\tau(t+1)l}$ be the amount of overlapping along the *Y-axis* between piece $\tau(t+1)$ and $\tau(t)$ when $b_{\tau(t)\tau(t+1)l}$ is turned on. $\pi\{C\}$ is the set of all the permutations of the pieces in C. Therefore, δ is the maximum overlap between the pieces of C in any order.

If inequality (14) is satisfied, then constraint (15) is a valid inequality for the *Nesting* problem. We say that constraint (15) is a **LU-cover** inequality.

$$\sum_{s=1}^{r} w_s - \delta \quad > \quad W \quad (14)$$

$$\sum_{s=1}^{r-1} \sum_{l=s+1}^{r} \sum_{k \in UD'_{sl}} b_{slk} \quad \leq \quad \sum_{s=1}^{r-1} (r-s) - 1. \quad (15)$$

3.2. *Cliques* and *covers*

These constraints are based on the same idea of the *LU covers* inequalities but in this case we consider a fixed permutation of the r pieces, e.g $\{p_1, \ldots, p_r\}$, and we have to check whether condition (14) is satisfied by the given permutation. In this case we only consider the *NFP*s that separate adjacent pieces in the order given by the permutation. That implies that inequality (15) has fewer variables.

We present only the case of three pieces, but it could be generalized to r pieces. The case of the three pieces, called *cliques*, has a right hand side of 1, and the case of r ($r > 3$) pieces, called *covers*, has a right hand side of $r-2$.

Proposition:
Let us consider three pieces, p_i, p_j and p_k. If there are two subsets $U'_1 \subseteq U_{jk}$, $U'_2 \subseteq U_{kl}, U'_2 \neq \emptyset$, such that $\forall s \in U'_1$ and $\forall t \in U'_2$ $\underline{y}^s_{i_j i_k} + \underline{y}^t_{i_k i_l} > W - w_l$ is satisfied, then inequality (16) is valid.

$$\sum_{s \in U'_1} b_{jks} + \sum_{s \in U'_2} b_{kls} \leq 1. \quad (16)$$

These inequalities could also be define in a horizontal sense.

3.3. Transitivity Inequalities

These constraints are designed to forbid incompatible slices being turned on. In other words, if two slices separating pieces $1-2$ and $1-3$ are turned on, the relative position of pieces $2-3$ can be limited and there could exist slices from NFP_{23} such that are incompatible with the previous ones.

In this section we present only the transitivity inequalities involving three pieces. This idea could be generalized considering n pieces, but it would be more complicated with more computational effort.

Proposition:
Let us consider 3 pieces, i, j y k. Let b_{ij1}, b_{ik1} and b_{jk1} be three variables defining, respectively, one *slice* of the NFP_{ij}, NFP_{ik} and NFP_{jk}. If $b_{ij1} = b_{ik1} = 1$ they define a region for the relative position of p_k with respect to p_j. If the slice defined by b_{jk1} does not intersect this region then these three variables cannot be equal to 1 simultaneously and the corresponding transitivity constraint is:

$$b_{ij1} + b_{ik1} + b_{jk1} \leq 2 \quad (17)$$

If there are other variables of NFP_{ij} incompatibles with b_{ik1} and b_{jk1} then can be added to the right hand side of (17).

3.4. Impenetrability Inequalities

The *Impenetrability* inequalities are based on the study of the sum of the coordinates of the pieces. If we relax the integer conditions of the variables and we solve the problem, then it is usual to find that all the pieces have been placed close to the origin. The idea of these inequalities is to move the pieces beyond the origin, depending on which binary variables are positive.

Let p_i and p_j be two pieces, $1 \leq i < j \leq N$. Then, we study how much the value of the sum $S := x_i + x_j + y_i + y_j$ could be improved using the binary variables. The idea is to minimize S in each one of the slices defined by the NFP_{ij}. An *Impenetrability* constraint has the following form:

$$S \geq \sum_{k=1}^{m_{ij}} \omega_{ij}^k b_{ijk}, \tag{18}$$

where the coefficients ω_{ij}^k are the solutions of the linear problem which consist of minimizing S subject to the constraints that define the slice b_{ijk}. These inequalities are valid by construction.

It would be interesting to add to the inequality other variables corresponding to other *NFP*s. Let us consider p_r and a variable $b_{irl} \in VNFP_{ir}$. If we want to include this variable to the right hand side of (18), we have to study in which way the coefficients ω_{ij}^k have to be modified. This study requires to check all the coefficients every time we want to include a new variable.

4. CONCLUSIONS

In this paper we have proposed a new Mixed Integer Formulation for the Nesting Problem. The HS formulation modifies the FL formulation in two ways. On the one hand, the definition of horizontal slices, which restrict the vertical position of the pieces. On the other hand, the lifted bound constraints. The use of horizontal slices allows us to fix many binary variables to 0. We have also introduced some new valid inequalities, which have been found studying the linear relaxation of the formulation. Again, the horizontal slices are very useful for defining strong valid inequalities. In these two aspects, the proposed formulation seems to improve the previous ones, as a preliminary computational experience has shown.

This work can be considered the first part of a study about this problem that will lead us to the design and implementation of exact and heuristic procedures. More concretely, in the second phase of our work we are developing a Branch-and-Cut algorithm. The formulation and the valid inequalities presented in this paper are the basic components of the algorithm, but other important questions have to be addressed, such as the branching strategy and the development of efficient separation algorithms for identifying violated inequalities.

5. ACKNOWLEDGEMENTS

This study has been partially supported by the Ministerio de Ciencia e Innovación of Spain through project DPI2008-02700, co financed by FEDER funds.

6. REFERENCES

[1] A.M.Gomes and J.F.Oliveira, "Solving irregular strip packing problems by hybridising simulated annealing and linear programming," *European Journal of Operational Research*, vol. 171, pp. 811–829, Oct. 2006.

[2] J.A.Bennell and J.F.Oliveira, "A typology of cutting and packing problems," *European Journal of Operational Research*, vol. 184, pp. 397–415, Nov. 2008.

[3] M.Fischetti and I.Luzzi, "Exact and heuristic mip models for nesting problems," 2003.

Matheuristics for Traffic Counter Location

Marco A. Boschetti * Vittorio Maniezzo † Matteo Roffilli †
Antonio José Bolufé Röhler ‡

* Dept. Mathematics
University of Bologna, Cesena, Italy
marco.boschetti@unibo.it

† Dept. Computer Science
University of Bologna, Cesena, Italy
vittorio.maniezzo@unibo.it

‡ Dept. Artificial Intelligence and Computer Systems
University of Habana, Habana, Cuba
bolufe@matcom.uh.cu

ABSTRACT

Matheuristic algorithms have begun to demonstrate that they can be the state of the art for some optimization problems. This paper puts forth that they can represent a viable option also in an applicative context. The possibility to get a solution quality validation or a model grounded construction may become a significant competitive advantage against alternative approaches. This view is substantiated in this work by an application on the problem of determining the best set of locations for a constrained number of traffic counters, to the end of estimating a traffic origin / destination matrix. We implemented a Lagrangean heuristic and tested it on instances of different size. A real world use case is also reported.

Keywords: Matheuristics, Traffic counters, Location problems, Real world applications

1. INTRODUCTION

Matheuristic algorithms are the state of the art for some optimization problems [1, 2, 3] and, besides their theoretical involvement, they can represent a viable option also in an applicative context. In fact, the possibility to get an online validation of the solution quality, for example by means of a bound, or a model grounded construction which justifies construction choices, may be a significant competitive advantage against alternative approaches. In spite of the relative youth of this application field, several works have in fact reported about the possibility to use matheuristics techniques for implementing applications targeted for real-world deployment.

This possibility is substantiated also in this work by an application on the problem of determining the best locations for a given number of traffic counters, to the end of estimating a traffic *Origin - Destination matrix* (*OD* matrix) of traffic flows. The application supports a planner in inferring the *OD* matrix by determining where to locate counters in such a way that the chosen positions will be the most informative for the specific estimation algorithm that shall be used.

The problem is already known in the literature, where it was presented under the name of *Network Count Location Problem* (NCLP). A problem closely related to the NCLP is the *Link Count Location Problem* (LCLP), which asks to determine the best position of a counter along a link. In this work we are only interested in the NCLP, possibly leaving the LCLP as a further study.

The most relevant literature contributions for the NCLP include the work of Ehlert et al. [4], where they propose a MIP-based tool which was put to actual use on a road network of 1414 directed links, divided into 23 O/D zones. This approach is related to the one we put forth here, while different approaches were used by Yang and Zhou [5], who used selection rules, and by Bell and Grosso [6, 7], who used path flows estimations. Overviews are also available for this problem, for recent ones see Cascetta and Pastorino [8] and Wang et al. [9].

2. PROBLEM SPECIFICATION

The general context in which the problem arises is that of inferring an *OD* matrix of traffic flows. Within this framework, the NCLP asks to determine which is the best positioning for a set C of traffic counters, that is, the positions which provide most information to a subsequent *OD* estimation algorithm. This should take into account also the possibility of having pre-installed fixed counters which cannot be moved and whose information must be considered for the subsequent OD estimation.

One possible formulation of the problem is the following.

Given a road network N represented by a multigraph $G = (V,A)$, with $V = V_s \cup V_c$ and $A = A_s \cup A_c$ where A_s is the subset of actual road network arcs, representing the different lanes of the roads of interests (or the carriageways in case of motorways), V_s the subset of its endpoints (crossways of the road network), V_c is a subset of dummy nodes, each of which is associated with an origin or with a destination and A_c is a subset of dummy arcs, which connect each origin or destination node to all nodes in V_s belonging to the zone modeled by that origin or destination.

We want to determine the arcs where the counters of set C are to be most conveniently located. That is, we want to determine the arc subset $\bar{A}$, $\bar{A} \subseteq A_s$, on whose arcs a traffic count $\bar{f}_{ij}$ will be obtained.

An obvious precondition is the ability to determine an estimate of the traffic flow f_{ij} on each arc $(i, j) \in A$. Details on a possible procedure for this can be found in Gabrielli et al. [10, 11]. An actual traffic count, $\bar{f}_{ij}$, could also be already available for the arcs of a subset of A_s.

The *OD* matrix is modeled as an index set $\Lambda = [\ell]$ of OD pairs, each of them with an associated demand $\omega_\ell \in \Omega$. Demands will even-

tually distribute over traffic flows φ_p, running on directed paths p, $p \in \Pi_\ell$, where Π_ℓ is the index set of paths for OD pair ℓ and $\Pi = \bigcup_{\ell \in \Lambda} \Pi_\ell$. The objective asks to minimize an additive generalized cost, which can be computed for each arc (i,j) in relation to the time needed for traveling through the arc, in accordance to Wardrop's principle, and it is a function $c_{ij}(f_{ij})$ of the flow through it. The basic traffic assignment problem is thus as follows:

$$(TAP)\ min \sum_{(i,j)\in A} c_{ij}(f_{ij}) \quad (1)$$

$$s.t. \sum_{p\in\Pi_\ell} \varphi_p = b_\ell, \qquad \ell \in \Lambda \quad (2)$$

$$f_{ij} = \sum_{p\in\Pi} \delta_{ij}^p \varphi_p \le u_{ij} \qquad (i,j) \in A \quad (3)$$

Here b_ℓ represents the origin to destination demand for OD pair ℓ, δ_{ij}^p is a constant equal to 1 if arc (i,j) belongs to path p, 0 otherwise, and u_{ij} is the theoretical capacity of arc (i,j).

A significant problem to be faced in this kind of applications is the inherent unreliability of the OD matrix. The matrix is usually obtained from interviews and/or inductions from geographic and economical data, and it is therefore intrinsically approximated. Moreover, OD data is possibly obsolete. This motivated substantial research aimed at updating OD matrices, including several methods based on actual traffic counts on road arcs.

The OD matrix estimation problem was modeled as a constrained quadratic optimization problem. Input data are the flows φ_p on each path $p \in \Pi$, the old OD matrix, $\Omega = \bar{[}\ \bar{}\ \varphi\bar{]}$, the set $\bar{F} = \{\bar{f}_{ij}\}$ of the sampled flows for each arc in $\bar{A}$, a lower bound L_ℓ and an upper bound U_ℓ for each OD pair $\ell \in \Lambda$.

The new OD matrix is computed as a trade-off between the objective of minimizing the quadratic difference from $\bar{\Omega}$ and that of minimizing the difference of the flows f_{ij} induced in each arc in $\bar{A}$ with $\bar{f}_{ij}$, where the $\bar{f}_{ij}$ are measured by actual traffic counters, under constraints on L_ℓ and U_ℓ. To compute it, we need the *usage ratio* of each arc (i,j) for each pair ℓ, which is computed as $\rho_{ij}^\ell = \frac{\sum_{p\in\Pi_\ell} \delta_{ij}^p \varphi_p}{\sum_{p\in\Pi_\ell} \varphi_p}$, where Π_ℓ is the index set of all paths for OD pair ℓ as computed by the assignment. The formulation of the OD refinement problem becomes as follows:

$$(ODP)\ min \sum_{\ell\in\lambda} (\omega_\ell - \bar{\omega}_\ell)^2 + \gamma \sum_{(i,j)\in A} \left(\sum_{\ell\in\Lambda} \omega_\ell \rho_{ij}^\ell - \bar{f}_{ij} \right) \quad (4)$$

$$s.t. L_\ell \le \omega_\ell \le U_{ij} \qquad \ell \in \Lambda \quad (5)$$

where γ is a user-defined parameters which biases the result toward having an OD matrix structurally close to the old one and away from having assignments close to the sampled ones, or vice-versa.

To determine subset $\bar{A}$ we propose to use the following model. The model is based on an operational assumption: each counter, when placed on a two way road, is able to provide data for both driving directions. Therefore, one counter will provide data for two arcs in A, when the two correspond to the driving directions of a two-way road. We need anyhow to have counting data associated to arcs in order to provide the needed input to the OD estimator.

In the model, we associate a binary variable x_{ij} to each arc (i,j) of the road network N. Each network arc $(i,j) \in N$ could correspond to one arc $(i,j) \in A$ or to a pair of arcs $(i,j) \in A$, $(j,i) \in A$, depending on whether it is a one-way or a two-way road. The x_{ij} variable is equal to 1 iff the arc will be chosen for hosting a counter. Furthermore, we associate a binary variable ξ_p to each possible path p between origins and destinations in N (i.e., between nodes in V_c). The model tries to minimize the number of OD pairs (i.e., the number of paths) which won't be sampled by any counter. Variables ξ act as slacks in the covering constraints, permitting to cover a path with an expensive slack variable if no counter can be used. The price c_p of each ξ_p variable could also be a function of prior OD values, when available. The problem asks then to solve the following Set Covering problem with an additional knapsack constraint:

$$(TCL)\ min \sum_{p\in\Pi} c_p \xi_p \quad (6)$$

$$s.t. \sum_{(ij)\in N} a_{ij}^p x_{ij} + \xi_p \ge 1, \qquad p \in \Pi \quad (7)$$

$$\sum_{(ij)\in N} x_{ij} \le n, \quad (8)$$

$$x_{ij}, \xi_p \in \{0,1\} \qquad (i,j) \in N, p \in \Pi \quad (9)$$

where n is the cardinality of C and a_{ij}^p is a coefficient equal to 1 if arc (i,j) enters path p, 0 otherwise. Notice that x variables can be fixed to trivially account for pre-installed counters.

3. A LAGRANGEAN SOLUTION

Formulation TCL can be effectively solved for small to mid sized problem instances. This is already enough for a number a of real world applications, thus a direct use of a MIP solver is an option to consider when facing an actual case. However, instances could become too big to be solved to optimality within a required time limit. In these cases heuristics are in order. We propose a Lagrangean approach for designing a metaheuristic able to effectively cope with big TCL instances.

3.1. Lagrangean relaxation

Formulation TCL can be simplified by relaxing the covering constraints 7, or the knapsack constraint 8 or both. After some preliminary testing, we went for option one and we relaxed the covering constraints, keeping the knapsack. The relaxed formulation becomes the following.

$$(LTCL)\ min \sum_{p\in\Pi} (c_p - \lambda_p)\xi_p - \sum_{p\in\Pi} \sum_{(ij)\in N} \lambda_p a_{ij}^p x_{ij} + \sum_{p\in\Pi} \lambda_p \quad (10)$$

$$s.t. \sum_{(ij)\in N} x_{ij} \le n, \quad (11)$$

$$x_{ij}, \xi_p \in \{0,1\}, \qquad (i,j) \in N, p \in \Pi \quad (12)$$

$$\lambda_p \ge 0 \qquad p \in \Pi \quad (13)$$

The deriving subproblem, with given penalties, can be easily solved by inspection, by setting to 1 all ξ variables with negative coefficient and by choosing the n variables of type x_{ij} with greater coefficients.

3.2. Lagrangean Metaheuristics

Formulation LTCL can be used both for obtaining a bound on the optimal solution cost and a feasible, high quality solution. We went along, implementing a Lagrangean metaheuristic [12] for the TCLP, based on a subgradient solution of the Lagrangean dual of

formulation LTCL. The general structure of the algorithm is as in Boschetti, Maniezzo [13]:

LAGRHEURISTIC()
1: identify an "easy" subproblem LR(λ)
2: **repeat**
3: solve subproblem LR(λ) obtaining solution **x**
4: check for unsatisfied constraints
5: update penalties λ
6: construct problem solution using **x** and λ
7: **until** (end_condition)

where subproblem LR corresponds to LTCL, and penalty updates is implemented as an adaptive subgradient algorithm, as specified in Boschetti et al. [12].

In our case, each iteration of the subgradient algorithm directly provides also a feasible problem solution, as the inspection of LTCL variable costs permits to determine a subset of n arcs, which will be those suggested for locating the traffic counters. A simple local search is used (and needed) to fine-tune the solutions.

4. USE CASES

We implemented an operational solution, coding the above algorithm in c# under .Net framework 4. The solution comprises also an IP optimization of formulation TCL, empowered by CoinMP (for which a c# wrapper is freely available [14]). Data was imported and exported from ESRI ArcGis [15] and preprocessed in PostGIS [16]. We had the possibility to test our approach on three real-world instances, defined on data of three municipalities in northern Italy.

The main characteristics of the instances are summarized in Table 1, where the columns show:

- *id*: an identifier of the instance
- *Surf*: the surface of the municipality, in square Km
- *Inh*: the number of inhabitants of the municipality
- *Dens*: the resident population density of the municipality
- *Nodes*: the number of nodes of the road graph
- *Arcs*: the number of arcs of the road graph
- *Zones*: the number of zones for which the OD movements are to be estimated
- *Count*: the number of counters to locate

In all instances the number of counters to locate is to be intended as a number in addition to those already installed in the territory.

Municipality				Road graph			
id	Surf	Inh	Dens	Nodes	Arcs	Zones	Count
A	56.89	10651	187	795	1898	14	25
B	45.13	25375	562	1904	5210	12	24
C	7.58	10275	1355	3469	8136	13	28

Table 1: Real world instances.

Notwithstanding with the relative small scale of the tested instances - which is anyway aligned with that of the biggest instances so far presented in the literature - the results were of interest. Each instance could be solved in less than 10 seconds on a 3 GHz Pentium Duo machine with 2 Gb of RAM, providing solutions which were of interest for the final user.

Figures 1 present input data (top) and final solution (bottom, counted arcs in red) for instance A, the smallest of the three. A noteworthy

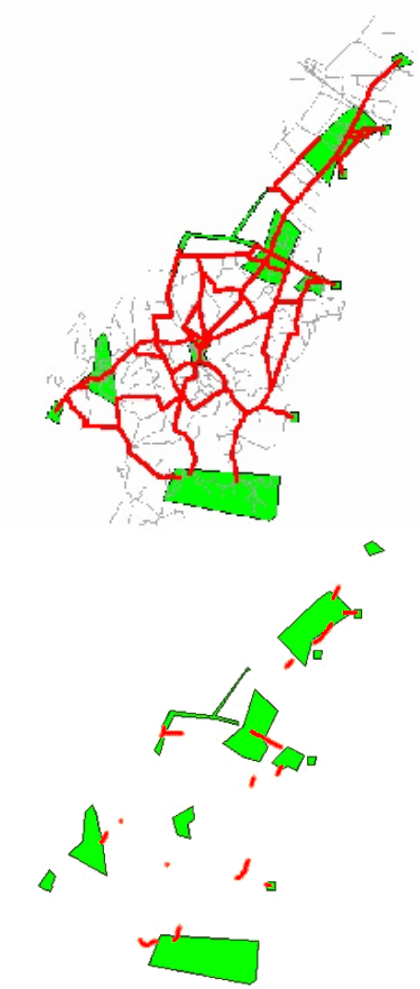


Figure 1: *Instance A: OD zones and transfer paths (top), counted arcs (bottom).*

characteristic of the solution was that the counting locations were set on nonintuitive arcs. In several cases in fact it is suggested to count traffic flows composed by many paths, which can be disambiguated considering the whole set of observations.

Figure 2 present a wide area view of the territory of interest for instance B, as several zones were defined outside of the municipality of interest because significant flows were originated far from the municipality. It was requested to also determine the flows specifically originated from the (relatively) far origins. In fact, some arcs correspond to highway tracts. The different zones internal to the municipality are here condensed in the central cluster. Again, the solution was able to provide a feasible scenario of interest for the operator.

Finally, figure 3 presents a wide area view of instance C, where the smallest roads are not drawn. The same considerations made for instance B can be applied also here.

In conclusion, we like to point out how the proposed procedure proved effective in the operational contexts where it was tested. A strong point we like to make is that the procedure was used in an operational process, dealing with real-world data and constraints and operating on a legacy field system, thus providing an endorsement for the use of matheuristics in real-world applications.

We are now considering bigger size instances. We are confident that the procedure can be used also for bigger municipalities as its primary use is for the location of *additional* counters, and the already located ones do not increase the instance complexity.

Figure 2: *Instance B: OD zones and transfer paths.*

Figure 3: *Instance C: OD zones and transfer paths.*

5. REFERENCES

[1] P. Hansen, V. Maniezzo, and S. Voss, "Special issue on mathematical contributions to metaheuristics editorial," *Journal of Heuristics*, vol. 15, no. 3, pp. 197–199, 2009.

[2] M. A. Boschetti, V. Maniezzo, M. Roffilli, and A. B. Röhler, "Matheuristics: Optimization, simulation and control," in *Hybrid Metaheuristics*, 2009, pp. 171–177.

[3] V. Maniezzo, T. Stützle, and S. Voss, Eds., *Matheuristics: Hybridizing Metaheuristics and Mathematical Programming*, 1st ed., ser. Annals of Information Systems. New York: Springer, 2010, no. 10, iSBN: 978-1-4419-1305-0.

[4] A. Ehlert, M. G. H. Bell, and S. Grosso, "The optimisation of traffic count locations in road networks," *Transportation Research Part B: Methodological*, vol. 40, no. 6, pp. 460–479, 2006.

[5] H. Yang and J. Zhou, "Optimal traffic counting locations for origin-destination matrix estimation," *Transportation Research Part B: Methodological*, vol. 32, no. 2, pp. 109 – 126, 1998.

[6] M. Bell and S. Grosso, "The path flow estimator as a network observer," *Traffic Engineering and Control*, vol. 39, no. 10, pp. 540–550, 1998.

[7] ——, "Estimating path flows from traffic counts," in *Traffic and Mobility*, H. Wallentowitzm, Ed. Berlin, Germany: Springer Verlag, 1999, pp. 85?–105.

[8] E. Cascetta and M. Postorino, "Fixed point approaches to the estimation of o/d matrices using traffic counts on congested networks," *Transportation Science*, vol. 35, pp. 134–147, 2001.

[9] H. Wang, K. Li, J. Sun, and Y. Liu, "Framework on hierarchical optimization of traffic count location for city traffic system," *Power Electronics and Intelligent Transportation System, Workshop on*, vol. 0, pp. 419–422, 2008.

[10] R. Gabrielli, A. Guidazzi, M. A. Boschetti, V. Maniezzo, and M. Roffilli, "Practical origin-destination traffic flow estimation," in *Proc. ODYSSEUS 2006, Third International Workshop on Freight Transportation and Logistics*, Altea (Spain), 2006.

[11] ——, "Adaptive traffic flow estimation," in *LION 2007 Working Papers, Learning and Intelligent OptimizatioN*, Andalo (Trento) - Italy, 2007.

[12] M. A. Boschetti, V. Maniezzo, and M. Roffilli, "A fully distributed lagrangean solution for a p2p overlay network design problem," *INFORMS Journal on Computing*, 2011, published online in Articles in Advance.

[13] M. A. Boschetti and V. Maniezzo, "Benders decomposition, lagrangean relaxation and metaheuristic design," *Journal of Heuristics*, vol. 15, no. 3, pp. 283–312, 2009.

[14] V. Maniezzo, "A c# wrapper for coinmp," January 2011, http://astarte.csr.unibo.it/coinORwrapper/coinORwrapper.htm.

[15] ESRI, "Arcgis," January 2011, http://www.esri.com/software/arcgis/index.html.

[16] "Postgis," January 2011, http://postgis.refractions.net/.

A Matheuristic Algorithm for Auto-Carrier Transportation

Mauro Dell'Amico * Simone Falavigna * Manuel Iori *

* DISMI, University of Modena and Reggio Emilia
Via Amendola 2, 42122 Reggio Emilia, Italy
{mauro.dellamico, simone.falavigna, manuel.iori}@unimore.it

ABSTRACT

We study a real-world distribution problem arising in the automotive field, in which cars and other vehicles have to be loaded on auto-carriers and then delivered to dealers. The solution of the problem involves both the computation of the routing of the auto-carriers along the road network and the determination of a feasible loading for each auto-carrier. We solve the problem by means of a heuristic algorithm that makes use of simple greedy and local search strategies for the routing part, and more complex mathematical modeling and branch-and-bound techniques for the loading part. Preliminary computational results show that good savings on the total routing distance can be obtained within small computational efforts.

Keywords: Vehicle routing, Matheuristics, Auto-carrier transportation

1. INTRODUCTION

The automotive industry represents a very important sector of modern economies, as confirmed by the weight of turnover in GDP (3.5% in Europe in 2009) and on the number of vehicles that circulate on roads (224 million vehicles in Europe in 2009). One of the main logistic issues in this sector concerns the delivery of vehicles (e.g., cars, vans or trucks) to dealers.

Usually vehicle manufacturers do not deliver their products directly, but rely on special logistic companies. These companies receive the vehicles from the manufacturers, stock them in storage areas and deliver them to the dealers when ordered. The deliveries are provided by special trucks, called auto-carriers, composed by a tractor and perhaps a trailer, both usually equipped with upper and lower loading planes. An example of a typical auto-carrier is depicted in Figure 1. The depicted loading is composed by identical vehicles, but, in most of the cases, loadings involving heterogeneous vehicles occur.

The loading capacity of an auto-carrier strongly depends on the vehicles dimensions and shapes. To increase such capacity auto-carriers are usually equipped with particular loading equipments. For example, vehicles may be partially rotated and the upper loading planes may be translated vertically and/or rotated, see again Figure 1. Both upper and lower planes can also be extended to increase their lengths. Additional loading constraints come from transportation laws, that impose maximum height, length and weight of the cargo. Note that the width is negligible, because vehicles cannot be transported side-by-side on the auto-carriers.

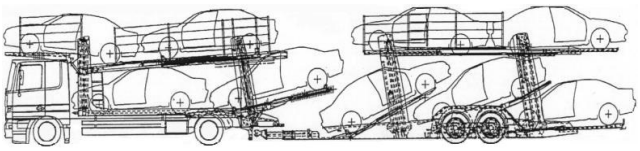


Figure 1: *An example of an auto-carrier with four loading planes, carrying nine vehicles.*

The dealers are spread out over large areas, and it is infrequent that a single dealer order can fill exactly the capacity of one or more auto-carriers. For this reason the companies are forced to load different orders from different dealers into the same auto-carriers. Note also that the auto-carriers are rear-loaded and the loadings must preserve a *Last In First Out* (LIFO) policy: it must always be possible to unload a vehicle at a dealer without moving other vehicles directed to following dealers.

This work is devoted to the study of a real-world case derived from the everyday activity of one of these logistic companies. The company delivers vehicles all over Italy through a large fleet of heterogeneous auto-carriers. Their activity involves multiple days, multiple depots, pickups-and-deliveries, not to mention the uncertainties that typically arise in routing problems. In this work we limit the study to one day (i.e., deliveries cannot be postponed) and one depot (the main depot of the company), and focus on the minimization of the number of kilometers traveled.

Despite these assumptions, the resulting combinatorial problem is very complex, as it requires not only the solution of a two-dimensional non-convex loading problem for each auto-carrier, but also the routing of the auto-carriers along the road network. Both these two sub-problems are NP-hard. Moreover, the size of the problems we address is very large: on average 800 vehicles are delivered everyday to about 200 dealers in the instances that were provided to us. It is thus natural to focus on heuristic techniques.

We developed a constructive heuristic and some local search techniques based on classical ideas from the vehicle routing literature. Any time one of these techniques has to determine the feasibility of the loading associated to a route, it invokes a given loading algorithm. Such algorithm is based on an approximation of the original non-convex two-dimensional loading problem, which is solved by means of 1) an integer linear model or 2) a combinatorial branch-and-bound technique. Our approach can be seen as a particular *matheuristic* algorithm, see Maniezzo et al. [1], because it integrates heuristic search techniques (for the routing) with mathematical modeling and exact techniques (for the loading).

The remaining of the paper is structured as follows. In Section 2 we formally describe the problem and briefly review the relevant literature. In Section 3 we present the approach we developed, and in Section 4 we finally present some preliminary computational results.

2. PROBLEM DESCRIPTION AND LITERATURE REVIEW

In the following we use the term *vehicle* to denote a transported item (e.g., a car, a truck, a van), and the term *auto-carrier* to denote

a truck that transports vehicles. We are given an heterogeneous fleet of auto-carriers. More in details, we are given T auto-carrier types. Each auto-carrier type t has a maximum weight capacity W_t and is formed by P_t *loading planes*. There are K_t auto-carriers available for each type t ($t = 1,\ldots,T$).

We are also given a complete graph $G = (N,E)$, where $N = \{0,1,\ldots,n\}$ is the set of vertices and E the set of edges connecting each vertex pair. Vertex 0 corresponds to the depot, whereas vertices $\{1,\ldots,n\}$ correspond to the n *dealers* to be served. The edge connecting dealers i and j is denoted by (i,j) and has an associated routing cost c_{ij} ($i,j = 0,\ldots,n$). The cost matrix is symmetric and satisfies the triangular inequality.

The demand of dealer i consists of a set of m_i vehicles. Each vehicle k demanded by dealer i has weight w_{ik} ($i = 1,\ldots,n$; $k = 1,\ldots,m_i$), and a particular two-dimensional shape, whose details will be discussed in Section 3.1. The demand of a dealer has to be completely fulfilled. This can be done by using one or more auto-carriers (i.e., *split deliveries* are allowed). Let M denote the total number of vehicles to be transported.

We denote a *route* by the triplet (S,τ,ϕ), where $S \subseteq \{1,\ldots,M\}$ is a set of vehicles to be transported, τ is an auto-carrier type, and $\phi : S \to \mathbb{N}$ is a function that gives the order in which a vehicle is delivered along the route. In particular all vehicles k demanded by the first dealer visited in the route have $\phi(k) = 1$, those demanded by the second dealer visited in the route have $\phi(k) = 2$, and so on ($k = 1,\ldots,|S|$). A route (S,τ,ϕ) is said to be *load-feasible* if

(i) the sum of the weights of the vehicles in S does not exceed the weight capacity of auto-carrier τ;

(ii) there exists a feasible loading of the vehicles in S on the P_τ platforms of auto-carrier τ;

(iii) when visiting the dealer in position $\widetilde{\imath}$ in the route, all vehicles k having $\phi(k) = \widetilde{\imath}$ can be downloaded directly from the auto-carrier, without moving vehicles directed to dealers to be visited later on along the route.

Checking Condition (i) is easy, whereas checking Conditions (ii) and (iii) involves the solution of a complex two-dimensional non-convex loading problem, whose details are shown in Section 3.1.

The *Auto-Carrier Transportation Problem* (A-CTP) calls for the determination of a set of routes such that each route is load-feasible, the demands of the dealers are completely fulfilled and the total cost is minimum.

The (A-CTP) belongs to the class of integrated loading and routing problems. It can be seen as a (particularly difficult) variant of the *Capacitated Vehicle Routing Problem with Two-dimensional Loading Constraints* (2L-CVRP), see Iori et al. [2]. In the 2L-CVRP the demands are sets of two-dimensional rectangular items and have to be loaded into two-dimensional rectangular loading spaces. Apart from the A-CTP, other variants of the 2L-CVRP that model real-world distribution problems have been studied by Gendreau et al. [3] (furniture distribution) and Doerner et al. [4] (timber distribution). We refer the reader to Iori and Martello [5] for a recent survey on routing problems involving loading constraints. For what concerns vehicle routing in general, we refer to the books by Toth and Vigo [6] and Golden et al. [7]. The latter also contains a comprehensive survey (Archetti and Speranza [8]) on routing problems involving split deliveries.

Other auto-carrier problems have been addressed in the literature. Agbegha et al. [9] focused their attention on the loading problem, and modeled it by dividing the auto-carrier into *slots* and assigning vehicles to slots. Incompatibilities arise as some vehicles cannot be assigned to adjacent slots. Tadei et al. [10] studied a large auto-carrier problem by considering both routing and loading aspects. They solved the loading problem by using the concept of *equivalent length* (in practice the length occupied on a plane by a vehicle after an possible rotation). They considered the case of deliveries occurring in multiple days and solved it through a heuristic based on an integer programming formulation.

3. A SOLUTION APPROACH

We developed simple heuristic algorithms based on classical strategies for the capacitated vehicle routing problem. We start with a randomized *closest neighbor* heuristic. We initialize a route by selecting a random vehicle among those to be delivered and a random auto-carrier among the available ones. We then extend the route by selecting the vehicle to be delivered whose dealer is closest to that of the last loaded vehicle. At any iteration we invoke the algorithm to be described below in Section 3.1 to check the feasibility of the loading. We continue extending the current route as long as the loading remains feasible. We then re-iterate by initializing a new route, until all vehicles are loaded.

The solution obtained by the above heuristic is optimized by using three simple local search procedures. The first one, denoted *move 1-0*, attempts to move all the vehicles assigned to a dealer in one route to another route. If the loading is feasible and the total cost of the involved routes is reduced, then the move is performed. The local search re-iterates, in a first-improvement fashion, until no further cost reduction is possible. The two other local search algorithms operate in a similar manner but have larger complexities. Local search *swap 1-1*, resp. *swap 2-1*, attempts to exchange all the vehicles demanded by a dealer, resp. two dealers, in one route with all the vehicles demanded by another dealer in another route. Also the local search procedures invoke the algorithm of Section 3.1 whenever they need to check the feasibility of a loading.

3.1. Solution of the loading problem

In this section we present an algorithm to determine if a given route (S,τ,ϕ) is load-feasible or not. As mentioned before, the exact solution of the two-dimensional non-convex loading problem is NP-hard and particularly complex in practice. Hence we content us with an approximate model of such problem. The reliability of the approximate modeling was tested together with the logistic company, by using their historical delivery database. Out of 20,335 auto-carrier loadings performed by the company (hence feasible), the model reported the correct answer for 20,210 cases, proving to be 99% accurate. Similar results were obtained for loadings that were known to be infeasible. In the following we denote *homogeneous* a loading that involves identical vehicles, and *heterogeneous* one that involves different vehicles.

The first easy check that our algorithm performs is based on the vehicles weights: if their sum is greater than the auto-carrier capacity, then the load is infeasible. Otherwise we perform a second quick check. For each type of vehicle and auto-carrier, the logistic company provided us what they define the *load-index*, i.e., the maximum number of such vehicles that can be loaded on such auto-carrier. For example, the load-index is nine for the vehicle and auto-carrier depicted in Figure 1. We use $d_{ik\tau}$ to denote the load-index, i.e., $d_{ik\tau}$ stands for the maximum number of vehicles having the same shape of vehicle k demanded by dealer i that can be loaded into auto-carrier τ.

Let $i(k)$ denote the dealer demanding vehicle k. We compute $\widetilde{d} = \sum_{k\in S} 1/d_{i(k)k\tau}$ and consider feasible a loading having $\widetilde{d} \leq 1$. Note that the load-index is a very approximate information and heterogeneous loadings may be feasible also when $\widetilde{d} > 1$. For this reason, whenever $1 < \widetilde{d} \leq 1.2$ and the loading is heterogeneous we invoke an integer linear program (ILP) to determine the feasibility. We consider infeasible homogeneous loadings with $\widetilde{d} > 1$ and hetero-

geneous loadings with $\widetilde{d} > 1.2$.

To describe the ILP we need some quite tedious but necessary notation. Each loading plane p of auto-carrier τ has length $L_{p\tau}$ and a possible maximum extension $A_{p\tau}$. Given a plane p, let $h(p)$ denote the plane placed side by side horizontally with p, if any (for example the two lower planes in Figure 1). The total extension of planes p and $h(p)$ is limited to be at most $\widetilde{A}_{ph(p)\tau}$. A vehicle has a certain length, and, whenever loaded on a plane, can be rotated by a certain degree. We denote $\ell_{kp\tau}$ the equivalent length that takes vehicle k when loaded on plane p of auto-carrier τ.

Similarly to what done for $h(p)$, let us denote $v(p)$ the plane placed vertically above/below plane p, if any (for example the upper and lower planes of the trailer depicted in Figure 1). A vehicle being particularly high when loaded on p may have a side effect on $v(p)$. For example we might be forced to lower down completely an upper plane, hence using also completely the length of the lower plane below, or we might be forced to rotate consistently the upper plane, losing in this way a portion of the lower plane length. To express this constraint we define $\lambda_{kv(p)\tau}$ the equivalent length on plane p used by vehicle k when loaded on plane $v(p)$ of auto-carrier τ.

We finally define a *precedence matrix* among planes: let b_{pq} take value one if loading a vehicle on plane p forbids unloading a vehicle loaded on plane q, 0 otherwise. When $b_{pq} = 1$ we say that p precedes q. For example, the right lower plane of Figure 1 precedes all other planes, whereas the right upper plane precedes only the left upper plane.

To model the loading problem as an ILP we define $x_{kp} = 1$ if vehicle k is assigned to plane p, 0 otherwise, for $k \in S$, $p = 1,\ldots,P_\tau$. We also define $a_p =$ length extension of plane p, for $p = 1,\ldots,P_\tau$. We obtain:

$$\sum_{p=1}^{P_\tau} x_{kp} = 1 \quad k \in S \tag{1}$$

$$\sum_{k\in S} (\ell_{kp\tau} x_{kp} + \lambda_{kv(p)\tau} x_{kv(p)}) \le L_{p\tau} + a_p \quad p = 1,\ldots,P_\tau \tag{2}$$

$$x_{kp} + x_{lq} \le 1 \quad p,q = 1,\ldots,P_\tau : b_{pq} = 1; \; k,l \in S : \phi(k) > \phi(l) \tag{3}$$

$$a_p + a_{h(p)} \le \widetilde{A}_{ph(p)\tau} \quad p = 1,\ldots,P_\tau \tag{4}$$

$$0 \le a_p \le A_{p,\tau} \quad p = 1,\ldots,P_\tau \tag{5}$$

$$x_{kp} \in \{0,1\} \quad p = 1,\ldots,P_\tau; k \in S \tag{6}$$

Constraints (1) impose that each vehicle is loaded on a plane. Constraints (2) model the maximum length of a plane, but also taking into account vertical effects. Constraints (3) impose the LIFO policy. Note that we suppose that vehicles having different order of visit and being assigned to the same plane can be loaded in such a way that the LIFO policy be satisfied. Constraints (4) model the limit on the maximum extension of two planes placed side by side, and constraint (5) give the appropriate range to the planes extensions. If model (1)–(6) produces a feasible solution, then we consider the route load-feasible.

We also developed an alternative strategy to the above model based on an enumeration tree. At each level of the tree we create a node by loading any still unloaded vehicle in any plane. For any plane we keep in memory the available residual lengths. For any dealer we keep in memory both the length that still has to be loaded, and the total residual available length in the auto-carrier that can be used by this dealer. When loading a vehicle in a plane, i.e., when creating a node, we update all residual lengths by considering LIFO policy, horizontal and vertical relations among platforms, if any, and maximum extensions. Whenever the residual available length exceeds the length that still has to be loaded for a dealer, we fathom the node. The tree is explored in a depth-first fashion. In Section 4 we compare the performance of this algorithm, denoted *branch-and-bound*, with that of the ILP model.

4. PRELIMINARY COMPUTATIONAL RESULTS

We coded our algorithms in C++ and run them on a Pentium Dual-Core, with 2.70 Ghz and 1.96 GB RAM, running under Windows XP. We tested the algorithms on instances derived from the real-world problem. We considered the daily distributions operated by the logistic company in the month of July 2009, obtaining in total 23 instances, one for each working day. We filled the cost matrix by computing the distances of the shortest paths, in kilometers, using a GIS-based software. The fleet we consider is made by two types of auto-carriers, one with two loading planes and the other with four.

The results we obtained are reported in Table 1. In the left part of the table, columns *n* and *M* report, respectively, the number of dealers and the number of vehicles to be delivered. The smallest instance has 96 dealers requests, for a total of 272 vehicles to be delivered. The largest instance requires instead the delivery of 1139 vehicles.

We run our algorithms by making use of the two options that we developed for the solution of the loading problem (see Section 3.1). The results that we obtained using the branch-and-bound are reported in the middle part of the table. For the starting heuristic algorithm and for the following local search methods, we present the objective function value of the best solution obtained, in column *km*, and the CPU seconds required by the algorithm, in column *sec*. The algorithms are run in sequence, starting from the closest neighbor heuristic and ending with the *Swap (2-1)*. Each algorithm starts from the best solution obtained by the previous one. In the *overall* columns we report the total CPU time required by all algorithms (sec_{tot}) and the time spent by the loading procedure (sec_{load}). Note that sec_{load} is a portion of sec_{tot}. The results that we obtained using the mathematical model are reported in the right part of the table. We only report, for comparison sake, sec_{tot} and sec_{load}. The model has been solved using Cplex 11.

All algorithms using the branch-and-bound option are very fast. Their execution requires 1.5 seconds, on average, and about 7 seconds in the worst case. About 70% of the cpu time used by the algorithms is spent in the execution of the loading procedure. In this case too, as in other routing and loading problems, the loading problem has a crucial effect on the overall problem. The three local search procedures are effective in reducing the number of kilometers traveled. The percentage reduction in the number of kilometers traveled is consistent for *move 1-0* (3.11% with respect to the solution provided by the greedy) and for *swap 1-1* (3.92% with respect to the solution provided by *move 1-0*), but less significative for *swap 2-1* (just 0.64% with respect to *swap 1-1*). The use of model (1)–(6) instead of the branch-and-bound leads to a consistent increase in the CPU times. The seconds dedicated to the computation of the loadings raise from 1.06 to 15.32, on average. We can conclude that the branch-and-bound is a more suitable solution method for these instances.

The results show that good savings on the number of kilometers traveled can be obtained within limited computational effort. On average we are able to reduce by 7.4% the number of kilometers that were traveled in the routes carried out by the company in July 2009. We believe further improvement is possible, and for future research we intend to embed the above local search techniques, and maybe new ones, into a metaheuristic framework.

instance			*branch-and-bound*										*model* (1)–(6)	
			greedy		*move (1-0)*		*swap (1-1)*		*swap (2-1)*		*overall*		*overall*	
day	*n*	*M*	*km*	*sec*	*km*	*sec*	*km*	*sec*	*km*	*sec*	sec_{tot}	sec_{load}	sec_{tot}	sec_{load}
01-Jul	228	832	57,132	0.05	56,184	0.16	54,179	0.92	53,347	0.06	1.19	0.59	14.06	13.35
02-Jul	221	1139	69,999	0.02	68,087	0.50	66,676	0.55	66,550	0.19	1.25	0.59	12.27	11.65
03-Jul	195	737	46,463	0.03	44,540	0.64	43,160	0.28	43,002	0.08	1.03	0.75	7.95	7.55
06-Jul	243	1063	69,135	0.05	65,565	0.47	61,262	1.30	60,968	0.17	1.98	0.94	25.95	24.58
07-Jul	165	629	33,469	0.02	31,362	0.14	30,249	0.30	30,179	0.05	0.50	0.28	7.86	7.55
08-Jul	206	810	52,028	0.05	48,444	0.38	46,417	0.98	46,066	0.13	1.53	0.98	19.91	19.33
09-Jul	200	941	57,682	0.05	56,522	0.77	54,866	1.80	54,538	0.42	3.03	2.57	29.20	28.53
10-Jul	199	803	47,632	0.03	45,187	0.69	44,097	0.25	43,884	0.08	1.05	0.80	10.42	10.08
13-Jul	244	1030	63,989	0.03	62,724	0.72	60,075	1.44	59,906	0.09	2.28	1.30	34.34	33.24
14-Jul	227	826	48,729	0.03	48,281	0.20	46,729	1.26	46,649	0.11	1.61	0.75	20.92	20.22
15-Jul	211	729	53,214	0.03	51,464	1.75	48,830	0.56	47,689	0.22	2.56	2.05	22.11	21.52
16-Jul	206	833	51,402	0.06	50,068	0.28	47,426	1.17	46,988	0.09	1.61	1.16	18.89	18.23
17-Jul	200	801	52,972	0.14	51,517	0.36	48,993	0.36	48,873	0.11	0.97	0.72	6.27	5.92
20-Jul	198	707	37,734	0.03	36,862	0.41	36,195	0.48	35,939	0.08	1.00	0.58	16.28	15.94
21-Jul	209	940	69,137	0.14	68,084	4.78	65,110	1.86	64,906	0.14	6.92	6.07	18.94	17.80
22-Jul	189	614	41,558	0.05	40,661	0.26	39,424	0.39	39,324	0.02	0.72	0.41	7.33	6.97
23-Jul	251	875	58,995	0.02	56,465	0.41	54,628	2.06	54,526	0.13	2.61	1.91	34.37	33.30
24-Jul	198	811	50,619	0.05	49,699	0.24	47,946	0.51	47,651	0.08	0.88	0.31	10.00	9.65
27-Jul	162	552	28,910	<0.01	28,320	0.09	27,407	0.14	27,279	0.03	0.27	0.16	5.05	4.94
28-Jul	176	556	30,479	<0.01	29,421	0.16	28,622	0.17	28,419	0.02	0.34	0.24	5.78	5.67
29-Jul	221	690	44,343	<0.01	43,339	0.36	41,200	0.48	40,652	0.13	0.97	0.50	16.20	15.78
30-Jul	204	614	42,935	<0.01	40,857	0.49	37,745	0.50	37,470	0.09	1.08	0.74	19.31	18.87
31-Jul	96	272	24,195	0.02	23,815	<0.01	23,168	0.06	22,900	<0.01	0.08	0.03	1.80	1.72
average			49,250	0.04	47,716	0.62	45,844	0.77	45,552	0.11	1.54	1.06	15.88	15.32
% *km* reduction					3.11		3.92		0.64					

Table 1: Preliminary computational results.

5. REFERENCES

[1] V. Maniezzo, T. Stutzle, and S. Voss, *Matheuristics: Hybridizing Metaheuristics and Mathematical Programming*, ser. Annals of Information Systems. New York: Springer, 2009, vol. 10.

[2] M. Iori, J. Salazar González, and D. Vigo, "An exact approach for the vehicle routing problem with two-dimensional loading constraints," *Transportation Science*, vol. 41, pp. 253–264, 2007.

[3] M. Gendreau, M. Iori, G. Laporte, and S. Martello, "A tabu search algorithm for a routing and container loading problem," *Transportation Science*, vol. 40, pp. 342–350, 2006.

[4] K. Doerner, G. Fuellerer, M. Gronalt, R. Hartl, and M. Iori, "Metaheuristics for vehicle routing problems with loading constraints," *Networks*, vol. 49, pp. 294–307, 2007.

[5] M. Iori and S. Martello, "Routing problems with loading constraints," *TOP*, vol. 18, pp. 4–27, 2010.

[6] P. Toth and D. Vigo, *The Vehicle Routing Problem*. Philadelphia: SIAM Monographs on Discrete Mathematics and Applications, 2002.

[7] B. Golden, S. Raghavan, and E. Wasil (eds.), *The Vehicle Routing Problem: Latest Advances And New Challenges*, ser. Operations Research/computer Science Interfaces Series. Berlin: Springer, 2008, vol. 43.

[8] C. Archetti and M. Speranza, "The split delivery vehicle routing problem: a survey," in *The Vehicle Routing Problem: Latest Advances and New Challenges*, B. Golden, R. Raghavan, and E. Wasil, Eds. Berlin: Springer, 2008, pp. 103–122.

[9] G. Agbegha, R. Ballou, and K. Mathur, "Optimizing auto-carrier loading," *Transportation Science*, vol. 32, pp. 174–188, 1998.

[10] R. Tadei, G. Perboli, and F. Della Croce, "A heuristic algorithm for the auto-carrier transportation problem," *Transportation Science*, vol. 36, pp. 55–62, 2002.

A New MIP Heuristic Based on Randomized Neighborhood Search

Davide Anghinolfi * Massimo Paolucci *

* Department of Communication, Computer and Systems Sciences
Via Opera Pia 13, Genova
{anghinolfi, paolucci}@dist.unige.it

ABSTRACT

A new simple MIP heuristic, called Randomized Neighborhood Search (RANS) is proposed, whose purpose is to produce within short time bounds high quality solutions especially for large size MIP problems as the ones characterizing real industrial applications. Starting from a feasible incumbent solution, RANS explores a neighborhood randomly defined by calling a MIP solver as a black box tool. RANS rationale is similar to the one of other MIP heuristics recently appeared in literature but, differently, it exploits only a randomization mechanism to guide the MIP solver. RANS has some self-tuning rules so that it needs as single input parameter the maximum computation time. This paper also presents a procedure for generating a first feasible solution based on the same randomization concepts, that can be used as an initialization alternative for particularly hard instances. RANS effectiveness is shown by an experimental comparison with other MIP heuristics.

Keywords: Mixed Integer Programming, MIP heuristics, Neighborhood search

1. INTRODUCTION

Mixed integer programming (MIP) is a flexible method for modeling complex optimization problems, as the ones emerging from many application contexts. A general MIP model (P) can be defined as finding $z = min\{f(x) : Ax = b, x \in S\}$, i.e., minimizing a linear objective function $f : S \rightarrow R$ subject to a set of linear constraints, where the set of decision variables is partitioned in general as $S = B \cup I \cup C$, being B, I and C respectively the sets of binary, integer and real variables. In addition, let denote G the set of general integer variables, i.e. $G = B \cup I$.

MIP belongs to the class of NP-hard problems and many research and practical MIP problems are still very difficult to solve. Therefore, complex combinatorial optimization problems from both academic research and real world applications have been tackled by specialized heuristics or metaheuristics. However, recently, a number of approaches, called *matheuristics*, have been proposed to melt or to associate ideas from metaheuristics with MIP solver algorithms (e.g., [1, 2, 3, 4, 5]).

In this paper a new simple but effective heuristic approach is proposed, which is able to face complex MIP problems exploiting a MIP solver for finding the solution to a sequence of smaller subproblems. The method, called *RAndomized Neighborhood Search* (RANS), iteratively performs local search steps seeking for an improved incumbent solution by calling a MIP solver as a black box exploring device. RANS adopts concepts similar to the *Iterated Greedy* (IG) algorithm proposed in [6] for scheduling problems: IG is a simple algorithm which starts from a feasible incumbent solution and iterates a destruction step followed by a construction step in order to seek for an improved solution. RANS has a self-tuning mechanism to settle the dimension of the MIP subproblems, so that they should be neither too much trivial nor hard to solve. Experimental tests show that this very simple random strategy that uses only hard fixing is quite effective in tackling very tough problems, in particular being able to provide quite good results (i.e., with a reduced gap from the best known solution) in short computation times.

This paper also presents a heuristic method, called *RElaxed RAndomized Neighborhood Search* (RERANS), to find an initial feasible solution for MIP problems that exploits randomization similarly to RANS. The idea is to progressively build the solution solving a sequence of partially relaxed MIP problems where only a subset of randomly chosen variables from G are left integer constrained, whereas the remaining ones are continuous relaxed. Actually, since RERANS needs solving a series of sub-problems, this method is not competitive with respect to other state-of-art general purpose algorithms for fast generating an initial solution, as for example the *Feasibility Pump* (FP) [7]; however, it may be specifically useful whenever MIP solvers or other initialization approaches need a very large time to succeed.

2. LITERATURE REVIEW

MIP heuristic methods recently appeared in literature are *Local Branching* (LB) [1], *Relaxation Induced Neighborhood Search* (RINS) [2], *Evolutionary Algorithm for Polishing* (Polishing) [3] *and Variable Neighborhood Decomposition Search* (VNDS) [4]. These methods generally include a high level component guiding the solution space exploration through the definition of neighborhoods of the incumbent solution, and a low level component responsible of the local search (LS), consisting of the definition of a MIP sub-problem solved by a MIP solver called as a black box module. All the methods need an initial feasible incumbent solution, usually provided as the first feasible solution produced by the MIP solver, and adopt as termination condition the maximum time limit.

LB, originally proposed in [1], is a strategic external branching framework exploiting a MIP solver as black box tactical solution tool. LB was applied to mixed 0-1 integer programming, and suggestions about its extension to general MIP are provided in [2]. The method performs soft variable fixing by means of the so-called *local branching constraints* that impose a bound k (the neighborhood radius) on the maximum Hamming distance of the binary variables from the incumbent x^c, so defining the neighborhood of x^c. Whenever the MIP solver improves the incumbent, the local branching constraint is reversed and the neighborhood of the new incumbent is explored. The method, which is exact in principle, is practically transformed in a LS heuristic having imposed a time limit for the execution of MIP solver; it starts with a given value for the maximum allowed distance k and it both reduces it whenever the MIP solver does not improve the incumbent and increases it during a diversification step. LB was successively reimplemented in [2] as a heuristic to improve the incumbent that is called within the standard branching exploration framework of a

MIP solver whenever a new incumbent is found. The authors in [2] show that the proposed reimplementation outperforms the original method.

RINS [2] is a heuristic to seek for improved incumbent that is called at nodes of a standard branching scheme. The method defines the neighborhood to be explored by performing a set of hard variables fixing, in particular fixing the integer and binary variables that have the same values in the incumbent x^c and in x^r, which is the solution of the linear relaxation of the considered node. When invoked at a node of the branching scheme, RINS does not consider any branching cuts introduced but only global bounds and cuts. An advantage of RINS is its simplicity: it is embedded in MIP solvers so that diversification is implicitly provided by standard branching; it has no distinction between general integer and binary variables; it has no control on the neighborhood dimension. Therefore, being RINS potentially very time consuming, a frequency parameter is used to limit number of nodes where the method is called.

Polishing [3] is a solution improving heuristic that, similarly to RINS, is called at nodes of the MIP solver branch-and-cut exploration tree, but it operates exploiting evolutionary algorithm concepts. Polishing maintains a fixed size population of the best P solutions found so far and when invoked it first generates M mutated solutions and then it performs C solution combinations. Mutation is used to increase both the diversity and the number of the solutions in the population: it is performed first randomly selecting a seed solution and then solving a MIP sub-problem having hard fixed a subset of randomly selected integer variables to the seed values. The fraction of variables to be fixed is initialized to 50% of the total number of variables and successively adapted (increased by 20% if the MIP sub-problem has no solution or no improvement is found; decreased by 25% if only the seed solution is found; unchanged if a new incumbent is found). Combination is performed extending the hard fixing mechanism of RINS: two solutions (or all the solutions) are selected from the population as parents, and a MIP sub-problem is solved having hard fixed the variables whose values agree in the parents. The new solution found is added to the population if better than the worst solution currently included. Similarly to RINS, a node limit L is imposed for sub-problem solution. Other algorithm parameters are the population dimension P, the number M of mutations and the number C of combination performed.

VNDS is a method very recently introduced in [4] that can be considered an evolution of *Variable Neighborhood Search Branching* (VNSB) [8]. Both algorithms differ from the LB and RINS approaches as they do not adopt a branching scheme as high level component but a *Variable Neighborhood Descent* (VND) search strategy which performs a local search by changing the neighborhood structure to avoid to be trapped in local optima. VNDS is a two-level VND scheme. At first level the absolute distances between incumbent and linear relaxation solution components, $\delta_j = \left|x_j^c - x_j^r\right|$ for $j \in B$ (only binary variables were considered in [4]), are computed and sorted in not decreasing way. Then, at second level, the k variables with smaller δ_j are fixed and the consequent sub-problem is solved by a MIP solver. If this improves the incumbent, a VND-MIP step is started, otherwise k is reduced and the process is iterated. The VND-MIP implements a VND where neighborhoods are obtained by LB constraints whose r.h.s. is increased when no improvement is found. VNDS adopts a mix of hard and soft fixing and needs to set a wide number of parameters. Therefore, the method appears more complicated than the ones above outlined also for the need of an appropriate parameter tuning.

3. THE RANS HEURISTIC

The RANS heuristic is a simple iterative search strategy that operates similarly to an iterated local search. The RANS algorithm starts from a first feasible solution x^c for the original MIP problem (P) and iterates the following main steps until the maximum time limit is reached:

1. *Solution destruction*. A subset $F \subseteq G$ of binary and integer variables is randomly selected such that $|F| = k$, where k is a parameter initialized as $k = 0.1 \cdot |G|$ and automatically tuned at each iteration. Then, a partially fixed MIP sub-problem (S) is defined, having fixed the variables $x_j = x_j^c$ for $j \in G \setminus F$ to their value in the incumbent solution.
2. *Solution construction (local search)*. Sub-problem (S) is solved by calling a MIP solver with the current upper bound $f(x^c)$ and the maximum allowed time for solving sub-problems t_{mip} as input parameters. Also the parameter t_{mip} is automatically determined by the algorithm as a function of the time needed to solve the linear relaxation of the original problem (P). If a new best solution is found, the incumbent for the next iteration is updated.
3. *Parameter adjustment and iteration*. If the sub-problem (S) is optimally solved within the available time, then k is increased as $k = k \cdot 1.1$; otherwise k is reduced as $k = k \cdot 0.9$ and a new iteration is started. In this simple way the algorithm adjusts the parameter k, which controls the dimension of the explored neighborhood (that is the number of binary/integer variables in (S)), depending on the experienced difficulty in solving sub-problems. Hence, the choice of the initial value of k is not critical.

It must be noted that, differently from RINS and LB, the proposed method does not operate within any branching framework, but at higher level can be viewed as an iterated LS. In fact, the solution perturbation, that in iterated LS produces a new starting solution, here consists in the definition of a partial solution obtained from a random *destruction*; then the LS, that here is the resolution of a sub-problem, re-constructs a complete solution. RANS neighborhood definition is based only on hard fixing. The neighborhood of the incumbent is randomly defined and its dimension is controlled by k so that the exploration is terminated in reasonable short time. The maximum time for solving sub-problems t_{mip} is determined (in seconds) as $\max\{T_{min}, 3 \cdot t_{rel}\}$, where t_{rel} is the time needed to solve the linear relaxation of (P) and T_{min} is the minimum time allotted to the MIP solver, which can be fixed once for all taking into account of the performances of the used computer and MIP solver. Actually the choice of T_{min} is not critical due to the self-tuning mechanism used for parameter k: anyway T_{min} should be chosen in order to let the MIP solver a sensible minimum time for exploring the branching tree also for problems whose linear relaxation is solved in few seconds. Note that setting a maximum time limit t_{mip} for solving sub-problems is not critical also in case of huge instances, because the auto-tuning of parameter k always allows reducing the neighborhood size so that sub-problems can be optimally solved. After few tests it was fixed $T_{min} = 30$s taking into account the behavior of Cplex solver on some "easy" instances. Note that the self-tuning of k controlling the sub-problem difficulty is similar to the adaptation of the fraction of variables to be hard fixed in Polishing mutation.

Besides the basic behavior described in the above three main steps, a differentiation mechanism is introduced in RANS to reduce the risk of stagnation, that is to remain blocked in a local optimum. It must be observed that, when an incumbent solution is not improved after several iterations, an advantage of the random hard fixing is that it is quite unlikely cycling over the same sub-problems. However, this implicit differentiation may not always be sufficient.

Hence a simple mechanism is devised based on maintaining a pool of solutions, corresponding to the set of last discovered incumbents, and to randomly backtrack to one of them whenever a maximum number of not improving iterations is reached. In particular, the last 10 incumbent solutions are recorded in the pool and the maximum number of not improving iterations is fixed equal to 30 (these latter values were chosen after a few tests). As for RINS in [2], it must be observed that the purpose of the proposed method is to face very difficult MIP problems, finding good solutions in computation times that are acceptable for real world applications. On the other hand, RANS may not be competitive on problems solved without difficulty by standard MIP solvers neither it can be used to prove optimality.

3.1. The initialization method

The RERANS is a method that can be activated to find an initial feasible solution in the cases where the MIP solver or other initialization heuristics are not able to succeed within the allowed time limit. The algorithm iterates the resolution of partially relaxed (R) problems determined from (P) by linearly relaxing all the binary and integer variables in G with the exclusion of a subset $T \subseteq G$ of variables that remain binary/integer constrained (T is initially empty). At each iteration, c variables randomly chosen among the relaxed ones are added to T (c is initialized equal to $0.1 \cdot |G|$) and r binary/integer constrained variables in T are relaxed (r is initialized equal to 0 and it is set to a positive value whenever the MIP solver is not able to find a solution to a sub-problem). The MIP solver is called to provide within t_{mip} the first feasible solution x^0 for problem (R). If it succeeds, then a new partially relaxed problem is defined: first, for each binary/integer constrained variable one deviational constraint

$$x_j - \delta_j^+ + \delta_j^- = x_j^0, \qquad j \in T \tag{1}$$

is added (or possibly updated if already present in the relaxed problem solved in the previous iteration), penalizing the deviational variables δ_j^+ and δ_j^- in the objective function with a large penalty cost. Then, the value of c is updated as $c = 1.2 \cdot c$ and r is reset to 0. When instead the MIP solver is not able to provide a feasible solution for (R) in the given time limit, the algorithm performs a rollback of the previous choices: the last c variables added to T are removed from T and the last r variables removed from T are reinserted in T. Then, the value of c is reduced as $c = 0.8 \cdot c$ and the value of r is set equal to $r = \min\{c, 0.2\,(|T| - c)\}$, so that the number of removed variables is upper bounded by the number of variables binary/integer constrained at the next iteration. The introduction of deviational constraints at an iteration h corresponds to soft fixing the variables that were in T at iteration h-1 so that they are driven towards the values of the feasible solution found at iteration h-1. Differently to hard fixing, this is a mechanism to memorize the feasible integer values found at an iteration for variables in T, without preventing the possibility that the same variables assume different values in the feasible solution generated at the next iteration (and consequently updating the deviational constraints). Similarly to RANS, parameter c is self-tuned in order to adjust the number of variables in T to control the difficulty (i.e., the time needed) to solve the partially relaxed problems. Finally, we adopt in RERANS a random backtracking strategy that is activated whenever no feasible solution is found for a partially relaxed problem within the given time limit. In these cases problem (R) is considered too difficult to solve and then a subset of r variables are removed from T, i.e., are linearly relaxed. Since a well-known difficulty of backtracking in hard fixing is choosing the right variables to unfix, also in this case we believe that a random choice can be a simple and effective general purpose strategy.

4. COMPUTATIONAL RESULTS

The performance of RANS was tested on a collection of 56 benchmark instances which includes the ones referred to in [2] and in [4], plus other instances from MIPLIB [9] selected among the ones optimally solved in more than one hour or still not optimally solved by a commercial solver. The RANS algorithm was implemented in C++ and the tests were performed on a 2.4GHz Intel Core 2 Duo E6600 computer with 4GB RAM, using Cplex 12.2 (configured to use only 1 thread) as general purpose MIP solver. The code of the implemented algorithm can be found at `http://www.discovery.dist.unige.it/rans.cpp`.

As the purpose is to evaluate the effectiveness of the compared methods in producing quality solutions within reasonable short time bounds (so verifying their suitability for industrial applications), a maximum time limit of one hour was fixed. RANS was compared with Cplex and other four methods: LB, RINS, Polishing and VNDS. Similarly to [4] only pure methods were considered, in particular LB, RINS and Polishing implementations directly incorporated within the Cplex branch-and-cut framework (note that for LB this choice corresponds to the re-implementation proposed in [2]). Therefore, the Cplex parameters were set in order to fix the node limit for sub-MIPs to 1000 for LB and RINS, and the RINS frequency to 100. These are the same settings adopted in [2] and [4]. As Polishing is considered a more time-intensive heuristic than the others, in Cplex it is not called throughout branch and cut like other heuristics but invoked only once after at least one feasible solution is available. Therefore, the Cplex parameters were set so that Polishing is invoked after the first feasible solution is found, so imposing operational conditions similar to the ones of RANS and leaving the Polishing evolutionary algorithm exploit at best the available time. The original VNDS code, kindly made available by Authors in [4], was used and two slightly different configurations were tested. The first, labeled VNDS1, corresponds to the second one adopted in [4] (there denoted as "VNDS 2"), and imposes the maximum time for solving sub-problems (t_{sub}) and for the VND-MIP procedure (t_{vnd}) as $t_{sub} = t_{vnd} = 1200$s. The second configuration, labeled VNDS2, was instead characterized by $t_{sub} = t_{vnd} = 300$s.

Being a randomized algorithms, 5 runs were executed for RANS and Polishing for each instance, then computing the average objective value. Similarly to [2], the used performance index was the ratio between the objective value obtained by the different methods and the best known solution, when available, or the best result obtained during these tests. Then, as in [2], the geometric mean (which is less sensitive to outliers) was adopted to perform an aggregate evaluation of the results. Note that for the sake of brevity only aggregate results are here shown. The results were aggregated according to the total number of binary and integer variables, as reported in Table 1. From this table RANS appears the most effective method for the Global group that includes all the instances. Table 1 highlights the aggregate results separating the instances of very small dimension from the others, and further subdividing this latter subset into medium (from 100 to 10.000 binary/integer variables) and large size (more that 10.000 binary/integer variables). Apart for the very small size instance group, in which a depth branching is required to find the optimal solution, the performances of RANS are always the best ones.

The overall behavior of the compared methods is shown in Figure 1 where is depicted the evolution of the geometric mean of objective ratios averaged over the whole benchmark set. Again Figure 1 highlights the effective behavior of RANS in finding good solutions within short time.

Finally, note that only for 3 instances the Cplex solver was not able of finding the initial solution within the t_{mip} bound. In these cases the the starting solution was generated by the RERANS pro-

Num. int. var. (Num. inst.)	RANS	Cplex	RINS	LB	VNDS1	VNDS2	Polishing
Global (56)	**1.45**	3.00	2.93	1.51	4.05	3.82	2.03
<=100 (3)	6.44	6.00	**4.66**	5.00	6.33	6.66	16.0
>100 (53)	**1.17**	2.84	2.83	1.32	3.92	3.66	1.24
100-10000 (36)	**1.15**	1.21	1.19	1.19	1.61	1.57	1.21
>10000 (17)	**1.20**	6.28	6.31	1.58	8.67	7.95	1.30

Table 1: *Aggregated average results*

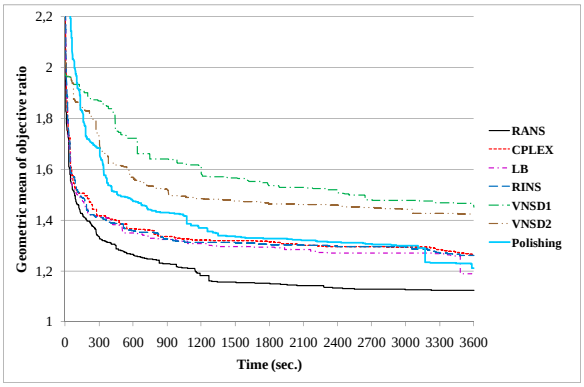


Figure 1: *The evolutions of geometric means of objective ratios.*

cedure. In Table 2 the comparisons between RERANS with Cplex and Cplex with the incorporated FP for the three benchmark instances, i.e. momentum2 (m2), rdrplusc21 (rd21), and van, initialized by RERANS are reported. For this simple test a 3600s time limit was fixed and the algorithm was stopped when the first feasible solution is found. The table shows both time ratios (time for first feasible solution/shortest time among the three methods for first feasible solution) and objective ratios for each instance and method. It can be observed that the time performances of RERANS for these challenging instances were quite good.

	Time ratio			Objective ratio		
	m2	rd21	van	m2	rd21	van
Cplex	7.358	1.000	11.98	1.000	1.000	11.39
Cplex+FP	-	2.425	8.000	-	1.094	11.39
RERANS	1.000	1.096	1.000	1.046	1.027	1.000

Table 2: *RERANS performance results*

5. CONCLUSIONS

This paper proposes RANS, a new heuristic approach to find in reasonably short time high quality solutions to difficult MIP problems. Perhaps the most relevant advantage of RANS is in its conceptual simplicity: the paper shows that the randomization strategy used in RANS is effective with respect to other methods, some of them quite complicated, as highlighted by the comparative experimental campaign performed on a benchmark made of widely referenced instances. Another advantage is that RANS does not need any parameter setting or tuning apart from choosing the maximum available time; this feature is mainly due to the adopted parameter self-tuning mechanism that adapts the neighborhood dimension according to the experimented difficulty in solving the partially fixed MIP problems in the maximum time available.

6. REFERENCES

[1] M. Fischetti and A. Lodi, "Local branching," *Mathematical Programming*, vol. 98, no. 1, pp. 23–47, 2003.

[2] E. Danna, E. Rothberg, and C. L. Pape, "Exploring relaxation induced neighborhoods to improve MIP solutions," *Mathematical Programming*, vol. 102, no. 1, pp. 71–90, 2005.

[3] E. Rothberg, "An evolutionary algorithm for polishing mixed integer programming solutions," *INFORMS J. on Computing*, vol. 19, pp. 534–541, 2007.

[4] J. Lazić, S. Hanafi, N. Mladenović, and D. Urošević, "Variable neighbourhood decomposition search for 0-1 mixed integer programs," *Computers & Operations Research*, vol. 37, no. 6, pp. 1055 – 1067, 2010.

[5] V. Maniezzo, T. Stützle, and S. Voß, *Matheuristics: Hybridizing Metaheuristics and Mathematical Programming*. Springer Publishing Company, 2009, vol. 10.

[6] R. Ruiz and T. Stützle, "A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem," *European Journal of Operational Research*, vol. 177, no. 3, pp. 2033–2049, 2007.

[7] M. Fischetti, F. Glover, and A. Lodi, "The feasibility pump," *Mathematical Programming*, vol. 104, pp. 91–104, 2005.

[8] P. Hansen, N. Mladenović, and D. Urošević, "Variable neighborhood search and local branching," *Computers & Operations Research*, vol. 33, no. 10, pp. 3034 – 3045, 2006.

[9] A. Martin, T. Achterberg, T. Koch, and G. Gamrath, "Miplib 2003," 2010. [Online]. Available: http://miplib.zib.de/

Towards an Ant Colony Optimization algorithm for the Two-Stage Knapsack problem

Stefanie Kosuch *

* Institutionen för datavetenskap (IDA)
Linköpings Universitet, Sweden
stefanie.kosuch@liu.se

ABSTRACT

We propose an Ant-Colony-Optimization algorithm for the Two-Stage Knapsack problem (TSKP) with discretely distributed weights. Three heuristic utility measures are proposed and compared. We argue why for the proposed measures it is more efficient to place pheromone on arcs instead of vertices or edges of the complete search graph. Numerical tests show that the algorithm is able to find near optimal or even optimal solutions after a relatively small number of generated solutions.

Keywords: Two-stage model, Knapsack problem, Ant-Colony optimization, Meta-heuristic, Utility ratio

1. INTRODUCTION

The knapsack problem is a widely studied combinatorial optimization problem. Special interest arises from numerous real life applications for example in logistics, network optimization and scheduling. The basic problem consists in choosing a subset out of a given set of items such that the total weight (or size) of the subset does not exceed a given limit (the capacity of the knapsack) and the total benefit of the subset is maximized. However, most real life problems are non-deterministic in the sense that some of the parameters are not (exactly) known at the moment when the decision has to be made. If randomness occurs in the capacity constraint, the main question that has to be answered is if a violation of the capacity constraint (i.e. an *overload*) could be acceptable. If an overload cannot be permitted in any case, the model maker has two possibilities: Either to force the feasible solutions of the resulting problem to satisfy the capacity constraint in any case. This generally leads to very conservative decisions and the resulting problem might even be infeasible or only have trivial feasible solutions. Or to allow for later corrective decisions at, naturally, additional costs. This latter model is called a *multi-stage* decision model in the literature (for an introduction to stochastic programming models see e.g. [1]).
In this paper we allow the item weights to be random and study a *two-stage* variant of the knapsack problem, denoted $TSKP$ in the remainder. We assume the weight vector to be discretely distributed, i.e. to only admit a finite number of realizations with non-zero probability. In fact, in [2] it has been shown that a stochastic combinatorial optimization problem can, under some mild assumptions, be approximated to any desired precision by replacing the underlying distribution by a finite random sample.
It is well known that in the case of finite weight distributions the $TSKP$ can be equivalently reformulated as a deterministic linear programming problem with binary decision variables (see e.g. [3]). However, the set of constraints and binary decision variables in the reformulation grows with both the number of items as well as the number of scenarios. It is thus typically very large, or even exponential in the number of items. Consequently, solving the deterministic equivalent reformulation of the $TSKP$ to optimality is only possible in very restricted cases. Instead, metaheuristics should be considered in order to obtain near optimal or even optimal solutions in shorter computing time. The aim of this paper is therefore to study some variants of an Ant-Colony-Optimization (ACO) algorithm for the $TSKP$ (for an introduction to ACO-algorithms and standard procedures see [4]).
In the last decade, several metaheuristics for Stochastic Combinatorial Optimization and Integer Programming problems (in the following denoted *SIP*) have been presented. There are two aspects why metaheuristics are important tools to solve *SIP*s: the size of *SIP*s (especially in the case of independently discretely distributed parameters or simply a high number of possible scenarios) and the question of how to evaluate the objective function. In fact, in most cases evaluating the objective function of an *SIP* is NP-hard. In other cases, no deterministic equivalent reformulation is known and only approximate values can be obtained (e.g. using Sample Average Approximation). Both difficulties can be tackled by applying appropriate metaheuristics (see e.g. [5]).
To the best of our knowledge, no special purpose metaheuristic for the $TSKP$ has yet been proposed. Our work is, however, inspired by previous works on ACO-algorithms for the related Multiply Constrained Knapsack problem *MCKP* (see e.g. [6],[7]). We think that an ACO-algorithm is a good choice to solve the $TSKP$ due to the possibility to effectively use utility measures. Moreover, ants are building (new) solutions without needing to evaluate the objective function, which, in the case of the $TSKP$, is an *NP*-hard problem itself. Thus, evaluation needs only to be done in order to compare solutions.

2. MATHEMATICAL FORMULATION, PROPERTIES AND AN APPLICATION

We consider a stochastic knapsack problem of the following form: Given a knapsack with fix weight capacity $c > 0$ as well as a set of n items. Each item has a weight that is not known in the first stage but comes to be known before the second-stage decision has to be made. Therefore, we handle the weights as random variables and assume that the weight-vector $\chi \in \mathbb{R}^n$ is discretely distributed with K possible realizations (or scenarios) $\chi^1, \ldots, \chi^K$. The corresponding, non-zero probabilities are denoted $p^1, \ldots, p^K$. All weights are assumed to be strictly positive.
In the first stage, items can be placed in the knapsack (*first-stage items*). The corresponding first-stage decision vector is $x \in \{0,1\}^n$. Placing item i in the knapsack in the first stage results in a reward $r_i > 0$. At the beginning of the second stage, the weights of all items are revealed. First-stage items can now be removed and additional items be added (*second-stage items*) in order to make the capacity constraint be respected and/or increase the total gain.
If item i is removed, a penalty d_i has to be paid that is naturally strictly greater than the first-stage reward r_i. The removal of item i is modeled by the decision variable y_i^- that is set to 1 if the

item is removed and to 0 otherwise. Similarly, we assume that the second-stage reward for this item $\bar{r}_i > 0$ is strictly smaller than its first-stage reward. If an item is added in the second stage we set the corresponding binary decision variable y_i^+ to 1. The resulting Two-Stage Knapsack problem with discrete weight distributions can be formulated as follows:

Two-Stage Knapsack Problem with discretely distributed weights (*TSKP*)

$$\max_{x\in\{0,1\}^n} \quad \sum_{i=1}^{n} r_i x_i + \sum_{k=1}^{K} p^k \mathscr{Q}(x,\chi^k) \tag{1}$$

$$\text{s.t.} \quad \mathscr{Q}(x,\chi) = \max_{y^+,y^-\in\{0,1\}^n} \sum_{i=1}^{n} \bar{r}_i y_i^+ - \sum_{i=1}^{n} d_i y_i^- \tag{2}$$

$$\text{s.t.} \quad y_i^+ \le 1 - x_i, \quad \forall i = 1,\dots,n, \tag{3}$$

$$y_i^- \le x_i, \quad \forall i = 1,\dots,n, \tag{4}$$

$$\sum_{i=1}^{n} (x_i + y_i^+ - y_i^-)\chi_i \le c. \tag{5}$$

The *TSKP* is a *relatively complete recourse* problem, i.e. for every feasible first-stage decision there exists a feasible second-stage decision. Moreover, given a first-stage decision and a realization of χ, solving the second-stage problem means solving a deterministic knapsack problem. Evaluating the objective function for a given first-stage solution is thus NP-hard.
As a simplified application consider an (online) travel agency that aims to fill the vacant beds (the deterministic capacity) of a hotel complex. Clients are travel groups whose exact number of travelers (the "weight" of the group) is still unknown at the moment the decision which groups to accept has to be made. This randomness can for example be a result of later cancellations. In order to maximize the final occupancy of the beds, the travel agent might allow an overbooking. If, in the end, the number of beds is not sufficient, one or more of the groups need to be relocated in neighboring hotels which leads to a loss of benefit. If beds are left unoccupied, last minute offers at reduced priced might be an option to fill these vacancies. A simple recourse version of this problem with a set of hotel sites has been previously considered in [8].

3. THE ACO-METAHEURISTIC

In the remainder we use the following notations:

- $\mathscr{A}$: set of ants
- t: "time", i.e. passed number of construction steps in current iteration ($t \le n$)
- $S_a(t)$: set of items chosen by ant a after time t
- $\tau_i(t)$: pheromone level on vertex/arc/edge i at time t
- η_i: utility ratio of item i
- ν_i: non-utility ratio of item i
- $\rho \in (0,1)$: global evaporation parameter
- $\rho_{loc} \in (0,1)$: local evaporation parameter
- $p_{ij}^a(t)$: transition probability = probability for ant a to go from vertex i to vertex j at time t

The basic structure of the ACO-algorithm for the *TSKP* is given in Algorithm 3.1. Its functioning is detailed in the following subsection. The *Transition of ants* step consists of the transition of the ants following the transition probabilities and the update of $S_a(t)$.

$IT \leftarrow 0$
while $IT < IT_{MAX}$ **do**
  $IT \leftarrow IT + 1$
  Initialization
  $t \leftarrow 0$
  while $t < n$ and $(\exists a \in \mathscr{A}\colon (n{+}1) \notin S_a(t-1))$ **do**
    $t \leftarrow t + 1$
    Compute transition probability
    Transition of ants
    Local pheromone update
  end while
  Global pheromone update
end while
return Best found solution

Algorithm 3.1: ACO-algorithm for the *TSKP*

3.1. The Complete Search Graph

Our search graph is based on the search graph proposed for the *MCKP* in [6], i.e. on a complete graph whose n vertices represent the n items. Note that the ants only construct the first-stage solution (solution vector x). In order to model the randomness of the first item chosen by an ant, we add an additional vertex 0 to the complete graph that is connected to all the other n vertices, with $p_{i0}^a(t) = 0$ for all $a \in \mathscr{A}$ and $t > 0$. Initially, all ants are placed on this vertex. We denote this vertex as *starting vertex*.
In the case of the *MCKP* one has a natural certificate of when an ant has come to an end of its solution construction: when either all items have been chosen or when adding any of the remaining items would lead to the violation of at least one of the constraints. As for the *TSKP* even adding all items in the first stage would yield a feasible solution, we add a *termination vertex* $n+1$ which is connected to all vertices, including the starting vertex.

3.2. Pheromone trails and update procedure

Several choices could be made for the way pheromone is laid by the ants (see [7]). In the simplest setting, the search graph is non-directed and pheromone is laid on *vertices*, i.e. items that are included in the best solutions found so far have a high level of pheromone. In the second variant, pheromone is placed on *edges* of the non-directed search graph, or, equivalently, pairs of items. In this setting the probability that an ant chooses a specific item at time t increases with the number of (good) previously found solutions that contain both this specific item as well as the item the ant has chosen at time $t-1$. In the third variant the graph is assumed to be a complete directed graph and pheromone is laid on *arcs*, i.e. directed edges. Contrary to the two former settings, this setting not only takes into account which items (or item pairs) had been added to former good solutions, but also in which order. In the following, when talking of an *element*, this refers to either a vertex, edge or arc of the search graph.
We use a local as well as a global update procedure (see e.g. [6]). The local update procedure is performed after every construction step. The pheromone level on the elements chosen during this step by an ant is slightly reduced, in order to diversify the produced solutions. For an element i the local update rule is as follows:

$$\tau_i \leftarrow (1 - \rho_{loc}) \cdot \tau_i + \rho_{loc}\tau_{min} \tag{6}$$

ρ_{loc} is the local evaporation parameter: The larger ρ_{loc}, the higher the evaporation and thus the higher the decrease of pheromone on the chosen elements. τ_{min} is a lower bound for the pheromone level.
The global update procedure is done once all ants have constructed their solutions. The evaporation of pheromone on all arcs is the

first part of the global updating:

$$\tau_i \leftarrow (1-\rho)\cdot\tau_i \tag{7}$$

where ρ is the global evaporation parameter.
In the second part of the global update procedure only the best found solutions are considered and the pheromone level on these solutions is intensified. In our setting we intensify the pheromone level on an element if and only if the element has been chosen in either the best solution found so far or in one of the λ best solutions found in the last iteration:

$$\tau_i \leftarrow \rho \tag{8}$$

Note that the maximum pheromone level is 1. If due to the update procedures the pheromone level on an element falls below a lower bound τ_0, it is set to τ_0.
In the case of pheromone on arcs we additionally diversify the solutions by storing the best solution as a set of items. The pheromone is then increased on all arcs that lead to one of these vertices.

3.3. Heuristic utility measures

An advantage of the *TSKP* compared to the *MCKP* is that we have a clearly defined "relevance factor" for each knapsack constraint: the probability of the corresponding scenario (see [9] for more information on utility measures for the *MCKP*). Our idea is thus to compute the overall utility ratio of an item as an average over the utility ratios of those scenarios where the item still fits the capacity. The problem is, however, that, once adding an item would lead to a violation of the capacity in one or more scenarios, deciding whether it is more profitable to remove an item and add the new one, or to discard the current item, is NP-hard. We overcome this problem by relying on the chosen utility measure: If the utility measure is chosen wisely, one might get good solutions by always discarding the current item (in the case of an overload).
While in the case of the *MCKP* two factors have to be considered (reward and used capacity), there are 2 more factors that play a role for the utility of an item in the two-stage setting: the second-stage reward and the second-stage penalty. This makes the definition of a good utility measure much more complex.
The utility measure for the termination vertex should depend on the penalty we would have to pay in the second stage if we add another item or the reward we could gain in the second-stage if we do not add any of the remaining items. We thus compute an additional "non-utility" ratio v_i for each item i. The utility ratio of the termination vertex is then defined as the minimum over these ratios: If for all items the non-utility ratio is high, termination might be the best choice.
We propose three different choices for the (non-)utility ratios. These are calculated with respect to the set $\mathscr{K}$ of scenarios where the respective item still fits in the knapsack.

Simple measure: Here we define the utility of an item to be the "average" ratio of first-stage reward and weight.

$$\eta_i^S = \sum_{k\in\mathscr{K}} p^k \frac{r_i}{\chi_i^k} \tag{9}$$

Note that this measure is not the exact mean of the reward-weight ratios over the scenarios where the item still fits as $\sum_{k\in\mathscr{K}} p^k < 1$ is possible. The exact mean would be obtained by dividing η_i^S by $\sum_{k\in\mathscr{K}} p^k$. The utility ratios do thus also depend on the probability that item i still fits the capacity (given by $\sum_{k\in\mathscr{K}} p^k$).
We define two non-utility measures. For half of the ants the first measure is applied and for the other half the second. The first non-utility ratio is defined to be the "average" ratio of second-stage penalty and weight over the instances where the item does not fit in the knapsack any more. Contrary to the utility ratios, these first non-utility ratios increase with $\sum_{k\notin\mathscr{K}} p^k$. The second non-utility ratio equals the reward we would gain on average in the second stage if we do not add the item and assume that it can be added in any scenario in the second stage.

$$v_i^S = \sum_{k\notin\mathscr{K}} p^k \frac{d_i}{\chi_i^k} \qquad v_i^S = \sum_{k=1}^{K} p^k \frac{\bar{r}_i}{\chi_i^k} \tag{10}$$

Difference Measure: We compare what we would gain by adding an item in the first and not the second stage ($r_i - \bar{r}_i$) with what we would loose if we would have to remove the item in the second stage ($d_i - r_i$):

$$\eta_i^D = \sum_{k\in\mathscr{K}} p^k \frac{r_i-\bar{r}_i}{\chi_i^k} \qquad v_i^D = \sum_{k\notin\mathscr{K}} p^k \frac{d_i-r_i}{\chi_i^k} \tag{11}$$

Ratio measure: Instead of differences we consider ratios:

$$\eta_i^R = \sum_{k\in\mathscr{K}} p^k \frac{r_i/\bar{r}_i}{\chi_i^k} \qquad v_i^R = \sum_{k\notin\mathscr{K}} p^k \frac{d_i/r_i}{\chi_i^k} \tag{12}$$

3.4. Transition probabilities

In this study we only consider the most traditional way of computing the transition probabilities from the pheromone level and utility ratio (see e.g. [4]): For a vertex $v \in \{1,\ldots,n+1\}$, the probability that an ant a currently sitting on vertex u moves to v is computed as follows:

$$\pi(u,v,S_a(t-1),\tau) = \frac{\tau_{i(u,v)}^{\alpha}(t)\,\eta_v^{\beta}(S_a(t-1))}{\sum_{w=1}^{n}\tau_{i(u,w)}^{\alpha}(t)\,\eta_w^{\beta}(S_a(t-1))} \tag{13}$$

Here α and β are two parameters that control the relative importance of pheromone level and utility ratio and $i(u,v) = v$ (vertex pheromone) or $i(u,v) = (u,v)$ (arc or edge pheromone). In the first iteration we only take the utility ratio into account. As a consequence, the pheromone level on the elements is initialized during the first global update procedure.

4. SUMMARY OF THE OBSERVATIONS MADE DURING THE NUMERICAL TESTS

4.1. Comparison of the 3 different variants to lay pheromone trails

During our tests we observed that, when pheromone is placed on vertices (or edges), the ants had difficulties to reproduce the best solution found so far and to search in its local neighborhood (even with $\lambda = 0$). As a consequence, the solution value of the best solution produced during an iteration was mostly strictly smaller than that of the the current best solution. This caused severe problems for the convergence of our ACO-algorithm. In contrast, when pheromone is laid on arcs, the quality of the best solution produced during one single iteration generally increased monotonically (however not strictly). These observations seem to be contradictory to what has been observed in previous studies of ACO-problems for the *MCKP* (see [6]). It can, however, be explained by the fact that our utility measure relies on the order in which the items have been added. More precisely, the set of items that are still allowed to be chosen depend heavily on the set of previously added items.

4.2. Comparison of the 3 different utility measures

For a representative comparison of the convergence behavior of our ACO-algorithm using the three different measures see Figure

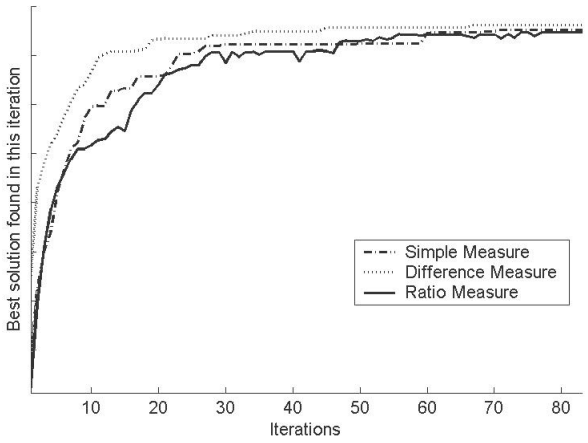


Figure 1: Representative convergence behavior using different utility measures

1 (test with pheromone on arcs). Our numerical tests on the chosen test instances showed that the difference measure seems to be better suited than the two other measures: Using the difference measure our algorithm found the optimal solution in around 16% of the tests while the other two measures were only rarely (on some instances never) able to produce optimal solutions. Concerning the runs where the optimal solution was not found the average (maximum) relative gap was of 0.03% (0.06%) for the difference measure versus 0.09% and 0.1% (0.18% and 0.19%) for the simple and ratio measure. The differences in the solution qualities are on the one hand due to the initial iteration where the ants find much better solutions based on the difference measure heuristic than based on one of the other two heuristics. On the other hand, the algorithm converges much faster to near optimal solutions in the former case and the quality of the best solution produced per iteration never decreases even when the best found solution is already close to the optimum.

5. FUTURE WORK

In case of instances with a high number of scenarios sampling should be considered. This means that at each iteration a set of scenarios is sampled whose cardinality is smaller than K. By increasing the sample size during the iterations convergence might be achieved. Moreover, one obtains a natural additional diversification of the produced solutions (see [5] for more details).

In order to evaluate the second-stage expectation for a given found first-stage solution we solved the K second-stage knapsack problems independently using an optimal knapsack algorithm from the literature. If needed, the CPU-time could be decreased by instead using an $FPTAS$. By increasing the performance ratio of the used approximation algorithm during the iterations, convergence might once more be achieved.
Last but not least, to fully evaluate the competitiveness of an ACO-approach to solve the $TSKP$ a comparison with other metaheuristics is clearly needed.

6. REFERENCES

[1] A. Shapiro, D. Dentcheva, and A. Ruszczynski, "Lectures on stochastic programming: Modeling and theory," in *MPS/SIAM Series on Optimization*. SIAM-Society for Industrial and Applied Mathematics, 2009, vol. 9.

[2] A. J. Kleywegt, A. Shapiro, and T. Homem-de-Mello, "The sample average approximation method for stochastic discrete optimization," *SIAM Journal on Optimization*, vol. 12, no. 2, pp. 479–502, 2002.

[3] A. A. Gaivoronski, A. Lisser, R. Lopez, and X. Hu, "Knapsack problem with probability constraints," *Journal of Global Optimization (Online First)*, 2010.

[4] V. Maniezzo, L. M. Gambardella, and F. de Luigi, *Ant Colony Optimization*. Springer Berlin / Heidelberg, 2004, ch. 5, pp. 101–117.

[5] L. Bianchi, M. Dorigo, L. M. Gambardella, and W. J. Gutjahr, "A survey on metaheuristics for stochastic combinatorial optimization," *Natural Computing: an international journal*, vol. 8, pp. 239–287, 2009.

[6] S. Fidanova, "Ant colony optimization for multiple knapsack problem and model bias," in *Numerical Analysis and Its Applications*, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2005, vol. 3401, pp. 280–287.

[7] L. Ke, Z. Feng, Z. Ren, and X. Wei, "An ant colony optimization approach for the multidimensional knapsack problem," *Journal of Heuristics*, vol. 16, pp. 65–83, 2010.

[8] T. Benoist, E. Bourreau, and B. Rottembourg, "Towards stochastic constraint programming: A study of online multi-choice knapsack with deadlines," in *Proceedings of the CP '01*. Springer London, 2001, pp. 61–76.

[9] H. Kellerer, U. Pferschy, and D. Pisinger, *Knapsack Problems*. Springer Berlin / Heidelberg, 2004.

Optimal Parts Allocation for Structural Systems via Improved Initial Solution Generation

Yang Zhang * Horst Baier *

* Institute of Lightweight Structures, TU München
München, Germany
{zhang, baier}@llb.mw.tum.de

ABSTRACT

In a mechanical structure, it is often the case that many of the parts are nominally identical. But actually they always differ slightly in physical and geometrical properties due to variation of material and manufacturing error. Parts allocation for a structural system aims at optimizing performance of the manufactured structure by assigning each of these parts to a proper position in the structure during the assembling period. In this paper, the parts allocation problem is addressed and the formulation of it as a nonlinear assignment problem (NAP) is presented. A method is developed to generate an initial solution for it. The technique is tested on benchmark examples. All the results show that it could always construct a high quality starting point from both view of objective and constraint violation. Compared to starting with the identity permutation and randomly generated ones, the standard 2-exchange local search algorithm starting with initial solutions generated by this method well solves most of the test problems in the meantime with a large reduction in total number of function evaluations.

Keywords: Initial solution, Nonlinear assignment problem, Local search, Parts allocation

1. INTRODUCTION

During structural manufacturing, we often need to assemble parts together to create a whole structure. Many of the parts are designed to be identical and could be swapped with each other without influence on characteristics of the assembled structure. But due to variation of material and manufacturing errors, parts that have been manufactured are always slightly different in some properties from each other. The parts allocation problem for a structural system is that, we want to find out how to allocate each of the parts at hand to the structure so that the assembled one could reach a best mechanical performance, such as minimum deflection at some point under certain loads and certain constraints.

There is a significant feature of this kind of problem, that each evaluation of a solution requires normally time-consuming computation, e.g. finite element analysis. For a large scale problem, each such analysis could lasts minutes even hours. Therefore, an applicable algorithm need not return the global optimum, but instead it has to be able to return a good enough solution with as few number of function evaluations as possible.

In this paper, the parts allocation problem for structural systems is formulated as a nonlinear assignment problem. Assignment problem (AP) is a type of problem in combinatorial optimization, which aims at finding a way to assign n items to n other items to obtain the minimum of a defined objective. There are many polynomial-time algorithms have been developed for linear assignment problem (LAP), such as Munkres (Hungarian) algorithm, shortest path algorithms and auction algorithms [1]. Well-known nonlinear assignment problems are quadratic assignment problem (QAP) and 3-index assignment problem (3AP), which have been shown that both are NP-hard problems [2, 3]. For even more general NAPs, so far, heuristic algorithms are widely studied and applied to find good quality solutions [4, 5].

A high quality initial solution is essential for any heuristic algorithm, which could reduce the total number of function evaluations while returning a same quality solution. There are several ways to construct initial solutions, for instance, by taking the identity permutation, a randomly generated permutation, or a heuristically determined starting point [4]. For the first two methods, they don't include any consideration of a specific problem, so there is no reason to take them as a good starting point.

The outline of this paper is as follows: in Section 2, we present the formulation of parts allocation problem for structural systems as a NAP. In Section 3, a procedure to generate an initial solution for the problem is defined. We apply the technique to some benchmark examples and present the test results in Section 4. Finally we reach the conclusion.

2. MATHEMATICAL FORMULATION OF THE PROBLEM

In this study, we assume the properties of each part that have been manufactured are measurable and are known. And we take the difference in properties of area of cross-section (A), Young's Modulus (E) and coefficient of thermal expansion (CTE) into account.

Consider we have n exchangeable parts have been manufactured and are to be assembled into n different positions of a structural system. The objective is to minimize the displacement at certain point or the maximum stress in the assembled structure under certain loads. We number the n positions and denote the properties of parts assigned to each position $\{A(i),E(i),CTE(i)\}$, i=1, 2, ..., n. We also number the parts at hand by 1,2,...n, and each with a property set $\{A_j,E_j,CTE_j\}$, j=1, 2, ..., n. To evaluate the displacement of the structure under certain loads, we usually need to perform a finite element analysis, which mainly solves a large system of linear equations as follows:

$$\mathbf{KU} = \mathbf{F} \tag{1}$$

where **K** is the master stiffness matrix that is dependent on properties A and E of parts at each position, **F** is the load vector which is dependent on CTEs, and **U** is the displacement vector to be computed.

We represent the assignment with a permutation matrix $\mathbf{X} = (x_{ij})_{n\times n}$, which satisfies following assignment constraints:

$$\sum_{i=1}^{n} x_{ij} = 1, \quad j = 1,2,...,n, \tag{2}$$

$$\sum_{j=1}^{n} x_{ij} = 1, \quad i = 1,2,...,n, \tag{3}$$

$$x_{ij} \in \{0,1\}, \quad i,j = 1,2,...,n, \tag{4}$$

and

$$x_{ij} = \begin{cases} 1 & \text{iff } j^{th} \textit{ part is allocated to position } i, \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

Thus the areas of cross-section at each position could be interpolated with following equation:

$$[A(1),A(2),\ldots,A(n)]^T = \mathbf{X}[A_1,A_2,\ldots,A_n]^T \tag{6}$$

Similar interpolation schemes are performed for E and CTE. With these interpolation formulas, the stiffness matrix and the load vector are both formulated as a function of entries in the permutation matrix **X**. Therefore, the unknown displacement components are normally highly nonlinear functions of $\{x_{ij}\}$. Further, the response of stresses in the structure that can be derived from **U**, are also nonlinear functions of $\{x_{ij}\}$. Finally, we formulated the parts allocation problems as a nonlinear assignment problem.

3. PROCEDURE FOR GENERATING AN INITIAL SOLUTION

Through interpolation equation (6), it could be seen that properties at each position are continuous functions of **X** if we make a continuous relaxation of the binary constraints on each x_{ij}. Therefore, displacements and stresses are also derived to be continuous functions of **X**. This continuity makes it mathematically meaningful to evaluate objective at points where entries of **X** lies between 0 and 1. Based on this fact, we designed a 3-step deterministic way to generate an initial solution for a parts allocation problem of size n:

Step1. Construct the matrix $\mathbf{X}^S = (x_{ij}^s)_{n\times n}$, where all the entries equals to 1/n. And evaluate the objective $f^S = f(\mathbf{X}^S)$.

Step2. Compute $c_{ij} = \partial f/\partial x_{ij}$ at $\mathbf{X}^S$, for i, j = 1,2,...,n.

Step3. Construct cost matrix $\mathbf{C} = (c_{ij})_{n\times n}$, and solve the linear assignment problem min $\sum_{i,j=1}^{n} c_{ij}x_{ij}^0$, where $\mathbf{X}^0 = (x_{ij}^0)_{n\times n}$ satisfies all the assignment constraints from equation (2) to equation (4).

We artificially create matrix $\mathbf{X}^S$ in Step1, which assign all the entries the same value so as to avoid bias of any specific possible solution. In Step2, we use finite difference method to evaluate the partial derivatives of f: set stepsize ε be a small positive value, then $c_{ij} \approx (f(\mathbf{X}^S + \Delta_{ij}) - f^S)/\varepsilon$, where Δ_{ij} is a $n \times n$ matrix with all the entries equal to zero except the one in position (i,j) equals to ε. The solution $\mathbf{X}^0$ in Step3 is just the initial solution we generated.

The procedure could be seen as making a linearization of the objective function around $\mathbf{X}^S$ and then finding the point that reduce the objective most with deepest descent method. Thus, if the problem is originally a LAP, then the initial solution we generated is exactly the optimal solution for the problem. For nonlinear assignment problems we could also expect to reach a good quality solution after Step3 if the derivatives of objective with respect to $\{x_{ij}\}$ do not change largely at different points.

The number of function evaluations we need to construct the initial point is n^2+1. It could be further reduce to n^2 if we simply assume f^S in Step1 is 0, which wouldn't influence the result in Step3 but reduce number of function evaluations by one.

4. EXAMPLES AND COMPUTATIONAL RESULTS

To show the quality of the initial solution generated by above method, we tested on several benchmark examples.

4.1. 10-Bar Truss Allocation Problem

We tested our method first with a 2D 10-bar truss structure shown in Figure 1. All the bars in the structure are designed to have the same length of 1000mm, the same circular cross-section of area A = $1000mm^2$ and use the same material with Young's modulus E = 68.95GPa, CTE = $23.6 \times 10^{-6}/^{\circ}C$. Thus all of them could be swapped with each other. Now assume we have manufactured ten bars to be allocated into the ten positions of the structure, and due to manufacturing errors, the properties A, E and CTE of each bar are different to design slightly. The objective is to find an allocation of the bars to minimize the displacement of node 1 under both a uniform thermal load of $\Delta T = 42.37^{\circ}C$ on the structure and a downward force of 29.4kN at node 1.

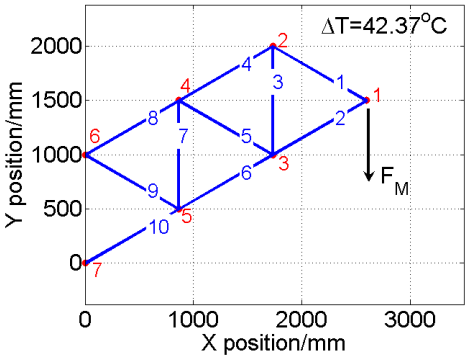


Figure 1: *10-bar truss structure under loads.*

We tested with three different situations where all the properties for each bar are manufactured with maximum error of 5%, 10% and 50% respectively. And for each error level, we randomly generated 10 instances from a uniform distribution. The stepsize ε used in Step2 is 10^{-3}. Munkres algorithm [6] is applied to solve the derived LAP in Step3.

For each instance, we compute relative error of objective of the initial solution with respect to that of the global optimum, which is found by enumerating all the possible permutations with total number of $10! \approx 3.6 \times 10^6$. The average relative errors are 0.00%, 0.01% and 0.98% for error level of 5%, 10% and 50% respectively. For lower error level, the properties of bars are less different. Therefore the change of the derivatives of objective with different allocations is less, which leads to higher quality initial solutions obtained through our method.

After generation of the initial solution, we use a standard 2-exchange local search algorithm starting with it to solve the problem (LS-Our). We compared the results with other two methods: one is using the same algorithm but starting always from the identity permutation (LS-Id); the other one is using the same algorithm but starting from a randomly generated initial solution (LS-Random). To reduce the occasionality of this method, we randomly generate 100 initial points for each instance and take the average performance to compare with others.

The statistical results of the 30 instances are listed in Table 1, where we use following notations: e_{ini} is the average relative error of the objective of initial solutions with respect to that of the global optimum. e_{final} is the average relative error of the objective of final solutions. p_{succ} is the percentage of successful runs, in which

the relative error of the final solution is less than 1%. n_{ite} is the average number of iterations and n_{func} is the average total number of function evaluations.

Method	e_{ini}	e_{final}	p_{succ}	n_{ite}	n_{func}
LS-Id	41.4%	0.28%	93%	8.5	384
LS-Random	46.9%	0.22 %	94%	8.0	363
LS-Our	0.33%	0.00%	100%	3.1	242

Table 1: Statistical results with different initial solution.

It could be seen that our procedure could generate quite high quality initial solutions and increase the ability of the algorithm to achieve successful solutions. Meanwhile, the average number of iterations and number of function evaluations is largely reduced though it requires n^2 times function evaluations at the beginning.

4.2. 25-Bar Truss Allocation Problem

4.2.1. Case without constraints

In practice, it is always the case that not all of the parts are designed to be the same and could be swapped with each other. However, we could usually divide all of the parts into several groups according to their geometry, so that parts in the same group could be exchanged. For this multiple groups problem, when constructing the initial solution, we simply treat each group independently by fixing the entries in permutation matrix of other groups to be $1/n_g$, where n_g denote the size of the corresponding group.

We tested this kind of problem with a 3D 25-bar truss structure presented in [7]. All the 25 bars are divided into 8 groups, and each group has 1,4,4,2,2,4,4,4 bars respectively as colored in Figure 2. Bars of the same group could be exchanged with each other and they differ in E, CTE and A. The values of these properties are designed to be identical as in Section 4.1. Our goal is to minimize the displacement of node 1 under a uniform thermal load of $\Delta T = 42.37^{\circ}C$ and some mechanical forces.

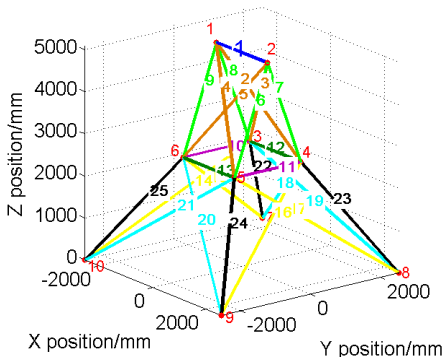


Figure 2: *25-bar truss structure.*

We applied three different load cases onto the structure, where the mechanical forces are different as listed in Table 2. We randomly generated 10 instances with manufacturing error of 5% for each load case. Statistical results are presented in Table 3.

The global optimum are still found by enumerating all the possible permutations with total number of approximately 3.2×10^7. The average iteration needed by the algorithm starting from the generated initial solution is close to 1, which means the procedure is able to find an initial solution very close to the global optimum.

Load case	Nodes	Loads		
		Fx/kN	Fy/kN	Fz/kN
1	1	4.45	-44.5	-44.5
	2	0	-44.5	-44.5
	3	2.22	0	0
	6	2.67	0	0
2	1	0	89.0	-22.2
	2	0	-89.0	-22.2
3	1	4.45	44.5	-22.2
	2	0	44.5	-22.2
	3	2.22	0	0
	6	2.22	0	0

Table 2: Load cases for 25-bar truss structure.

Method	e_{ini}	e_{final}	p_{succ}	n_{ite}	n_{func}
LS-Id	5.25%	0.01%	100%	12.7	406
LS-Random	4.93%	0.01%	100%	12.0	383
LS-Our	0.00%	0.00%	100%	1.2	128

Table 3: Statistical results with different initial solution.

4.2.2. Case with stress constraints

Except the goal to minimize the objective, mechanical structures are always required to fulfil some constraints, typically like limitation of maximum stress. We further add a stress constraint to above problem:

$$\sigma_{\max}/\sigma_A - 1 \leq 0 \tag{7}$$

where $\sigma_{\max}$ is the maximum stress in the structure, σ_A is the allowable stress. In our problem, σ_A is selected to be the maximum stress when bars are all manufactured without error. And the objective is still to minimize the displacement of node 1 under different loads.

We use penalty method to deal with constraints. Denote t equals to the left hand side of the constraint equation (7), and introduce following penalty function to be added to the objective:

$$p(t) = \begin{cases} \alpha t & \mathrm{t} > 0, \\ 0 & \mathrm{t} \leq 0, \end{cases} \tag{8}$$

where α is a large constant so that the penalty of violation increases quickly and large enough to dominate the objective. Statistical results are shown in Table 4, where vio_{ini} is the average value of positive t of initial solutions.

Method	e_{ini}	vio_{ini}	e_{final}	p_{succ}	n_{ite}	n_{func}
LS-Id	5.01%	1.36%	0.30%	87%	14.6	468
LS-Random	4.73%	1.07%	0.30%	88 %	14.3	460
LS-Our	1.82%	0.27%	0.20%	93%	6.7	305

Table 4: Statistical results of case with stress constraints.

As could be seen, the procedure could return a starter with both smaller objective and less violation of the constraint. And the quality of final solution is higher with a reduction in total number of function evaluations.

4.3. 72-Bar Truss Allocation Problem

Finally, we applied the procedure on a large scale problem which contains totally 72 bars in the structure as shown in Figure 3. All the bars are divided into 4 groups with 8,16,16,32 bars respectively. Still, the properties of bars deviate from design with maximum error of 5%. We apply two load cases where the mechanical forces are the same as presented in [7] and the uniform thermal load are identical as former examples. Our goal is to minimize

the displacement of node 20 under loads. We randomly generate 10 instances for each load case. The statistical results of cases without and with stress constraints are presented in Table 5 and 6 respectively.

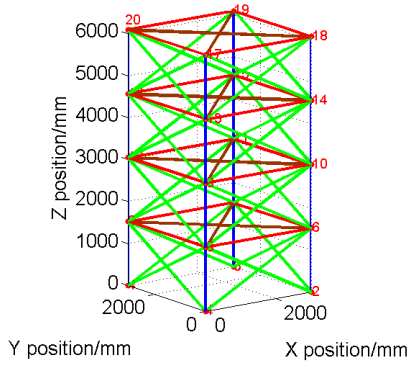


Figure 3: *72-bar truss structure.*

Method	e_{ini}	e_{final}	p_{succ}	n_{ite}	n_{func}
LS-Id	11.8%	0.03%	100%	129	98405
LS-Random	11.6%	0.03%	100%	122	93571
LS-Our	0.16%	0.01%	100%	29.1	23795

Table 5: Statistical results of case without constraints.

Method	e_{ini}	vio_{ini}	e_{final}	p_{succ}	n_{ite}	n_{func}
LS-Id	11.8%	2.62%	0.17%	100%	136	103676
LS-Random	11.5%	2.62%	0.19%	96.5%	132	100546
LS-Our	4.78%	0.36%	0.24%	85%	70.4	55387

Table 6: Statistical results of case with stress constraints.

The total number of possible combinations is $8! \times 16! \times 16! \times 32! \approx 4.6 \times 10^{66}$. We have no way to find the global optimum in this case. So for each instance, we take the best solution obtained by all the three methods as the reference solution and the relative error are calculated with respect to it.

For this large scale problem, comparing to the total number of combinations, the number of function of evaluations we need are much smaller. Although the percentage of successful run is relative low starting from our initial solution, the average final relative error is still of the same level. And the reduction on total number of function evaluations is still significant.

5. CONCLUSION

In this paper, parts allocation problem for structural systems is presented and formulated into a nonlinear assignment problem. Procedure for constructing an initial solution for solving this kind of problem is established.

The procedure has been tested on a 10-bar truss, a 25-bar truss and a large-scale 72-bar truss allocation problem. The performance for problems with stress constraints is also studied. All the results show that our procedure could construct a high quality initial solution for parts allocation problems. A standard 2-exchange local-search algorithm starting from this initial point is able to solve most of our test examples with fewer total number of function evaluations compared with starting from the identity permutation or randomly generated initial solutions.

6. ACKNOWLEDGEMENTS

The authors gratefully acknowledge DAAD (German Academic Exchange Service) for awarding the first author DAAD Scholarship to carry out study at Institute of Lightweight Structures, TU München, Germany.

7. REFERENCES

[1] R. Burkard, M. Dell'Amico, and S. Martello, *Assignment Problems*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2009, ch. Linear sum assignment problem, pp. 73–144.

[2] S. Sahni and T. Gonzalez, "P-complete approximation problems," *Journal of the Association of Computing Machinery*, vol. 23, no. 3, pp. 555–565, July 1976.

[3] A. M. Frieze, "Complextiy of a 3-dimensional assignment problem," *European Journal of Operation Research*, vol. 13, no. 2, pp. 161–164, June 1983.

[4] P. M. Pardalos and L. S. Pitsoulis, *Nonlinear Assignment Problems: Algorithms and Applications (Combinatorial Optimization)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2000, ch. Heuristics for Nonlinear Assignment Problems, pp. 175–215.

[5] E. Aarts and J. K. Lenstra, *Local Search in Combinatorial Optimization*. Princeton, NJ, USA: Princeton University Press, 2003, pp. 57–214.

[6] J. Munkres, "Algorithms for the Assignment and Transportation Problems," *Journal of the Society for Industrial and Applied Mathematics*, vol. 5, no. 1, pp. 32–38, March 1957.

[7] H. Adeli and O. Kamal, "Efficient optimization of space trusses," *Computers and Structures*, vol. 24, no. 3, pp. 501–511, 1986.

Partitioning a service region among several vehicles

John Gunnar Carlsson *

* Industrial and Systems Engineering, University of Minnesota
111 Church St SE, Minneapolis, MN 55455
jgc@isye.umn.edu

ABSTRACT

We consider an uncapacitated stochastic vehicle routing problem in which vehicle depot locations are fixed and client locations in a service region are unknown, but are assumed to be i.i.d. samples from a given probability density function. We present an algorithm for partitioning the service region into sub-regions so as to balance the workloads of all vehicles when the service region is simply connected (has no holes) and point-to-point distances follow some "natural" metric, such as any L_p norm. This algorithm can also be applied to load-balancing of other combinatorial structures, such as minimum spanning trees and minimum matchings.

Keywords: Location, Geometry, Algorithms, Vehicle routing

1. INTRODUCTION

Optimal assignment of a workload between several agents is a common objective that is encountered in resource allocation problems. Frequently, workloads are assigned in such a way as to minimize the total amount of work done by all agents. In other situations, one may want an *equitable* assignment that balances the workload evenly across all agents. Equitable assignment policies are commonly encountered in queueing theory [1, 2, 3], vehicle routing [4, 5, 6], facility location [7, 8, 9, 10], and robotics [11, 12], among others.

Our motivation for this research comes from an industrial affiliate in the form of a stochastic vehicle routing problem. Our objective is to partition a geometric region so as to assign workloads to vehicles in an equitable fashion. Partitioning and routing occupy two different strategic tiers in the optimization hierarchy; partitioning is done at a (high) tactical management level, while routing optimization is operational and made on a day-to-day basis. Hence, a natural strategy, especially in the presence of uncertainty, is to segment the service region into a collection of sub-regions and then to solve each routing sub-problem induced at the sub-regions independently of the others. This approach was used, for example, by [5], who treated the problem as a two-stage optimization problem (partitioning and routing) and implemented a tabu search and multistart heuristic to consider the problem of partitioning a planar graph optimally. This problem is also often considered in the context of facility location [7, 8, 10] and robotics [12].

In this paper, we give an algorithm that takes as input a planar, simply connected (not having holes) region R, together with a probability density $f(\cdot)$ defined on R. Contained in R is a collection of n *depot points* $P = \{p_1, \ldots, p_n\}$, representing the starting locations of a fleet of vehicles. We assume (purely for expositional purposes) that each point p_i corresponds to exactly one vehicle. The vehicles must visit clients whose exact locations are unknown, but are assumed to be i.i.d. samples from the density $f(\cdot)$. Our goal is to partition R into n disjoint sub-regions, with one vehicle assigned to each sub-region, so that the workloads in all sub-regions are asymptotically equal when a large number of samples is drawn.

For each sub-region R_i, we will solve a travelling salesman problem, in which the point set consists of a depot point plus all points in R_i. See figure 1.

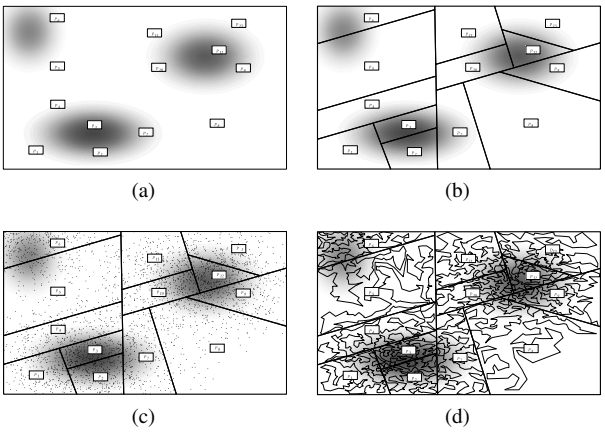


Figure 1: *Inputs and outputs to our problem. We begin with a depot set and a density $f(\cdot)$ defined on a region R (1(a)), which we then partition (1(b)). This partition should be constructed so that, when points are sampled independently from $f(\cdot)$ (1(c)), the TSP tours of all the points in each sub-region are asymptotically equal (1(d)).*

Our problem turns out to be a special case of the *equitable partitioning problem*, in which we are given a pair of densities $\lambda(\cdot)$ and $\mu(\cdot)$ on a region R and we want to partition R into n sub-regions R_i with $\iint_{R_i} \lambda(\cdot)\, dA = \frac{1}{n} \iint_R \lambda(\cdot)\, dA$ and $\iint_{R_i} \mu(\cdot)\, dA = \frac{1}{n} \iint_R \mu(\cdot)\, dA$ for all i. The case where $\lambda(\cdot)$ and $\mu(\cdot)$ are both atomic measures consisting of gn and hn points for some positive integers g and h is a well-studied problem in combinatorial geometry known as a *red-blue* partition [13, 14, 15], and several fast algorithms are already known for this problem. Our problem consists of a "mixed" case where $\lambda(\cdot)$ is an atomic measure consisting of n depot points and $\mu(\cdot)$ represents the TSP workload over a sub-region when points are sampled from $f(\cdot)$.

The outline of this paper is as follows: first, we describe a necessary condition for optimality of a partition of R that follows immediately from well-known results from geometric probability. Next we give an algorithm that finds an optimal partition of R when R is a simply connected polygon. Finally, we present some simulation results that show the solution quality of our algorithm when applied to some simulated problems and a case study.

2. SUMMARY OF KEY FACTS AND FINDINGS FROM RELATED WORK

In this section we summarize the important theoretical results that form the basis of our partitioning algorithm. We consider the travelling salesman problem (TSP) in a planar region R, where the distance between two points is Euclidean, or any other "natural" metric such as the Manhattan or sup norm. The well-known *BHH theorem* [16] says that the length of an optimal TSP tour of a set of points follows a law of large numbers:

Theorem 1. *Suppose that $\{X_i\}$ is a sequence of random points i.i.d. according to a probability density function $f(\cdot)$ defined on a compact planar region R. Then with probability one, the length $\mathrm{TSP}(\{X_1,\ldots,X_k\})$ of the optimal travelling salesman tour traversing points $\{X_1,\ldots,X_k\}$ satisfies*

$$\lim_{k\to\infty}\frac{\mathrm{TSP}(\{X_1,\ldots,X_k\})}{\sqrt{k}}=\beta\iint_R\sqrt{f_c(x)}\,dA \qquad (1)$$

where β is a constant and $f_c(\cdot)$ represents the absolutely continuous part of $f(\cdot)$.

It is additionally known that $0.6250\le\beta\le 0.9204$ [17]. This result was subsequently improved in [18], which showed that a similar law of large numbers holds for any *subadditive Euclidean functional*, such as a minimum-weight matching, minimum spanning tree, Steiner tree, or Delaunay triangulation, with different constants β. Applying a standard coupling argument to (1) gives the following result:

Theorem 2. *Let R be a compact planar region and let $f(\cdot)$ be an absolutely continuous probability density defined on R. Let $\{X_i\}$ be a collection of i.i.d samples drawn from $f(\cdot)$. Let $\{R_1,\ldots,R_n\}$ be a partition of R. If a partition of R into n disjoint pieces $R_1,\ldots,R_n$ satisfies*

$$\iint_{R_i}\sqrt{f(x)}\,dA=\frac{1}{n}\iint_R\sqrt{f(x)}\,dA \qquad (2)$$

for $i\in\{1,\ldots,n\}$, then asymptotically, the lengths of the TSP tours $\mathrm{TSP}(\{X_1,\ldots,X_k\}\cap R_i)$ will differ by a term of order $o(\sqrt{k})$, where k is the number of points sampled. Hence, the maximum tour length over any sub-region R_i differs from the optimal solution by a term of order $o(\sqrt{k})$.

As a special case, we remark that when $f(\cdot)$ is the uniform distribution on R, if a partition of R into n disjoint pieces $\{R_1,\ldots,R_n\}$ satisfies

$$\mathrm{Area}(R_i)=\mathrm{Area}(R)/n$$

then asymptotically, the lengths of the TSP tours $\mathrm{TSP}(\{X_1,\ldots,X_k\}\cap R_i)$ will differ by a term of order $o(\sqrt{k})$.

3. THE EQUITABLE PARTITIONING PROBLEM ON A SIMPLY CONNECTED SERVICE REGION

3.1. Analysis

The optimality condition defined in theorem 2 is easy to achieve, in the absence of other criteria; for example, a partition might consist exclusively of vertical lines, with each vertical strip cutting off $\iint_{\text{strip}}\sqrt{f(x)}\,dA=\frac{1}{n}\iint_R\sqrt{f(x)}\,dA$. For this reason, we will impose additional constraints on our algorithm that should, in principle, give a better solution. Recall that in our original problem statement, we assumed that our service region R contained a set of *depot points* $P=\{p_1,\ldots,p_n\}$. A natural constraint to impose is that each sub-region R_i should contain the depot point that we have assigned to it.

This still leaves us with considerable freedom; we have not yet imposed any constraints on the shape of the sub-regions. A further property that might be desired is that for any two points $u,v\in R_i$, the shortest path between u and v be contained in R_i. When the input region R is convex, this constraint is equivalent to requiring that each sub-region R_i also be convex. When R is not convex, the property that we desire is called *relative convexity* [13]: each sub-region R_i should be convex "relative" to the input region R, so that the shortest path between $u,v\in R_i$ (which may not be a straight line) must itself be contained in R_i. Our main result in this paper is the following theorem:

Theorem 3. *Given a simply connected region S with m vertices, a probability density $\mu(\cdot)$ defined on S such that $\iint_S\mu(x)\,dA=1$, and a collection of points $P=\{p_1,\ldots,p_n\}\subset S$ where the vertices of S and the points in P are all in general position, there exists a partition of S into n relatively convex sub-regions $S_1,\ldots,S_n$ with disjoint interiors, where each sub-region S_i contains exactly one point from P and satisfies $\int_{S_i}\mu(x)\,dA=1/n$. Furthermore, we can find such a partition in running time $\mathcal{O}(nN\log N)$, where $N=m+n$.*

Using theorem 2, by setting $\mu(\cdot)=\sqrt{f(\cdot)}$, the algorithm described in theorem 3 partitions S into n sub-regions whose TSP tours (for points sampled from the density $f(\cdot)$) are asymptotically equal when a large number of points is sampled. For purposes of brevity we will assume that $\mathrm{Area}(S)=1$ and that $f(\cdot)$ is the uniform distribution, so our goal is to partition S into relatively convex pieces of area $1/n$, each containing a point p_i. The reader is invited to refer to [19] for the complete generalization of our algorithm and a proof of its running time. An example of the input and output of our algorithm is shown in figure 2. We let ∂ denote the *boundary*

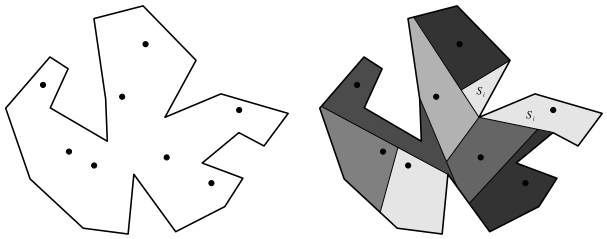


Figure 2: *Inputs S and P (2(a)) and output (2(b)) to our problem, where $\mu(\cdot)$ is the uniform distribution on S. Note that the region marked S_i consists of two polygons joined at a vertex, but still satisfies our relative convexity constraint.*

operator, e.g. ∂S denotes the boundary of S. We let $|\cdot|$ denote the cardinality operator, e.g. $|P|=n$. We begin with some definitions:

Definition 1. *Let S be a compact, simply connected planar region, and let $P=\{p_1,\cdots,p_n\}\subset S$ denote a set of n points, where n is even. A partition $\{S_1,S_2\}$ of S into 2 (relatively) convex subregions is said to be an* equitable (relatively) convex 2-partition *if we have*

$$\frac{\mathrm{Area}(S_1)}{|P\cap S_1|}=\frac{\mathrm{Area}(S_2)}{|P\cap S_2|}.$$

Definition 2. *An S-*geodesic *between two points u and v in a simple polygon S, written $G(u,v|S)$, is the shortest path between u and v contained in S.*

Definition 3. *A sub-region $\tilde{S}$ of a simple polygon S is* relatively convex *to S if, for every pair of points $u,v\in\tilde{S}$, the S-geodesic $G(u,v|S)$ lies in $\tilde{S}$.*

Definition 4. *Given two points u and v on ∂S, the* left shell $\mathscr{L}(u,v|S)$ *consists of all elements of S lying on or to the left of $G(u,v|S)$. If*

u or v does not lie on ∂S, then we define $\mathscr{L}(u,v) = \mathscr{L}\left(u',v'\right)$, where u' and v' are obtained by extending the endpoints of $G(u,v|S)$ via straight lines to ∂S (see figure 3).

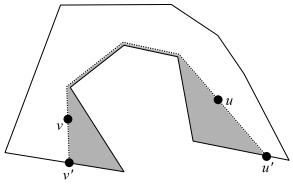


Figure 3: *The geodesic $G(u,v|S)$, its extension points u' and v', and the induced left shell $\mathscr{L}(u,v|S) = \mathscr{L}\left(u',v'|S\right)$.*

Definition 5. *Given a point u on ∂S and a positive integer $\alpha < 1$, define $\text{LShell}_\alpha(u) := v$ to be the unique point on ∂S such that $\text{Area}(\mathscr{L}(u,v|S)) = \alpha$.*

This section consists of a proof of the following theorem:

Theorem 4. *Let x_0 and x_1 be two points on ∂S. If $\text{Area}(\mathscr{L}(x_0,x_1|S)) = \frac{k}{n}$ for some integer $k \le n/2$ and $|\mathscr{L}(x_0,x_1|S) \cap P| > k$, then we can find a relatively convex equitable 2-partition of S and P in running time $\mathscr{O}(N \log N)$, where $N = m+n$.*

Note that theorem 4 is more than sufficient to prove theorem 3 when $n = 2^j$ for some positive integer j and $f(\cdot)$ is the uniform distribution, since we can always meet the necessary conditions of theorem 4 with $k = n/2$ (by dividing S in half with any geodesic, and counting the number of points on either side), and then apply theorem 4 recursively to both sub-regions. This can also be used more generally for other n, although we have omitted the discussion here for brevity (see [19] for the complete result). The remainder of this section consists of a sketch of a proof of this theorem.

As in the theorem, let x_0 and x_1 be two points on ∂S such that $\text{Area}(\mathscr{L}(x_0,x_1|S)) = \frac{k}{n}$ for some integer $k \le n/2$ and $|\mathscr{L}(x_0,x_1|S) \cap P| > k$. Construct another point x_2 on ∂S so that $\text{Area}(\mathscr{L}(x_2,x_0|S)) = \frac{k}{n}$. Then either $|\mathscr{L}(x_2,x_0|S) \cap P| < k$ or $|\mathscr{L}(x_2,x_0|S) \cap P| > k$ (if we have equality then we are finished), and in either case we can derive an equitable 2-partition:

Case 1

Suppose that $|\mathscr{L}(x_2,x_0|S) \cap P| > k$. Then $|\mathscr{L}(x_0,x_2|S) \cap P| < n-k$ and $\text{Area}(\mathscr{L}(x_0,x_2|S)) = \frac{n-k}{n}$. Hence, $\mathscr{L}(x_0,x_1|S)$ contains too many points (relative to its area) and $\mathscr{L}(x_0,x_2|S)$ contains too few points. Consider a family of left shells $\mathscr{L}(x_0,x|S)$, where

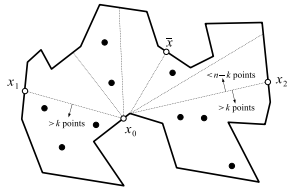


Figure 4: *A family of left shells cutting off area $\frac{k}{n}, \frac{k+1}{n}, \ldots, \frac{n-k}{n}$, with $k = 2$ and $n = 9$.*

x traverses ∂S clockwise from x_1 to x_2; see figure 4. The function $\phi(x) := \text{Area}(\mathscr{L}(x_0,x|S)) - \frac{k}{n}|\mathscr{L}(x_0,x|S) \cap P|$ is piecewise continuous, increasing on each of its components, and decreasing at each discontinuity. Since $\phi(x_1) < 0$ and $\phi(x_2) > 0$, the intermediate value theorem guarantees the existence of a point $\bar{x}$ where $\phi(\bar{x}) = 0$ and our equitable 2-partition is obtained. We can find this by performing a binary search for $i \in \{k,\ldots,n-k\}$, where for each i we compute the point $\text{LShell}_{i/n}(x_0)$ and the number of points contained therein. The preceding argument guarantees that we must find an equitable 2-partition somewhere in this procedure.

Case 2

Suppose that $|\mathscr{L}(x_2,x_0|S) \cap P| < k$. Then, as $|\mathscr{L}(x_0,x_1|S \cap P)| > k$, we have a left shell containing too many points (relative to its area) and another left shell containing too few points. Hence, there

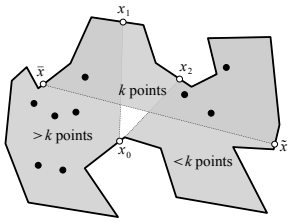


Figure 5: *An equitable geodesic shell exists between $\bar{x}$ and $\tilde{x}$ with $k = 4$ and $n = 9$.*

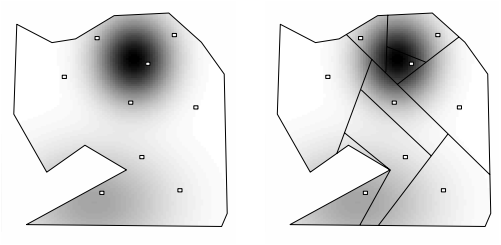
must exist some pair of points $\bar{x},\tilde{x}$ in ∂S such that $\bar{x} \in \partial\mathscr{L}(x_0,x_2|S)$ and $\tilde{x} \in \partial\mathscr{L}(x_1,x_0|S)$ (see figure 5), where $\text{Area}(\mathscr{L}(\bar{x},\tilde{x}|S)) = \frac{k}{n}$ and $|\mathscr{L}(\bar{x},\tilde{x}|S) \cap P| = k$. This is because the function $\text{LShell}_{k/n}(x)$ is continuous in x (for $x \in \partial S$), and the assumption that our points lie in general position ensures that as x traverses ∂S from x_0 to x_2, the elements of P will enter and exit $\mathscr{L}\left(x,\text{LShell}_{k/n}(x)\right)$ one by one.

4. COMPUTATIONAL RESULTS

Theorem 2, our criterion for optimal partitioning, is an asymptotic result. We are guaranteed that vehicle workloads will differ by terms of order $o\left(\sqrt{k}\right)$, but we have not yet established that workloads are in fact balanced when this algorithm is employed (e.g., that the convergence in k may be slow in practice). In this section we give some examples that suggest that vehicle workloads will in fact be balanced in a practical setting when point-to-point distances are Euclidean. We also present the results of a case study in which we apply our partitioning algorithm as a pre-processing stage in a non-Euclidean vehicle routing problem using data supplied from an industrial affiliate. In this problem, we are given the map of a road network of a city, and we must use our fleet of vehicles to traverse every road. This is a multi-vehicle variant of the *Chinese Postman Problem* (CPP), a well-studied routing optimization problem first described in [20].

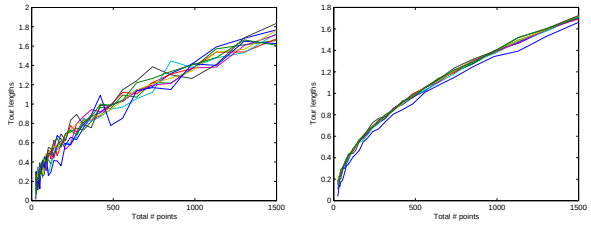
4.1. Simulation results

We first present the results of a simulation in which we construct a synthetic data set with $n = 9$ depots where $f(\cdot)$ is a mixture of three Gaussian distributions, truncated to lie within a simple polygon $S \subset [0,1]^2$. One of the polygons that forms the input to our simulation is shown in figure 6. For each polygon, we generate 20 scenarios, with each scenario consisting of 30 samples of k points in S, for k between 50 and 1500 (and hence we performed a total of 600 simulations per polygon). TSP tours were computed using the Lin-Kernighan heuristic from Concorde [21]. Tour lengths for a particular scenario, and the average vehicle tour lengths over all scenarios, are shown in figure 7. As the plots show, the vehicle workloads are well balanced by partitioning; these suggest that the



(a) (b)

Figure 6: *The input and output to our simulation.*



(a) (b)

Figure 7: *Tour lengths of the* 9 *vehicles in a particular random scenario, and average tour lengths over* 20 *scenarios (7(b)).*

$o(\sqrt{k})$ term of theorem 2 may be negligible, although the variability between vehicle tours for small k is still high. This is not surprising since our partition is "asymptotically optimal" and makes no guarantees for the tour lengths when the number of points is small. A second observation is that our algorithm performs well when many scenarios are averaged, as suggested in figure 7(b). For a related application, figure 8 shows the result of this algorithm applied to a map of Hennepin County, Minnesota, where $\mu(\cdot)$ is the population density and P represents the 29 largest post offices. Rather than producing equal TSP tour lengths, this partitions so that each mail carrier services the same number of houses each day.

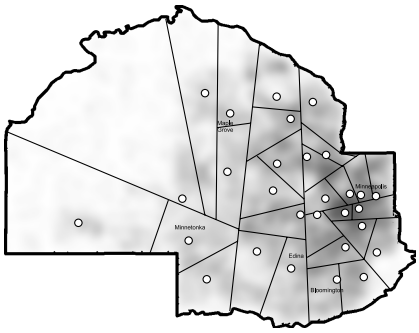


Figure 8: *An equitable partition of Hennepin County, Minnesota. All sub-regions have the same total population and each sub-region contains one post office.*

4.2. Case study

As a final example, we show in figure 9 a partition of the road network of a city that was provided by an industrial affiliate. The objective in this problem is to traverse every street segment in the city with a fleet of vehicles originating at various depots. Although heuristics for these kinds of problems are already known [22], they do not take advantage of the fact that our road map is a planar graph, and consequently vehicle tours may not be geographically separate. In a practical setting it is desirable to separate one vehicle's route from another in an obvious geographic way so as to localize drivers to specific areas of the city.

In our partition, each sub-region contains a depot and all sub-regions contain (approximately) the same total amount of roads.

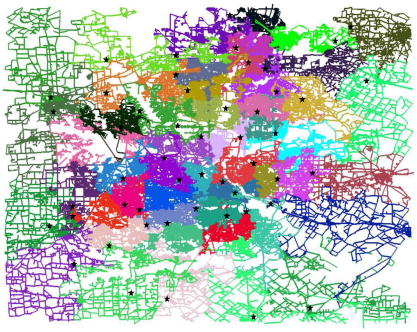


Figure 9: *An equitable partition of a road network that is relatively convex with respect to the metric induced by the road network. All sub-regions have the same total road mass and each sub-region contains a depot.*

Each sub-region is "relatively convex" to the metric induced by the road network (i.e. for any two points $u, v \in R_i$, the shortest path from u to v lies in R_i). The lengths of the total amount of roads in each sub-region differ by a factor of at most 1.11.

5. REFERENCES

[1] Y. Azar, "On-line load balancing," in *Online Algorithms*, ser. Lecture Notes in Computer Science, vol. 1442. Springer Berlin / Heidelberg, 1998, pp. 178–195.

[2] Y. He and Z. Tan, "Ordinal on-line scheduling for maximizing the minimum machine completion time," *Journal of Combinatorial Optimization*, vol. 6, no. 2, pp. 199–206, June 2002.

[3] H. Kellerer, V. Kotov, M. G. Speranza, and Z. Tuza, "Semi on-line algorithms for the partition problem," *Operations Research Letters*, vol. 21, no. 5, pp. 235 – 242, 1997.

[4] J. G. Carlsson, D. Ge, A. Subramaniam, and Y. Ye, "Solving the min-max multi-depot vehicle routing problem," in *Proceedings of the FIELDS Workshop on Global Optimization*, 2007. [Online]. Available: http://www.stanford.edu/~yyye/MDVRP-JGSWY.pdf

[5] D. Haugland, S. C. Ho, and G. Laporte, "Designing delivery districts for the vehicle routing problem with stochastic demands," *European Journal of Operational Research*, vol. 180, no. 3, pp. 997 – 1010, 2007. [Online]. Available: http://www.sciencedirect.com/science/article/B6VCT-4K9C5B8-5/2/d783603a3a80c1d1e6379a16d47d59ce

[6] M. Pavone, N. Bisnik, E. Frazzoli, and V. Isler, "Decentralized vehicle routing in a stochastic and dynamic environment with customer impatience," in *RoboComm '07: Proceedings of the 1st international conference on Robot communication and coordination*. IEEE Press, 2007, pp. 1–8.

[7] B. Aronov, P. Carmi, and M. Katz, "Minimum-cost load-balancing partitions," *Algorithmica*, vol. 54, no. 3, pp. 318–336, July 2009. [Online]. Available: http://www.springerlink.com/content/v42887v071p41701/

[8] O. Baron, O. Berman, D. Krass, and Q. Wang, "The equitable location problem on the plane," *European Journal of Operational Research*, vol. 183, pp. 578–590, 2007.

[9] O. Berman, Z. Drezner, A. Tamir, and G. O. Wesolowsky, "Optimal location with equitable loads," *Annals of Operations Research*, vol. 167, no. 1, pp. 307–325, March 2009.

[10] Z. Drezner and A. Suzuki, "Covering continuous demand in the plane," *Journal of the Operational Research Society*, vol. 61, no. 5, pp. 878–881, 2010.

[11] M. Jäger and B. Nebel, "Dynamic decentralized area partitioning for cooperating cleaning robots," in *in ICRA 2002*, 2002, pp. 3577–3582.

[12] M. Pavone, A. Arsie, E. Frazzoli, and F. Bullo, "Distributed policies for equitable partitioning: theory and applications," in *Proceedings of the 47th IEEE Conference on Decision and Control*. Piscataway, NJ, USA: IEEE Press, 2008, pp. 4191–4197.

[13] S. Bereg, P. Bose, and D. Kirkpatrick, "Equitable subdivisions within polygonal regions," *Computational Geometry*, vol. 34, no. 1, pp. 20 – 27, 2006, special Issue on the Japan Conference on Discrete and Computational Geometry 2004. [Online]. Available: http://www.sciencedirect.com/science/article/B6TYS-4H877G8-3/2/635100921efef04dc3e364aa283b958b

[14] S. Bespamyatnikh, D. Kirkpatrick, and J. Snoeyink, "Generalizing ham sandwich cuts to equitable subdivisions," *Discrete and Computational Geometry*, vol. 24, pp. 605–622, 2000. [Online]. Available: http://dx.doi.org/10.1007/s004540010065

[15] A. Kaneko and M. Kano, "Discrete geometry on red and blue points in the plane - a survey," in *in Discrete and Computational Geometry, The Goodman-Pollack Festschrift*. Springer, 2003, pp. 551–570.

[16] J. Beardwood, J. Halton, and J. Hammersley, "The shortest path through many points," *Proceedings of the Cambridge Philosophical Society*, vol. 55, pp. 299–327, 1959.

[17] D. L. Applegate, R. E. Bixby, V. Chvatal, and W. J. Cook, *The Traveling Salesman Problem: A Computational Study (Princeton Series in Applied Mathematics)*. Princeton, NJ, USA: Princeton University Press, 2007.

[18] J. M. Steele, "Subadditive euclidean functionals and nonlinear growth in geometric probability," *The Annals of Probability*, vol. 9, no. 3, pp. 365–376, 1981. [Online]. Available: http://www.jstor.org/stable/2243524

[19] J. G. Carlsson, "Equitable partitioning for multi-depot vehicle routing," *INFORMS Journal on Computing*, Under revision, see http://www.tc.umn.edu/~jcarlsso/equitable-partitioning-IJOC-revision.pdf.

[20] M. K. Kwan, "Graphic programming using odd or even points," *Chinese Math.*, vol. 1, pp. 273–277, 1962.

[21] W. Cook, "Concorde TSP Solver," http://www.tsp.gatech.edu/concorde.html, 1997–2005.

[22] G. N. Frederickson, "Approximation algorithms for some postman problems," *J. ACM*, vol. 26, pp. 538–554, July 1979. [Online]. Available: http://doi.acm.org/10.1145/322139.322150

A Bi-Objective Approach for Selection of Sugarcane Varieties in Brazilian Companies

Margarida Vaz Pato [*] [†] Helenice de Oliveira Florentino [‡]

[*] Instituto Superior de Economia e Gestão, Universidade Técnica de Lisboa, Portugal, FCUP
Address: Depto. Matemática, ISEG, Rua do Quelhas, 6, 1200-781, Lisboa, Portugal
mpato@iseg.utl.pt

[†] Centro de Investigação Operacional, Faculdade de Ciências, Universidade de Lisboa, Portugal

[‡] Depto. Bioestatística, Instituto de Biociências, Universidade Estadual Paulista, Botucatu, Brasil
Address: Rubião Júnior, P. O. Box 510, CEP 18618-000, Botucatu, São Paulo, Brazil
helenice@ibb.unesp.br

ABSTRACT

The selection of sugarcane varieties is an important problem faced by sugarcane mill companies confronted by the issue of efficiency and the reduction of damage to the environment. Here the authors present the problem of sugarcane variety selection in the light of technical constraints and the aim to minimize collection and transport costs of the residue from sugarcane harvest and maximize energy obtained from the residue. This problem will be presented and formalized within bi-objective binary linear programming. The study is mainly devoted to the application of a bi-objective genetic algorithm to solve real problems addressed in the São Paulo State of Brazil. Results from the computational experiment undertaken will be reported.

Keywords: Selection of sugarcane varieties, Bi-objective genetic algorithm

1. INTRODUCTION

Brazil is the world's largest sugarcane producer. This crop is mainly used to obtain ethanol, sugar and energy. Currently, the big worry for environmental and governmental organizations arises from the residue generated when harvesting. On one hand, the common practice of burning the straw prior to harvest brings about serious environmental damages and will soon be prohibited. On the other hand, the absence of burnings, leading to the additional straw accumulating on the soil creates favourable conditions for parasites and delays sugarcane shooting, thus compromising the next crop. Therefore, the destiny of this residual material in the field has been the subject of many studies. Of particular interest is the one devoted to the selection of sugarcane varieties designed to cope with environmental and economic requirement issues, in short referred to as SSVP.

A model for the SSVP will be given, followed by a brief presentation of a bi-objective genetic algorithm and, finally, by computational results.

2. MATHEMATICAL MODEL

The SSVP consists of determining which of the n varieties adapted to local soil and climate conditions should be planted in each of the k plots. They should, at the same time offer the lowest possible field-to-mill transfer cost and maximum energy balance for residual biomass from the sugarcane harvest. Moreover, the solution must satisfy sucrose and fibre limits for sugarcane, recommended by the company, use the whole area set aside for sugarcane plantation and respect the specific varieties' area limits.

To construct a bi-objective binary linear programming model for the SSVP we consider the decision variables $x_{ij} = 1$ if sugarcane variety i is planted in plot j, $x_{ij} = 0$, in the opposite case (for all $i = 1,2,\ldots,n$; $j = 1,2,\ldots,k$) and the parameters:

c_{ij}: transfer cost of the residual biomass produced from 1 ha of sugarcane variety j on plot i;

e_{ij}: energy balance of the biomass from 1 ha of variety j on plot i;

s_{ij}: estimated sucrose production from plot j should it be planted with variety i;

Slo: minimum quantity established for the total sugar to be extracted from the planting area;

f_{ij}: estimated fibre content of sugarcane planted in plot j with variety i;

Flo, Fup: lower and upper bounds established for the total quantity of fibre;

L_j: area of plot j;

Lup_i: maximum area for variety i.

The model follows:

$$\text{minimize } f_1(x) = \sum_{i=1}^{n}\sum_{j=1}^{k} c_{ij}x_{ij} \tag{1}$$

$$\text{maximize } f_2(x) = \sum_{i=1}^{n}\sum_{j=1}^{k} e_{ij}x_{ij} \tag{2}$$

subject to

$$\sum_{i=1}^{n}\sum_{j=1}^{k} s_{ij}x_{ij} \geq Slo \tag{3}$$

$$Flo \leq \sum_{i=1}^{n}\sum_{j=1}^{k} f_{ij}x_{ij} \leq Fup \tag{4}$$

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad j = 1,2,\ldots,k \tag{5}$$

$$\sum_{j=1}^{k} L_j x_{ij} \leq Lup_i \qquad i = 1,2,\ldots,n \tag{6}$$

$$x_{ij} = 0 \text{ or } 1 \qquad i = 1,2,\ldots,n; j = 1,2,\ldots,k \tag{7}$$

This multi-objective optimization problem (MOP) is similar to the one presented in [1], however more complete from the practical perspective insofar as it preserves the quality of sugarcane in terms of fiber and optimizes both cost and energy balance.

The SSVP is NP-hard, hence non-exact methods are required to cope with the medium/high dimension instances of the SSVP characterizing the most frequent real cases arising from companies in the Mid South region of Brazil.

3. BI-OBJECTIVE GENETIC ALGORITHM

From among the many types of non-exact multi-objective methods, the genetic or evolutionary heuristics have proved to be successful in obtaining solutions for difficult MOPs. The reason for this is that they deal with a population of solutions with different characteristics as to the optimization goals. [2] covers the actual research and application in the field. Genetic heuristics have been successfully applied for multi-objective problems with knapsack and semi- assignment type constraints, e.g. [3],[4].

Within the new bi-objective genetic algorithm we developed for SSVP each individual of the population is characterized by a single chromosome that represents a solution for the SSVP. The chromosome is encoded through an integer valued vector whose k components provide the sugarcane varieties selected. Hence, in this representation each gene is a variety, the very one proposed for the plot. The solution may or not be feasible and, in this case, both cost and energy are penalized. To evaluate the individual's fitness, the simple rank concept is used, thus giving relevance to the dominance relations, as within NSGA type algorithms [2].

The dimension of the population in every generation is N=100 and the maximum number of generations is $Nmax$=2000. Two different processes are used to generate the individuals of the initial population: one is a constructive algorithm to produce Ng=4 individuals by enforcing the bounding constraints of the SSVP - (3) (4) and (6) and the other algorithm randomly generates the remaining N-Ng individuals.

As to the operators, five basic operators are applied to the current population, to create the population of the next generation: selection, crossover, mutation, repair and elitism. The selection operator is a standard binary tournament to build the *Pool*, giving priority to an individual with a low cost and a high energy balance. The crossover is the one point procedure. When a child is not feasible, it is repaired through the action of the repair operator, the above constructive algorithm. Afterwards, each child replaces any one of the parents in the *Pool*, but only if it is fairly better than that parent is as regards the dominance relation. Then mutation applies with probability p_m=0.05 on each gene of all the chromosomes of the *Pool*. If a gene is going to mutate, the sugarcane variety for the respective plot is randomly chosen by giving equal probability to all the n varieties. Again, if the mutant is not feasible, then the repair operator is applied. Finally, within the elitist operator, all the potentially efficient individuals of the previous generation, here represented by S^*, are included in the *Pool* and the population for the next generation is determined by eliminating the $|S^*|$ less fitted individuals from the *Pool*.

4. COMPUTATIONAL RESULTS

The bi-objective genetic algorithm was tested along with an exact method with an SSVP instance corresponding to a small company of the São Paulo State in Brazil [5], thus producing results that will be given at the talk. This company deals with 10 sugarcane varieties and possesses a total area of 315.81 ha. Other 80 simulated instances, corresponding to fields from 405 to 6075 ha, have also been solved with the above algorithm.

The effect of the genetic evolution on the initial population for all the 81 test instances and the computing times will be shown. The quality of the solutions obtained from the genetic algorithm is accessed through performance measures [6]]. These figures show that, at low computing times, the spread within the non-exact frontier is high and the cardinality of this frontier is also significant.

All the programs were coded in MATLAB [7] and ran on CORE 2 QUAD computers with 2.83 GHz and 2G RAM at the Department of Biostatistics, UNESP, Botucatu, Brazil.

5. FINAL COMMENTS

Results obtained from the computational experiment reveal the favorable behavior of the bi-objective genetic heuristic specially devised for SSVP, both from the mathematical and the practical perspectives.

Hence, this methodology will be appropriate in helping managers of sugarcane mills in the Brazilian Mid South region to plan their production activities.

6. ACKNOWLEDGEMENTS

Thanks are due to FUNDUNESP and FAPESP, Brazil (grants No. 2009/14901-4, No. 2010/07585-6) and to FCT, Portugal (project POCTI/ISFL/152) for the financial support.

7. REFERENCES

[1] H. O. Florentino, E. V. Moreno, and M. M. P. Sartori, "Multiobjective optimization of economic balances of sugarcane harvest biomass," *Scientia Agricola (Brazil)*, vol. 65, pp. 561–564, 2008.

[2] C. A. C. Coello, G. B. Lamont, and D. A. V. Velduizen, *Evolutionary algorithms for solving multi-objective problems*, 2nd ed. New York: Springer, 2007.

[3] K. Florios, G. Mavrotas, and D. Diakoulaki, "Solving multiobjective, multiconstraint knapsack problems using mathematical programming and evolutionary algorithms," *European Journal of Operational Research*, vol. 203, pp. 14–21, 2010.

[4] P. R. Harper, V. de Senna, I. T. Vieira, and A. K. Shahani, "A genetic algorithm for the project assignment problem," *Computers & Operations Research*, vol. 32, pp. 1255–1265, 2005.

[5] A. D. Lima, "Otimização do aproveitamento do palhiço da biomassa residual da colheita de cana-de-açúcar," Ph.D. dissertation, Faculdade de Ciências Agronômicas, UNESP, Botucatu, Brazil, 2009.

[6] Y. Collette and P. Siarry, *Multiobjective optimization: Principles and case studies*. Berlin: Springer, 2003.

[7] M. version 7.6.0.324 (R2008a), *High performance numeric computation and visualization software: Reference Guide*. Natick, USA: The MathWorks Inc., 2008.

An Imputation Algorithm Applied the Nonresponse Problem

Jose Brito * Nelson Maculan † Luiz Ochi ‡ Flavio Montenegro § Luciana Brito °

* ENCE, Escola Nacional de Ciências Estatísticas
Rua André Cavalcanti,106, sl 403, CEP:20231-050, Rio de Janeiro, Brazil
jose.m.brito@ibge.gov.br

† COPPE, Universidade Federal do Rio de Janeiro
P.O. Box 68511, 21941-972 Rio de Janeiro, Brazil
maculan@cos.ufrj.br

‡ UFF, Universidade Federal Fluminense, Instituto de Computação
Rua Passo da Pátria 156, Bloco E, 3º andar, São Domingos, Niterói, RJ, Brazil
satoru@dcc.ic.uff.br

§ IBGE, Instituto Brasileiro de Geografia e Estatística, DPE/COMEQ
Av.Chile, 500, 10º Andar, Centro, Rio de Janeiro, RJ, Brazil
flavio.montenegro@ibge.gov.br

° UNIPLI, Centro Universitário Plínio Leite
Av. Visconde do Rio Branco, 123, Centro, Niterói, RJ, Brazil
luroquebrito@hotmail.com

ABSTRACT

This work describes an imputation algorithm to solve the nonresponse problem in surveys. The nonresponse is associated the occurrence of missing values in at least one variable of at least registry or unit of the survey. In order to prevent the negative effects of nonresponse, an intense research has been produced in this area and many procedures have been implemented. Among these, we detach the imputation methods, that consist basically of substituting a missing value by some suitable one, according some criterion or rule. In this work we propose a new imputation algorithm that combines the clustering method and GRASP metaheuristic.To evaluete its performance we present a set of computational results considering data from Brazilian Demographic Census 2000.

Keywords: Nonresponse, Imputation, GRASP, Cluster Analysis, Survey

1. INTRODUCTION

Nonresponse is a normal but undesirable feature of a survey [1]. It is characterized by incomplete records of a survey database, which may occur in the phase of data collection or data estimation. Nonresponse occurs when, at least for one sampling unit (household, person, etc) of the population or sample [2] of the survey, there is nonresponse to one question of a questionnaire (record) or the information given is not usable. Or else, when at least one item of a questionnaire was not completed (survey variable). Incomplete questionnaires due to nonresponse are common in surveys, but deserve attention. Therefore, a considerable amount of money has been spent in the development and improvement of procedures associated to data assessment, in order to prevent the occurrence of nonresponse or to minimize its negative effects. There has been extensive research in this field, which is reported in many studies, such as [1, 3, 4, 5]. Among the procedures being developed are those classified as imputation methods, which basically consist in replacing a missing data with an estimated value, according to a criterion or rule [1]. With the purpose of treating the nonresponse issue, the present study introduces a method that combines an imputation rule, a technique of cluster analysis [6, 7] and GRASP metaheuristics [8, 9] (Greedy Randomized Adaptive Search).

2. NONRESPONSE AND IMPUTATION

There are two types of nonresponse: (1) total nonresponse, which corresponds to the units from which no usable information was collected, and partial nonresponse, corresponding to the units from which there is at least one variable with a missing value and which are not part of the total nonresponse set. The present study has focused on the treatment of partial nonresponse. Then, the concept of nonresponse is described in greater detail, with emphasis on some procedures for the treatment of nonresponse through imputation methods. At first we may consider a set of p variables associated e.g. to the sociodemographic characteristics of a survey defined by $X_1, X_2, ..., X_p$. Such characteristics are obtained for n persons (records), which determines a matrix X_{np} that has for each input X_{ij} the value of the jth variable (characteristic) observed in the ith $i = 1, ..., n$ record. If a M_{ij} indicating variable of the observation of the corresponding data is associated to each X_{ij}, we'll have $M_{ij} = 1$, If there is a value for X_{ij} and $M_{ij} = 0$, If it is otherwise. And based on this, a M matrix that defines the pattern of the missing data is defined. In the present article, we shall treat the missing data associated to one single variable X_j (Univariate Missing Data), known as the study variable. That is, the matrix M shall have zero elements in only one of its columns. The remaining variables $(p-1)$ shall be treated as explicative variables, that is, variables correlated with the variable of interest and that can be used to predict the values of this variable.

When incomplete records are found in a given database, that is, when there is missing information on one of the variables of the database, data can be imputed. Imputation is a procedure through which the missing values for one or more study variables "are

filled" with estimated values [1]. These "replacements" must be performed according to a rule. The imputed values can be classified into three main categories: (i) values constructed using a device for automatic imputation of missing values, considering an imputation statistical rule (ii) values observed for elements with similar response; (iii) values constructed by expert opinion or "by the best possible estimate" [1]. The categories (i) and (ii) can be called statistical rules because they use a statistical method aimed to produce a replacement value reasonably close to the original value. The (i) category is frequently based on regression prediction [1]. Imputation is especially used in the treatment of partial nonresponse, which concerns the simulations presented in this article, although it can also be used in the treatment of total nonresponse.

There are several methods of imputation [1, 5], such as: (1) Nearest Neighbor Imputation: a function of the distance between the complete and incomplete records is calculated considering the explicative variables $(p-1)$. The value of the observed unit with the smallest distance to the non-respondent unit will be substituted for the missing item. (2) Hot Deck Imputation: the variable X_j associated to an incomplete record is substituted for a value obtained from a distribution estimated from the available data (complete records). A complete record (donor) is selected in order to provide values for the missing information in the incomplete record (recipient). This method is typically implemented in two stages: in the first stage, a set of data is distributed into k groups (imputation classes) considering the explicative variables $(p-1)$ associated to the study variable. Once the k groups are defined, in the second stage, the group of each incomplete record is identified. The complete records of a group are used to estimate the unknown values in the incomplete records. (3) Mean imputation: it is a simple method applicable to continuous variables. It substitutes the missing values with the general mean for the variable.

3. METHODOLOGY

The present study shall treat the problem of nonresponse with the type of imputation classes used in the Hot Deck method, expanding the use of these classes to the cases of mean imputation (which is then based on records associated to each one of these classes). Since the definition of the imputation classes has direct impact on the incomplete records, a new methodology for the definition of the classes shall be proposed in this study, with the application of the cluster analysis, a technique widely used to solve the problem of obtaining homogeneous groups (clusters) from a database with special characteristics or attributes [7]. The clusters formed are characterized as follows: the objects of one cluster are very similar and the objects or different clusters are very dissimilar, considering the objective function (that aggregates the distances) shown in the equation below.

$$f = \sum_{l=1}^{k} \sum_{\forall o_s, o_r \in C_l} d_{sr} \qquad (1)$$

The function presented in the equation 1 considers for each cluster $C_l, l = 1, ..., k$ the sum of all the objects that are part of the group. Therefore, minimizing f consists in allocating all the objects to the clusters in such a way that the total sum of the distances (dissimilarities) between two objects from each one of the clusters is minimum.

Regardless the objective function considered or other distance functions, this is not a simple task because of the combinatorial nature of this type of problem (see also [10, 11]). If a process of exhaustive search is used to obtain an optimal solution, all solutions shall be enumerated, that is, all the possibilities of combination of the objects n in groups k. In general, the m number of possibilities grows exponentially as a function of n [6]. Such characteristic makes it impracticable to obtain the exact resolution of average and large instances of these problems. A previous study on metaheuristics applied to cluster problems [12, 13, 14, 15] suggests that it is a good alternative for the resolution of several clustering problems. In general, with the application of metaheuristics, feasible solutions of higher quality than those from heuristics (local minimums) are obtained.

Considering the last observation, and with the purpose of constructing the classes used in the imputation of data, a cluster algorithm that uses GRASP meta-heuristics was developed [9] and whose objective function is the equation (1). The GRASP is an iterative greedy heuristic to solve combinatorial optimization problems. Each iteration of the GRASP algorithm contains two steps: construction and local search. In the construction, a feasible solution is built using a randomized greedy algorithm, while in the next step a local search heuristic is applied based on the constructed solution.

3.1. Grasp Algorihtm

Construction Procedure: Considering a D set formed by objects n (records of a database) and a fixed number of clusters k, k objects of D are selected, with each object allocated to a cluster $C_l, l = 1, .., k$. Then, in each construction iteration, each one of the $(n-k)$ objects is allocated considering their proximity to the objects o_j that are already part of each group C_l. That is, in each iteration, there is a list of candidates LC composed of objects o_i not yet allocated to a cluster and two vectors q and g. Each position q contains the number of the cluster where the closest object o_j is located (using the 1 equation of each object o_i). The vector g corresponds to the distance of the object o_j in the database located at the shortest distance from each object o_i. Based on the referred information, a LCR restricted candidate list is constructed, which is formed by the o_i objects, so that $g_i \leq g_{min} + \alpha(g_{max} - g_{min})$. Being g_{max} and g_{min}, respectively the maximum and minimum distances found in g. Then, an object LCR (element) is randomly selected and allocated to one of the clusters considering the information stored in q. Every time a new object is inserted in one of the clusters, the candidate list is updated. And when $LC = \emptyset$ all the objects shall be allocated to one of the clusters k.

Local Search Procedure: At this step, the reallocation of objects between the clusters k is sought, in order to reduce the value of the equation (1), and consequently, produce more homogeneous clusters (classes) for performing the imputation. Considering the solution obtained in the construction step, in each iteration of this procedure, two clusters C_r and C_l are selected from the clusters k defined in the construction step. Afterwards, various (random) selections of an object $o_i \in C_r$ and an object $o_j \in C_l$ are performed, and in each selection the distances d_i, d_{il}, d_j, d_{jr} are calculated. The values for d_i and d_j correspond respectively to the sum of the distances from object o_i to the other objects C_r and the sum of the distances from object o_j to the other objects C_l. And d_{il} represents the sum of the distances from object o_i to the other objects C_l. An equal definition is applied to d_{jr}, though considering the sum of the distances between the object o_j and the objects C_r. After the calculation of the distances d_i, d_{il}, d_j, d_{jr}, three types of reallocations are assessed:

(1) The object o_i is allocated to cluster C_l and the object o_j is allocated to cluster C_r and $d = -d_i + d_{il} - d_j + d_{jr}$ is calculated.

(2) The object o_i is allocated to cluster C_l and $d = -d_i + d_{il}$ is calculated

(3) The object o_j is allocated to cluster C_r and $d = -d_j + d_{jr}$ is calculated.

The reallocation that produces the greatest reduction (lowest value

of d) in the objective function given by (1) shall be applied in the current solution. Such reallocations are performed until the improvements w (reductions) in the value of the objective function are obtained, or until the number of replacement attempts is equal to a value of $n_{C_r} * n_{C_l}$. Being n_{C_r} and n_{C_l}, respectively the number of objects in clusters C_r and C_l. When at least one of the conditions is satisfied, we get back to the main loop and select two new clusters. At the end of the local search, the new candidate solution generated is checked and compared to the best results obtained so far, considering previous GRASP iterations.

3.2. Imputation Algorithm

The imputation algorithm considers, as input, a database with n records, with complete information for the $(p-1)$ explicative variables, $X_1, X_2, ...X_{p-1}$. Besides, the missing information for the study variable X_p in a given number $n* < n$ of records, or else, a percentage of nonresponse. Then, the two basic steps of the algorithm are described:

• The algorithm GRASP is applied in the determination of the imputation classes considering the number of clusters equal to k. The objective function presented in the equation 1 and used in the GRASP considers, for cluster purposes, the distances between the explicative variables $(p-1)$.

• Once the classes are constructed, the procedure of mean imputation is applied in each one of the incomplete records $n*$ in relation to X_p. This implies determining to each class C_l $(l = 1, ..., k)$ each incomplete record i is allocated and assign a value $\bar{X}_l$ that corresponds to the mean (in class l) complete records in relation to variable X_p.

• Thus, $\bar{X}_l = \sum_{i \in C_l} \frac{x_{ip}}{n_{l*}}$, being n_l* the number of complete records in cluster C_l and x_{ip} the value of the variable X_p in the nth complete record that is part of the cluster C_l.

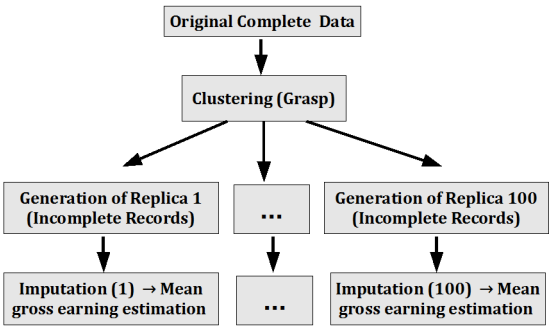


Figure 1: *Phases of the Imputation Algorithm*

4. RESULTS

The present section contains a few computational results obtained with the application of the imputation algorithm, implemented in Delphi language (version 6.0) and run on Windows 7. All the computational experiments are performed in a 16 GB RAM I7 PC with a 2.93 GHz I7 processor. Prior to the presentation of the results, a small description of the data used in the study is made, as well as of the nonresponse mechanism [1, 5, 16] considered for the database used in the experiments.

4.1. Data

In order to perform the experiments, a real database, more specifically, a file of the Sample of the 2000 Brazilian Demographic Census (state of Rio Grande do Sul) was used. Based on this file, nine weighted areas (WAs) were drawn for the simulations with the imputation algorithm. A weighted area is a small geographical area formed by a mutually exclusive enumeration areas (cluster of census segments), which comprise, each one of them, a set of records of households and people [17]. We decided to work with the file of people, where each record is related to the individual characteristics of each inhabitant. And of the variables available in these records, six variables $X_1, ..., X_6$ were selected to be considered in the imputation, as follows: sex, relationship with the responsible person, age in years, highest completed level of education, schooling years and the gross earnings from the main occupation. The five first variables (all categorical) are explicative and correlated to the earnings in reais (quantitative), which was the study variable considered.

4.2. Mechanisms that Lead to Missing Data and the Generation of Incomplete Records

As in any other study aimed to assess whether the method of imputation produces good estimates for the imputed variable [2], the nonresponse mechanism must be considered. That is, since information on a given study variable is missing, these values shall be imputed on a subset of records. In particular, concerning the earnings, it is known that the loss of information is greater for classes with higher income, which characterizes a mechanism of nonresponse called Not Missing at Random (NMAR). This means that the probability of non-information of each input in the nth column of X shall depend on the values observed for the variable X_p in matrix X (see section two). Such mechanism was used to perform the simulations considering a database where all the records contain the information for the study variable (original records). With the application of the nonresponse mechanism, subsets of incomplete records in relation to the gross earnings from the set can be generated, and consequently apply imputation to these records. The number of incomplete records generated in the simulation depends on the rate of nonresponse considered.

One possible procedure for the generation of incomplete records consists in assigning a previous value pr $(0 \leq pr \leq 1)$ that corresponds to the probability of nonresponse (missing information) to the study variable in each original record. In the present study, in particular, such probability was obtained considering the variables relationship with the responsible person (11 categories), highest completed level of education, (10 categories) and schooling years (four categories). According to the category informed for each one of these variables, a probability pr of 0.1, 0.2 or 0.3 of the earning value (X_6) not being informed was attributed to each record. The more the category is related to high earnings, the greater the probability is [16]. Once this probability is defined, a value between 0 and 1 is drawn for each record, and this value is compared to the probability of nonresponse (pr) of the record. If the probability of the record is lower than the value drawn, such record shall have the gross earning value informed at the incomplete database, and, otherwise, it shall be considered a missing data on this database. With the use of this procedure, r replicas can be generated from the complete database, which correspond to the database with different incomplete records.

4.3. Computational Experiments

Initially, for the applying and validating of the imputation algorithm to the records associated to the nine files of people (WAs)

(see section 4.1), a rate of nonresponse of 10% was defined and $r = 100$ replicas of the original databases were generated with different subsets of incomplete records for each r replica. Applying mean imputation to the incomplete records, we obtain for each replica the complete records and the imputed records. Considering such information, the values $\bar{X_m}^r$ e $\bar{X_c}^r$ were calculated, which correspond to the means associated to X_p considering: all the records of each replica (complete and imputed) and only the complete records. It is also said that the same classes of imputation (clusters) were used in all the replicas. In this particular experiment, the algorithm GRASP was applied considering the values k equal to 4, 6 and 8. Still concerning the GRASP, the number of iterations was fixed in 50, improvements equal to 20 and the parameter α equal to 0.5.

Table (1) shows the results obtained with the application of the imputation algorithm to the records of the nine instances used in the simulations. The first column contains the number of the instance and column two contains the records of each WA. Column three contains the number of constructed clusters (classes of imputation). Columns four and five contain the value of the objective function (1) and the processing time (seconds) to construct the clusters, generate the 100 replicas and apply the imputation. Columns six, seven and eight contain the values of $\bar{X_p}$, $\bar{X_m}$ e $\bar{X_c}$ that correspond, respectively, to the mean of the incomes of all records (original database) and the mean of the means of $\bar{X_m}^r$ and $\bar{X_c}^r$ considering the 100 replicas, that is: $\bar{X_m} = \frac{\sum_{r=1}^{100} \bar{X_m}^r}{100}$ $\bar{X_c} = \frac{\sum_{r=1}^{100} \bar{X_c}^r}{100}$. Finally, column nine contains the value of ρ that corresponds to the relative mean deviation between $\bar{X_p}$ and $\bar{X_m}^r$: $\rho = \sum_{r=1}^{100} \frac{|\bar{X_p} - \bar{X_m}^r|}{\bar{X_m}^r}$.

WA	n	k	Time	F_{OBJ}	$\bar{X_p}$	$\bar{X_c}$	$\bar{X_m}$	ρ
1	178	4	18	2369.3	561.5	559.1	561.5	3.5
		6	6	1262.9		556.1	561.3	3.0
		8	3	783.5		555.2	559.5	3.6
2	222	4	34	3875.4	513.3	509.3	512.2	1.6
		6	11	2095.9		509.8	513.7	1.6
		8	5	1359.7		508.2	512.5	1.6
3	289	4	77	7260.7	373.6	367.6	372.5	2.7
		6	24	4012.4		366.7	371.9	3.1
		8	11	2695.6		367.0	372.0	2.8
4	334	4	113	9268.9	355.3	349.5	354.1	1.7
		6	36	4932.8		350.2	354.2	1.4
		8	17	3349.6		350.2	354.8	1.3
5	410	4	215	12248.0	1174.6	1162.9	1171.1	1.5
		6	64	6808.8		1161.5	1172.9	1.7
		8	30	4359.1		1165.2	1176.1	1.6
6	476	4	332	17383.3	547.3	544.0	547.9	1.3
		6	105	9326.4		541.3	546.4	1.5
		8	49	6201.4		541.9	546.3	1.4
7	539	4	485	21402.2	440.2	438.3	439.2	1.1
		6	153	11655.5		435.3	438.2	1.4
		8	71	7591.6		437.4	440.5	1.3
8	628	4	764	28575.4	590.9	583.4	588.0	0.9
		6	240	14730.3		584.4	589.4	0.9
		8	113	9858.2		582.8	588.5	0.9
9	710	4	1121	38222.6	446.7	443.4	445.8	0.8
		6	349	20743.3		442.8	445.8	0.9
		8	160	13498.0		442.8	445.8	0.9

Table 1: Results for the Imputation Algorithm

The analysis of the results of columns 6, 7 and 8 of table (1) shows that the application of the imputation algorithm has made it possible to obtain good estimates for the mean, considering the 100 replicas. In particular, the values between 0.8% and 3.6% in column nine indicate that the means in relation to the imputed records were reasonably close to the real mean value $\bar{X_p}$.

Based on the results obtained, and despite the need for a greater number of experiments, the combination of GRASP and cluster analysis with an imputation method can be a good alternative to the treatment of the problem of nonresponse and produce good quality estimates for databases with incomplete records. In order to improve this procedure in the future, we intend to adapt it to the treatment of categorical variables. Also, we intend to use other objective functions for the construction of the clusters, as well as other metaheuristics such as ILS or Genetic Algorithms [9].

5. ACKNOWLEDGEMENTS

The FAPERJ (project APQ1 E-26/111.587/2010) (`http://www.faperj.br`) and CNPQ (project 474051/2010-2) (`http://www.cnpq.br`) for the financial suport.

6. REFERENCES

[1] C. E. Sarndal and S. Lundstrom, *Estimation in Surveys with Nonresponse*. John Wiley and Sons Ltd, 2005.

[2] S. L. Lohr, *Sampling: Design Analysis*. Brooks/Cole, Cengage Learning, 2010.

[3] J. G. Bethlehem and H. M. P. Kersten, "On the treatment of nonresponse in sample surveys," *Journal of Official Statistics*, vol. 1, no. 3, pp. 287–300, september 1985.

[4] J. G. Bethlehem, "Reduction of nonresponse bias through regression estimation," *Journal of Official Statistics*, vol. 4, pp. 251–260, december 1988.

[5] R. J. A. Little and D. B. Rubin, *Statistical Analysis with Missing Data*. John Wiley and Sons Ltd, 2002.

[6] A. R. Johnson and D. W. Wichern, *Applied Multivariate Statistical Analysis*. Prentice Hall. Fifth Edition, 2002.

[7] H. C. Romesburg, *Cluster Analysis for Researchers*. Lulu Press, 2004.

[8] T. A. Feo and M. G. C. Resende, "Greedy randomized adaptive search procedures," *Journal of Global Optimization*, vol. 6, pp. 109–133, 1995.

[9] F. Glover and G. Kochenberger, *Handbook of Metaheuristics*. Kluwer Academic Publishers, 2003, pp. 219–249.

[10] P. Hansen and B. Jaumard, "Cluster analysis and mathematical programming," *Mathematical Programming*, vol. 79, pp. 191–215, 1997.

[11] P. A. L. J. Hubert and J. J. Meulman, *Combinatorial Data Analysis: Optimization by Dynamic Programming*. Philadelphia: Society for Industrial and Applied Mathematics, 2001.

[12] M. C. G. Guojun and W. Jianhong, *Data Clustering: Theory, Algorithms and Applications*. ASA-SIAM Series on Statistics and Applied Probability, 2007.

[13] M. J. Brusco and D. Steinley, "A comparison of heuristics procedures for minimum within-cluster sums of squares partitioning," *Psychometrika*, vol. 72, pp. 583–600, 2007.

[14] W. Sheng and X. Liu, "A genetic k-medoids clustering algorithm," *Journal of Heuristics*, vol. 12, pp. 447–446, 2006.

[15] M. C. V. Nascimento, F. M. B. Toledo, and A. C. P. L. F. Carvalho, "Investigation of a new grasp-based clustering algorithma applied to biological data," *Computers and Operations Research*, vol. 37, pp. 1381–1388, 2010.

[16] S. Albieri, "A ausência de respostas em pesquisas: Uma aplicação de métodos de imputação. dissertação impa," 1989.

[17] http:/www.censo2010.ibge.gov.br/altera_idioma.php?idioma=_EN.

Automatic Generation of Algorithms for the Non Guillotine Cutting Problem

J. Alejandro Zepeda * Víctor Parada * Gustavo Gatica † Mauricio Sepúlveda *

* Informatics Engineering Department, University of Santiago of Chile
Santiago, Chile
{jose.zepeda,victor.parada,mauricio.sepulveda}@usach.cl

† Universidad Andrés Bello
Santiago, Chile
ggatica@unab.cl

ABSTRACT

There exist several optimization problems for which an efficient solution algorithm have not been found, they are used in decision making for a lot of production and service processes. In practice, hard problems must be solved in an operational, tactical and strategically way inside several organizations. Using this assumption, developing algorithms for finding an approximate solution or "a good solution" is encouraging.

The automatic generation of optimization programs is an emerging field of research. The construction of programs is developed through several evolving-nature hyper-heuristics or local search method. We used Genetic Programming to find algorithms rewritten as pseudo-code and analyze them to get new knowledge.

The experiment evolved individuals to solve the Non-Guillotine Cutting Stock Problem, a NP-Hard Problem. We tested the population obtained over a data set of instances from literature, the fittest individual averaged 5.4% of material waste and was the object of our analysis. We found interesting blocks of genetic code that resemble intuitive human solutions, and we believe that crafting the terminal and functional elements to facilitate the comparison may help to find interesting even human-competitive algorithms.

Keywords: Genetic programming, Cutting Stock Problem, Algorithms

1. INTRODUCTION

There exist several optimization problems for which an efficient solution algorithm have not been found [1, 2]. They are used in decision making for a lot of production and service processes. In practice, hard problems must be solved in an operational, tactical and strategically way inside several organizations [3]. Generally the main goal of finding the best solution is sacrificed, as either it is not in the computational scope or the search cost is higher than the benefits. Using this assumption, developing algorithms for finding an approximate solution or ä good solutionïs encouraging. An algorithm to solve an optimization problem needs to maximize or minimize some given objective function, so the whole partial solution set must belong to the feasible solution space.
The automatic development of optimization programs is a field of intense research, having Burke as its mayor exponents [4]. The feasible solution is an individual, in this case a computer program that solves a given problem, and the objective function is an evaluator for some characteristics to be searched, for example efficacy, simplicity, size, etc. The Genetic Programming (GP) [5, 6] can be used as a tool to generate algorithms, if some primitives are designed to be easy to comprehend and close to some programming language to establish some parallelism. GP could evolve those structures and find algorithms, rewritten as pseudocode and analyzed to get new knowledge. Some related works have been published by [7] who solved the coloring graph, by [8] who evolved "greedy programs" to solve the Traveling Sales Problem and by [4] who have generated programs to solve the packing problem [4, 9, 10]. This research presents one algorithm generated through GP to solve a NP-Hard Problem, the Non-Guillotine Cutting Stock Problem (NGCSP) [11].

2. GENERATING ALGORITHMS

The generating process of algorithms through GP is presented in a preliminary sequence of general steps depict by [12]: The first step is a clear definition of the problem domain, but without any statement about how to solve it; NGCSP was modeled as a set of data structures and procedures to simulate the process of non-guillotine cutting, i.e., the sheet, the pieces, the geometric constraints, the dynamic process (to obtain a layout pattern through some degrees of freedom to use the entities and behaviors), and an evaluator to assess the result. In this research, we define a set of terminals and functions which fulfill the Closure and Sufficiency properties, using the entities and their behaviors yet mentioned; Then the objective function quantify the fitness of the individual using the model's evaluator. We selected the execution parameters of GP after being identified through local search for different probabilities of mutation and crossover to find the ones best suited for the evolutionary process. Finally, the evolutionary process is run and eventually the fittest individual would be found. This iterative process may require the redefinition of some step, until achieving the generation of algorithms with the performance needed.

The NGCSP considers a rectangular sheet of area A with (W,L) as dimensions, being W the width and L the length. Let R a set of rectangular pieces of lesser dimensions (w_i,l_i), $i = 1,2,\ldots,n$, and area a_i [13]. A layout is a set of pieces cut from the sheet, minimizing the waste of material and fulfilling some rules of geometric feasibility. The mathematical formulation is:

$$\text{Min } Z(x) = W \cdot L - \sum_i w_i \cdot l_i \cdot x_i \text{ where } x_i \geqslant 0, \forall i \in N \qquad (1)$$

There were defined 20 operations, among terminals and functions, and a fitness function that evaluate the performance of each individual. In this case, the fitness is the used area ratio for a fitness case or problem instance (1), being T_p the total pieces cut from container sheet, see equation (2):

$$f = \frac{\sum_{i=1}^{T_p} a_i}{A} \quad (2)$$

Furthermore, let h be the height of the tree in which it is mapped an algorithm automatically generated using GP. Here h is set to 14 and let δ be the total of nodes of a full (strictly) binary tree. Let N be the total of nodes of each individual generated and Parsimony P be the ratio between N and δ. So as to simplify the analysis, it is defined the Correctness C as the total of semantic errors shown in an individual divided by N. Let raw fitness RF, a fitness measure taken directly from the domain problem, here being understood as a measure of error e equals to the sum of ratios of wasted area considering the N_e fitness cases or examples from the domain problem, as shown in (3).

$$RF = \sum_{j=1}^{N_e} 1 - f \quad (3)$$

The standard fitness is calculated using the additional selective pressures C and P, being P a penalty over P_i, being $i = RF$, P, C all summing 1, then $SF = RF * p_{RF} + P * p_P + C * p_C$. To setup the parameters used, a local search tool, ParamILS [14], resulting in the crossover probability of 90%, a mutation probability Swap and Shrink of 1%, respectively. The kernel used is GPC++ developed by [15], a personal computer with an Intel Core I7-940 2.93Ghz processor and 8 GB RAM.

Evolution provided a population of 1500 individuals trained to solve the problem, evolved over a group of 44 instances [16, 17]. Later the same population was tested over a data set of 8 instances published by Hopper, and selected the individual that depicted the best pattern layouts, the smaller waste of material (see Figure 1). Its bloating zones of useless code were cleaned, and this stripped genetic code was synthesized as pseudo-code, analyzed and described. The convergence of the experiment was similar to that of a Genetic Algorithm [18], being very fast in the first generations. Annex 1 shows the best algorithms, whose average loss rate is 5.4%, also includes control parameters, pseudo-code, associated algorithmic complexity and layouts obtained.

3. CONCLUSIONS

It was common to obtain individuals with high polynomial algorithmic complexity $O(n^4)$, with nested looping code apparently unnecessary or redundant and useless code inflation, resulting in a slower execution. In analyzing the algorithms, there are genetic constructs with intuitive procedures, where a cycle of placement of pieces, it is reviewed if it is possible that minor available piece at the time be used a wasted area as a result of impossibility placing there the current minor piece available. The discovered algorithm has a genetic fragment called "greedy" that have been appeared frequently in the fittest individuals, with some variations in shape but easily recognizable in the structural. Within the conditional loop checking the existence of parts, it is included the placement of the piece achieving best fit. Thus, in each step, a decision is taken to put the item that best fit the current situation and the rest remains to be considered a sub-problem. The algorithm optimizes the problem evolved since for all the test instances used a deterministic procedure to find a solution of a certain quality (greater than 90%). An interesting modification to improve the current results would be to add to the set of primitive selectors some terminals for basic allocation strategies. Moreover, given the frequent presence of similar code fragments, the use of ADF would benefit overall performance [5]. Based on the foregoing, we conclude that PG is capable of evolving two-phase algorithm, a constructive and a Local Search. The evolution found a way to solve the problem, and it is perfectly possible to enhance the results in the way to generate new, better and human-competitive solutions [6, 19].

4. ANNEX A: ALGORITHM SPECIFICATIONS

Number of Generation: 1362	Size of Population: 1500
Pc, Pm, Pu: 0.95, 0.04, 0.0	Random Seed: 12470
Used ADF: No	Aptitude: 1.65411

Table 1: Algorithm specifications

Algorithm 1: ADD PIECE

Require: A piece p.

1: $l = l + p$
2: $lA = lA + p$
3: $lL = lL + p$
4: $lW = lW + p$

Algorithm 2: REMOVE PIECE

Require: A piece p.

1: $l = l - p$
2: $lA = lA - p$
3: $lL = lL - p$
4: $lW = lW - p$

Algorithm 3: PUT PIECE

Require: A piece p A space e.
Ensure: Boolean n.

1: **if** PUT PIECE(p,e) **then**
2: REMOVE PIECE(p)
3: e $<--$ availbleSpaceBottomLeft()
4: **return** True
5: **else**
6: **return** False
7: **end if**

Algorithm 4: PUT PIECE

Require: A piece p.
Ensure: Boolean n.

1: e $<--$ availbleSpaceBottomLeft()
2: **if** PUT PIECE(p,e) **then**
3: REMOVE PIECE(p)
4: e $<--$ availbleSpaceBottomLeft()
5: **return** True
6: **else**
7: **return** False
8: **end if**

5. REFERENCES

[1] D. J. M. Garey, *Computers and intractability. A guide to the theory of NP-completeness*. W.H. Freeman and Company, San Francisco, Calif, 1979.

Algorithm 5: GREEDY

Ensure: Boolean b.

```
1: loop = false
2: ad <-- availableArea()
3: while l.notEmpty() && noChange <
   maxTryoutsWithEnhance do
4:     loop = true
5:     PUT PIECE(piece(bestFit))
6:     if ad == availableArea() then
7:         noChange++
8:         return loop
9:     end if
10: end while
```

Algorithm 6: SUB RUTINE 2

Ensure: Boolean b.

```
1: loop = false
2: ad <-- availableArea()
3: while l.notEmpty() && noChange <
   maxTryoutsWithEnhance do
4:     loop = true
5:     p <-- piece(maxWidth)
6:     PUT PIECE(piece(maxWidth))
7:     if ad == availableArea() then
8:         noChange++
9:         return loop
10:    end if
11: end while
```

Algorithm 7: CICLE

Ensure: Boolean b.

```
1: loop = false
2: ad <-- availableArea()
3: while l.notEmpty() && noChange <
   maxTryoutsWithEnhance do
4:     loop = true
5:     p <-- piece(maxWidth)
6:     PUT PIECE(p)
7:     if ad == availableArea() then
8:         noChange++
9:         return loop
10:    end if
11: end while
```

Algorithm 8: SUB RUTINE 1

Ensure: Boolean b.

```
1: loop1 = false
2: ad1 <-- availableArea()
3: while PUT PIECE(piece(maxWidth)) && l.notEmpty() &&
   noChange1 < maxTryoutsWithEnhance) do
4:     loop1 = true
5:     p <-- piece(maxWidth)
6:     PUT PIECE(p)
7:     if waste() then
8:         loop2 = false
9:         ad1 <-- availableArea()
10:        while GREEDY() && l.notEmpty() &&
           noChange2 < maxTryoutsWithEnhance do
11:            loop2 = true
12:            removeMinAreaPiece()
13:            if ad2 == availableArea() then
14:                noChange2++
15:            end if
16:            if no loop2 then
17:                p1 <-- piece(maxWidth)
18:                p2 <-- piece(minArea)
19:                putBlock(vertical, p1, p2)
20:            end if
21:            if ad1 == availableArea() then
22:                noChange1++
23:            end if
24:            return loop1
25:        end while
26:        return loop
27:    end if
28: end while
```


[2] C. H. Papadimitriou, *Computational complexity*. John Wiley and Sons Ltd, 2003.

[3] R. Reid and N. Sanders, *Operations Management*, 4th ed. Wiley, 2009.

[4] E. Burke, G. Kendall, J. Newall, E. Hart, P. Ross, and S. Schulenburg, "Hyper-heuristics an emerging direction in modern search technology," *International Series in Operations Research & Management Science*, 2003.

[5] J. R. Koza, *Genetic Programming IV: Routine Human-Competitive Machine Intelligence*. Kluwer Academic Publishers, 2003.

[6] ——, "Human-competitive results produced by genetic programming," *Genetic Programming and Evolvable Machines*, pp. 1–34, 2003.

[7] J. Shen, *Solving the graph coloring problem using genetic programming*. Stanford, California, 2003.

[8] B. Swope, *Evolution of a Path Generator for a Round-Trip Symmetric Traveling Salesperson Problem Using Genetic Programming*. Stanford, California, 2003.

[9] E. Burke and M. R. Hyde and G. Kendall, "Evolving bin packing heuristics with genetic programming," *In Parallel Problem Solving from Nature*, vol. PPSN IX, pp. 860–869, 2006.

[10] E. Burke, M. Hyde, G. Kendall, and J. Woodward, "Automatic heuristic generation with genetic programming: evolving a jack-of-all-trades or a master of one," *GECCO'07 – Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pp. 1559–1565, 2007.

[11] G. Wascher, H. Haussner, and H. Schumann, "An improved typology of cutting and packing problems," *European Journal of Operational Research*, vol. 183(3), pp. 1109–1130, 2007.

[12] R. Poli, W. Langdon, and N. McPhee, *A Field Guide to Genetic Programming*. Lulu Enterprises, UK Ltd., 2008.

[13] V. Parada, R. Palma, D. Sales, and A. Gomes, "A comparative numerical analysis for the guillotine two-dimensional cutting problem," *Annals of Operations Research*, vol. 96(1), pp. 245–254, 2002.

[14] F. Hutter, H. Hoos, K. Leyton-Brown, and T. Sttzle, "Paramils: An automatic algorithm configuration framework," *Journal of Artificial Intelligence Research*, vol. 36(1), pp. 267–306, 2009.

[15] A. Fraser. (1993) Genetic programming c++ class library. [Online]. Available: http://www.cs.ucl.ac.uk/staff/W.Langdon/ftp/weinbenner/gp.html

Algorithm 9: MAIN

```
1:  l <−− listofreferencestoavailablepieces.
2:  lA <−− listofreferencestoavailablepiecessortedbyarea.
3:  lL <−− listofreferencestoavailablepiecessortedbylength.
4:  lW <−− listofreferencestoavailablepiecessortedbywidth.
5:  e <−− container
6:  if SUB RUTINE 1() then
7:      loop = false
8:      ad1 <−− availableArea()
9:      while SUB RUTINE 2() && l.notEmpty() &&
        noChange1 < maxTryoutsWithEnhance() do
10:         loop = true
11:         ad2 <−− availableArea()
12:         while PUT PIECE(rotate(piece(maxWidth))) &&
            l.notEmpty() && noChange2 <
            maxTryoutsWithEnhance do
13:             PUT PIECE(piece(bestFit))
14:             if ad2 == availableArea() then
15:                 noChange2++
16:             end if
17:             if ad1 == availableArea() then
18:                 noChange1++
19:             end if
20:         end while
21:     end while
22:     if no loop then
23:         return void
24:     end if
25:     if no CICLE() && waste() then
26:         ad3 <−− availableArea()
27:         while l.notEmpty() && noChange3 <
            maxTryoutsWithEnhance do
28:             ad4 <−− availableArea()
29:             while putBlock(horizontal, piece(minLength),
                piece(maxWidth)) && l.notEmpty() &&
                noChange4 < maxTryoutsWithEnhance do
30:                 PUT PIECE(piece(bestFit)
31:                 if ad4 == availableArea() then
32:                     noChange4++
33:                 end if
34:             end while
35:             if ad3 == availableArea() then
36:                 noChange3++
37:             end if
38:         end while
39:     end if
40: end if
```

Table 2: Performance for the fittest individual.

#	Pieces	Placed	Area	% used area	Instance
0	93	92	40000	94.4	hoppern6a
1	94	88	40000	94.0575	hoppern6b
2	94	89	40000	93.85	hoppern6c
3	96	91	40000	95.795	hoppern6d
4	94	90	40000	94.3175	hoppern6e
5	173	170	40000	96.125	hoppern7a
6	170	170	40000	95.97	hoppern7b
7	161	158	40000	94.505	hoppern7c
8	170	170	40000	91.38	hoppern7d
9	182	181	40000	95.815	hoppern7e

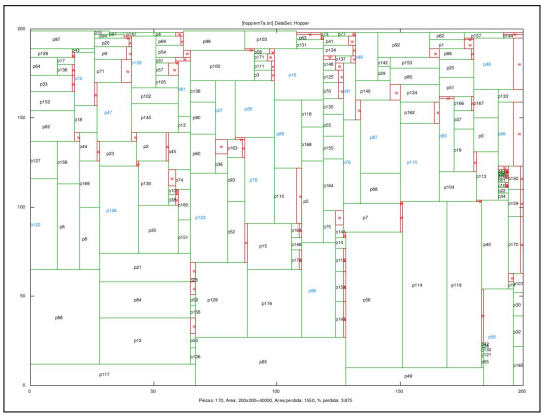


Figure 1: Pattern layout generated by the fittest individual.

[16] J. Beasley, "A population heuristic for constrained two-dimensional non-guillotine cutting," *European Journal of Operational Research*, vol. 156(3), pp. 601–627, 2004.

[17] E. Hopper and B. Turton, "A review of the application of meta-heuristic algorithms to 2d strip packing problems," *Artificial Intelligence Review*, vol. 16(4), 2001.

[18] D. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, 1st ed. Addison-Wesley Professional, 1989.

[19] A. Menon, *Frontiers of Evolutionary Computation*. Springer, 2004.

Enhancements to the best fit heuristic for the orthogonal stock-cutting problem

Jannes Verstichel * † Patrick De Causmaecker † Greet Vanden Berghe *

* CODeS, KAHO Sint Lieven
Gebroeders De Smetstraat 1, 9000 Gent, Belgium
{jannes.verstichel, greet.vandenberghe}@kahosl.be

† CODeS, KU Leuven Campus Kortrijk
Etienne Sabbelaan 53, 8500 Kortrijk, Belgium
patrick.decausmaecker@kuleuven-kortrijk.be

ABSTRACT

We present several enhancements to the best fit heuristic for the orthogonal stock-cutting problem. The solution quality of the heuristic is improved by applying additional placement policies and new orderings of the items. These additions are combined with an optimal time implementation of the heuristic to improve the heuristic's scalability. Experiments on a large test set from the literature show significantly better results in shorter calculation times compared to the original best fit heuristic.

Keywords: Orthogonal stock-cutting, Best fit heuristic

1. INTRODUCTION

Over the years, extensive research has been performed in the domain of cutting and packing problems. The results have been applied in different fields of operations research, for example, the paper and metal industries. Several bibliographic papers exist on typologies for cutting and packing problems [1, 2]. We focus on the two dimensional orthogonal stock cutting problem, which was proven to be NP hard [3]. The goal is to place a number of rectangular items on a rectangular sheet as densely as possible without item overlap, resulting in a minimal height of the sheet needed for placing all the items. A 90 degree rotation of the items is allowed and each stock sheet has a fixed width and infinite length, allowing all items to be placed on a single sheet. Several approaches exist for tackling this problem. A linear and dynamic programming approach is presented in [4], while [5] uses artificial neural networks to solve the problem. One of the best known heuristics for this problem is the bottom left (fill) heuristic and its variants [6, 7, 8]. A best fit heuristic, which outperforms the bottom left based heuristics on all benchmarks with more than 50 items and most smaller instances, is presented by Burke et al. [9]. The scalability of this heuristic has been strongly improved by Imahory and Yagiura [10]. They reduce the time complexity of the best fit heuristic to $O(nlogn)$ and show that the heuristic performs very well for very large data instances. Several metaheuristic approaches to the orthogonal stock cutting problem exist. These are mostly hybridisations that generate different input sequences for existing heuristic approaches in order to improve their results [8, 11, 12]. Other approaches use genetic algorithms [8, 11, 13, 14]. An interesting comparison of different (meta) heuristic approaches and genetic algorithms can be found in [12]. In [15] a metaheuristic combining the best fit heuristic and a simulated annealing bottom left fill hybridisation further improves on the results of [9].

In this abstract, we present several enhancements to the original best fit heuristic. In Section 2, we introduce this adapted best fit heuristic. Next, we improve the time complexity of the heuristic by using the data structures from [10] in Section 3. In Section 4 the results of the heuristic, both with respect to solution quality and computation time, are discussed. Finally, in Section 5 we draw conclusions from our research.

2. THE THREE-WAY BEST FIT HEURISTIC

The original best fit heuristic consists of a preprocessing step, a solution construction and a postprocessing step [9]. In the preprocessing step, all rectangles are rotated in such a way that their width turns out to be their largest dimension. Next, the rectangles are ordered by decreasing width. When this step is finished, the solution construction begins. In this step the lowest gap, i.e. the lowest sequence of x coordinates with an identical height, is located using the sheet skyline. Next the rectangle that fits the width of this gap best, possibly after rotation, is placed in the gap using a predefined placement policy, after which the sheet skyline is updated. If no rectangle can be found to fit the current gap, the skyline at the gap is raised so that it levels with the lowest of the rectangles neighbouring the gap. This process continues until all rectangles are placed on the sheet. After the construction phase, the postprocessing part of the heuristic tries to further improve the solution quality. This is done by checking if the topmost rectangle is placed in portrait, i.e. it has been rotated. If this is the case, the postprocessing step tries to improve the solution by rotating the rectangle by 90 degrees and placing it on the sheet at the lowest possible level. If this leads to an improvement, the process is repeated for the new topmost rectangle. When this procedure does not lead to an improvement, or when the topmost rectangle is already oriented in landscape, the postprocessing step terminates.

The proposed three-way best fit heuristic adds some additional steps to both the preprocessing and the solution construction step. In the preprocessing step, the original best fit heuristic uses a decreasing width ordering of all rectangles. Therefore, the rectangles are always selected for placement in a width decreasing order. We suggest to add two more orderings to the solution process: decreasing height order and decreasing surface order. Applying each one of these orderings ensures a significant disruption of the rectangle sequence compared to the width ordering. The rectangles are always rotated in such a way that their width turns out to be their largest dimension before applying any of the three orderings. The solution construction will be executed for each ordering individually.

With respect to the solution construction step, the original best fit heuristic uses three placement policies: leftmost, tallest and shortest neighbour. Depending on the length of the rectangle that is placed and the length of the gap defining neighbours, a placement policy will decide wether to place the rectangle at the left or the

right side of the gap. We suggest the addition of three more placement policies: rightmost, minimal difference and maximal difference neighbour. These policies will place the new rectangle respectively at the right side of the gap, next to the neighbour with ending height closest to the new rectangle and next to the neighbour with ending height furthest from the new rectangle. An example of the minimal and maximal difference placement policies is shown in Figure 1.

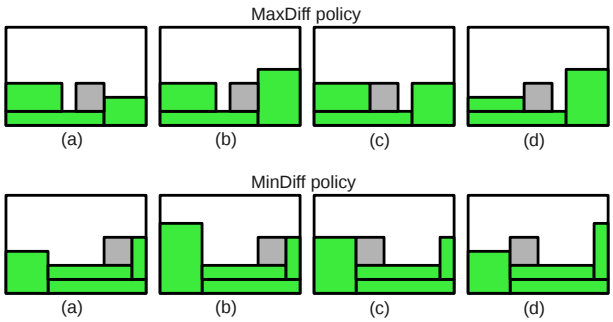


Figure 1: *Example of the maximal difference policy (top) and minimal difference policy (bottom).*

By using both the old and new placement policies and combining them with the decreasing width, height and surface orders, we create a very performant extension to the best fit heuristic. We can call this new heuristic a three-way best fit heuristic as the rectangles are ordered in three different ways during the search for a good solution. In fact, this heuristic solves the problem once for each ordering and placement policy combination. Due to its simple nature and efficient implementation with respect to, for example, overlap checks, the computation times are kept short. An advantage of the heuristic is that orderings and placement strategies can easily be added or removed if wanted. For example, when all shapes under consideration are square, it does not make sense to use more than one of the proposed orders, as they will all result in the same initial sequence.

In some cases, rectangles may have one dimension, we can say the rectangle's width without loss of generality, larger than the sheet width. The best fit heuristic will not prioirtise the placement of these rectangles, as they can only be placed after rotation. The larger the width/length ratio of these rectangles, the higher their probability of being among the last rectangles that are placed. This behaviour strongly decreases the worst case performance of the best fit heuristic. Therefore, we propose the addition of one more rule to the three-way best fit heuristic. It rotates all rectangles with a dimension larger than the sheet width, such that their height is the largest dimension. We apply this rotation after the ordering, such that the rectangle sequence is not changed when compared to the heuristic without this rotation.

3. AN OPTIMAL TIME THREE-WAY HEURISTIC

Imahori and Yagiura [10] analyse the time and space complexity of the original best fit heuristic. They propose alternative data structures to reduce the time and space complexity, and prove that their implementation is optimal. By reducing the time complexity from $O(n^2+W)$ to $O(nlogn)$, they manage to solve instances with 2^{20} rectangles in under 10 seconds. In this section, we discuss the applicability of Imahori and Yagiura's data structures to the new three-way best fit heuristic.

In the original best fit heuristic, the sheet skyline is stored in a integer array, where each element i represents the height of the skyline at width i. The optimal time best fit heuristic stores the sheet skyline using both a heap and a doubly linked list. This allows for a significant improvement with respect to time complexity when compared to using the original data structures [10]. We can now determine the location and size of the lowest available gap in constant time, while updating the skyline requires only $O(logn)$ time, which is a great improvement compared to the original approach [9].

The original best fit heuristic stores the rectangles in an ordered list, iterating the list for each placement until the best fitting rectangle is found. In the optimal time best fit heuristic, the items are stored in a balanced binary tree based on their width. Both the original item and its rotated copy are placed in this tree, in order to allow a $O(logn)$ complexity for finding the best fitting rectangle for the current gap. This balanced tree is however not directly compatible with the previously introduced three-way best fit heuristic. This is due to the mismatch between the alternative orderings of the items, based on the height or the size of the rectangles, and the rectangle selection procedure which is based on the width of the gap. When using this data structure combined with a decreasing height ordering, the items will be placed with their height as the largest dimension. As this portrait placement is not desirable with respect to solution quality, a more advanced decreasing height ordering must be implemented. This ordering will sort the items based on their height, while making a distinction between normal items, oriented in landscape, and rotated items that are oriented in portrait. When ordering all the items and their rotated copy using this advanced height ordering, the same priority list is created as when 'expanding' the original height ordered list (i.e. adding the rotated copies at the correct place in the list). A disadvantage of this ordering is its inconsistency with respect to the width of the items. Therefore it is not possible to use this advanced ordering to obtain the best fitting rectangle in $O(logn)$ time. Instead, the data structure will return a 'good' fitting rectangle, without the guarantee that no better fitting rectangle is available.

The main reason for using the alternative orderings however, is the strong disruption of the priority sequence generated compared to using the decreasing width ordering. While the optimal time data structures cause a slightly different disruption compared to using the original data structure, their overall solution quality is comparable. Furthermore, the difference in computation time for large problem instances will be huge, as we change from $O(n^2)$ to $O(nlogn)$ time complexity. Therefore we propose the usage of these datastructures in a new optimal time three-way heuristic (notice the absence of the 'best fit' part). With respect to the three-way best fit heuristic, we will use the $O(logn)$ sheet skyline data structure to improve its performance, while maintaining the original rectangle selection procedure.

4. COMPUTATIONAL RESULTS

We discuss the performance of the best fit heuristic and its optimal time variant on a set of benchmark problems from the literature (Table 1). Due to the very large computation times needed to solve the $i19$ and $i20$ instances from Imahori and Yaguira (2010) for the original and three way best fit heuristic, these instances were only used for comparing scalability. All the other experiments ignored these instances.

Data source	#Problems	#Rectangles
Hopper (2000)	70	17 to 199
Hopper and Turton (2001)	21	16 to 197
Burke et al. (2004)	13	10 to 3152
Beasley (1985)	12	10 to 50
Imahori and Yagiura (2010)	170	2^4 to 2^{20}

Table 1: *Benchmarks from the literature.*

By combining the different ordering strategies and placement poli-

cies into a three-way best fit, we can improve the solution quality. Using the three-way best fit heuristic produces significantly better results compared to the original best fit heuristic. Statistical analysis using a T-test showed a certainty of more than 99.9999% that the three-way best fit outperforms the original best fit heuristic. When looking at the optimal time variant, we find the results are not significantly different from those of the standard three way best fit heuristic ($p-value = 0.158$). Especially for the larger problem instances, we can see that both heuristics produce very similar results. This is confirmed by a statistical analysis which shows only a 70.79% confidence interval that the heuristics perform significantly different on the instances from Imahori and Yagiura [10]. When considering the largest problem sizes only, $i14$ to $i18$, this confidence interval becomes even smaller ($p-value = 0.933$).

The test set from Imahori and Yagiura [10] contains instances with up to 2^{20} rectangles, and allows for an easy comparison of the scalability of the different heuristics. Figure 2 shows the computation times for the original best fit heuristic, three-way best fit heuristic and optimal time three-way heuristic on this test set. The three-way best fit heuristic clearly benefits from using the optimized gap location process, as the computation times are lower than those of the original implementation for all but the largest instances. Note that the three-way best fit heuristic solves each problem 18 times, which is 6 times more than the original best fit heuristic. We can also see that using the optimal time implementation [10] makes the heuristic significantly faster for all but the smallest test instances. For instances with 2^{18} items, the optimal time three-way heuristic requires only 1.60% of the time needed by the original best fit heuristic to solve the same problem, while obtaining a better result. For these instances, the computation time needed by the optimal time three way heuristic is only 0.46% of the time needed by the three-way best fit heuristic. Furthermore, the optimal time heuristic performs slightly better than the three-way best fit heuristic on these instances.

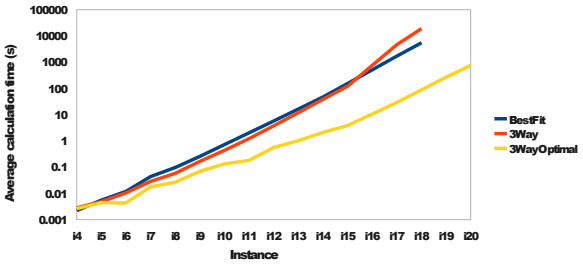


Figure 2: *Average computation times of the original best fit, three-way best fit and optimal time three-way heuristic, for the Imahori and Yagiura instances.*

5. CONCLUSIONS

In this abstract we presented several enhancements to the best fit heuristic from Burke et al. [9]. We introduced new placement policies and additional orderings of the items in order to obtain better solutions for rectangular stock-cutting problem. These enhancements allow for a significantly better performance compared to the original best fit heuristic, on a large test set from the literature. As the addition of the new placement policies and orderings increased the computation time of the heuristic, a more efficient implementation of the heuristic was used. The three-way best fit heuristic uses a more efficient way of storing and locating the gaps [10] to reduce its computational complexity. Due to this improvement, this heuristic has smaller computation times than the original best fit heuristic for all but the largest problem instances. Next, we further improved the scalability of the heuristic, by also applying the rectangle selection procedure from [10]. This resulted in an optimal time three-way heuristic, with a slightly altered rectangle selection that no longer guarantees the selection of the best fitting rectangle for a given gap. Due to this changed rectangle selection procedure, the heuristic obtains slightly, but not significantly, different results than the three way best fit heuristic. The optimal time three-way heuristic is, however, much faster than the three-way best fit heuristic on all but the smallest instances. For instances with 2^{18} items, the optimal time three-way heuristic requires only 0.46% of the time required by the three-way best fit heuristic. Therefore, we propose the usage of the optimal time three-way heuristic when small computation times are important. When the quality of the solutions is more important than the computation times, combined usage of both three-way heuristics is advised when no more than 2^{16} items need to be placed. When more than 2^{16} items need to be placed, the optimal time three-way heuristic is recommended as it performs best both with respect to average solution quality and computation time.

6. ACKNOWLEDGEMENTS

Research funded by a Ph.D. grant of the Agency for Innovation by Science and Technology (IWT)

7. REFERENCES

[1] H. Dyckhoff, "A typology of cutting and packing problems," *European Journal of Operational Research*, vol. 44, no. 2, pp. 145–159, January 1990.

[2] G. Wascher, H. Hausner, and H. Schumann, "An improved typology of cutting and packing problems," *European Journal of Operational Research*, vol. 183, no. 3, pp. 1109–1130, December 2007.

[3] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness (Series of Books in the Mathematical Sciences)*. W. H. Freeman & Co Ltd, January 1979.

[4] P. C. Gilmore and R. E. Gomory, "A linear programming approach to the cutting-stock problem," *OPERATIONS RESEARCH*, vol. 9, no. 6, pp. 849–859, November 1961.

[5] C. H. Dagli and P. Poshyanonda, "New approaches to nesting rectangular patterns," *Journal of Intelligent Manufacturing*, vol. 8, no. 3, pp. 177–190, May 1997.

[6] B. S. Baker, E. G. Coffman_jr, and R. L. Rivest, "Orthogonal packings in two dimensions," *SIAM Journal on Computing*, vol. 9, no. 4, pp. 846–855, 1980.

[7] Chazelle, "The bottomn-left bin-packing heuristic: An efficient implementation," *IEEE Transactions on Computers*, vol. C-32, no. 8, pp. 697–707, August 1983.

[8] S. Jakobs, "On genetic algorithms for the packing of polygons," *European Journal of Operational Research*, vol. 88, no. 1, pp. 165–181, January 1996.

[9] E. K. Burke, G. Kendall, and G. Whitwell, "A new placement heuristic for the orthogonal stock-cutting problem," *Operations Research*, vol. 52, pp. 655 – 671, 2004.

[10] S. Imahori and M. Yagiura, "The best-fit heuristic for the rectangular strip packing problem: An efficient implementation and the worst-case approximation ratio," *Computers & Operations Research*, vol. 37, no. 2, pp. 325–333, February 2010.

[11] A. R. Babu and N. R. Babu, "Effective nesting of rectangular parts in multiple rectangular sheets using genetic and heuristic algorithms." *International Journal of Production Research*, vol. 37, no. 7, p. 1625, 1999.

[12] E. Hopper and B. Turton, "An empirical investigation of meta-heuristic and heuristic algorithms for a 2d packing problem," *European Journal of Operational Research*, vol. 128, no. 1, pp. 34–57, January 2001.

[13] ——, "A genetic algorithm for a 2d industrial packing problem," *Comput. Ind. Eng.*, vol. 37, no. 1-2, pp. 375–378, 1999.

[14] B. Kroger, "Guillotineable bin packing: A genetic approach," *European Journal of Operational Research*, vol. 84, no. 3, pp. 645–661, August 1995.

[15] E. K. Burke, G. Kendall, and G. Whitwell, "A simulated annealing enhancement of the best-fit heuristic for the orthogonal stock-cutting problem," *INFORMS Journal on Computing*, vol. 21, no. 3, pp. 505–516, February 2009.

Bi-dimensional Bin-Packing Problem: A Multiobjective Approach

A. Fernández * C. Gil * R. Baños * A. L. Márquez * M.G. Montoya * M. Parra *

* University of Almería
Carretera de Sacramento s/n, Cañada de San Urbano, 04120 Almería, Spain
{afdezmolina, cgilm, rbanos, almarquez, dgil, mariaparra}@ual.es

ABSTRACT

The bin-packing problem (BPP) and its multi-dimensional variants, have a large number of practical applications, including production planning, project selection, multiprocessor scheduling, packing objects in boxes, etc. The two-dimensional bin packing (2D-BPP) consists of packing a collection of objects (pieces) in the minimum number of bins (containers). This paper works with an extending of the classical single-objective formulation to cope with other designing objectives. It presents a new multi-objective memetic algorithm that uses a population of individuals (agents) that are optimized using evolutionary operators (mutation and crossover) and a local-search optimizer specially designed to solve the MO-2DBPP. The Pareto-optimization concept is used in the selection process. Results obtained in several test problems show the good performance of the memetic algorithm in comparison with other previously proposed approaches.

Keywords: Two-dimensional bin packing problem, Memetic algorithm, Multi-objective optimization

1. INTRODUCTION

The bin-packing problem (BPP) and its multi-dimensional variants, have a large number of practical applications in industry (e.g. cutting stock), in computer systems (e.g. assignment of segments of track on disks), in machine scheduling (e.g. minimizing the number of machines necessary for completing all tasks by a given deadline), etc. [1]. The traditional two-dimensional BPP (2DBPP) [2] consists of packing a collection of objects, characterized by having different heights and widths, in the minimum number of bins (containers). The family of bin packing problems is included in the category of NP-hard problems [3], which implies that there is no known method to obtain the optimal solution in a polynomial time. Recently, some authors have proposed multi-objective formulations of the 2DBPP (MO-2DBPP) that consider other objectives to minimize in addition to the number of bins. One of these multi-objective formulations with applications in container loading, tractor trailer trucks, pallet loading, cargo airplanes, etc. consists of minimizing not only the number of bins used to store the pieces, but also the imbalance of the objects according to the centre of gravity of the bin. This paper presents a new multi-objective [4] memetic algorithm that uses a population of individuals (agents) that are optimized using evolutionary operators (mutation and crossover) and a local-search optimizer specially designed to solve the MO-2DBPP. The Pareto-optimization concept [5] is used in the selection process.

2. MULTI-OBJECTIVE TWO-DIMENSIONAL BIN-PACKING PROBLEM

Most papers dealing with the 2DBPP try to solve single-objective formulations, where the aim is to minimize the number of bins needed to pack all the objects. Recently, other authors have proposed simultaneously optimizing other objectives. In particular, Liu et al. [6] applied particle swarm optimization to solve the multi-objective two-dimensional bin packing problem (MO-2D-BPP), by considering minimizing, not only the number of bins, but also the imbalance of the bins according to a centre of gravity. This formulation is described as follows: Given a set of n rectangular objects where h_i, w_i, and γ_i are the height, width and weight of object i, respectively (i=1,2,...,n), and given an unlimited number of bins, all of which have a height H, width W and centre of gravity (λ_x,λ_y) the goal is to insert all the objects without overlap in the minimum number of bins (*nBIN*), with the centre of gravity (*CG*) of the bins as close as possible to the desired *CG*. The desired *CG* in this case is the bottom of the bin, and therefore, the aim is to minimize the average euclidean distance d_i between the *CG* of the objects stored in the bin with respect to the *CG* of the bin. The definition of the centre of gravity is provided below:

$$CG = \frac{1}{nBin}\sum_{j=1}^{nBin}\sqrt{(\lambda_{x,j}-\lambda_{d,x})^2+(\lambda_{y,j})^2} \quad (1)$$

$$\lambda_{x,j} = \frac{\sum_{i=1}^{n} X_{ij}x_i\gamma_i}{\sum_{i=1}^{n}\gamma_i} \qquad \lambda_{y,j} = \frac{\sum_{i=1}^{n} X_{ij}y_i\gamma_i}{\sum_{i=1}^{n}\gamma_i} \quad (2)$$

where:

h_i, w_i, and γ_i: height, width and weight of item i;

x_i and y_i: center of gravity of item i in positions x and y;

$X_{ij} \in \{0,1\}$, where $i = \{1,...,I\}, j = \{1,...J\}$. If item j is assigned to bin i, $X_{ij} = 1$, otherwise $X_{ij} = 0$;

H and W: height and width of bins;

$(\lambda_{x,j}, \lambda_{y,j})$: coordinates of the centre of gravity of bin j;

$\lambda_{d,x}$: desired center of gravity of bin i in direction x.

CG: balance of the bins according to a centre of gravity (objective 2);

In order to minimize the load balancing of an individual, the fitness function used determines the average balancing of each bin, taking into account the sum of the Euclidean distances from the centre of each object to the desired CG of the bin, and taking into account their weight. Figure 1 offers a graphical description of this second objective in a bin which contains a single object.

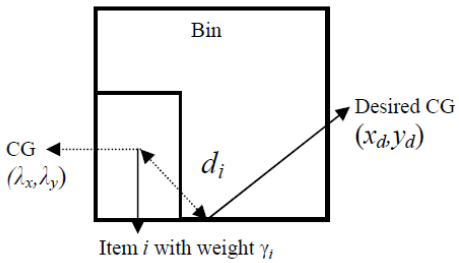


Figure 1: *Graphical representation of load balancing.*

2.1. Description of the operators used in MA2dbpp

Four different mutation operators are used in order to insert objects in the bins using the list of available rectangular spaces. One of these operators (mutation 4) takes some ideas of the strategy recently proposed by Grunert da Fonseca and Fonseca [7] that is based on performing a permutation between two objects of different bins such that the variation is smaller than when a single object is moved from one bin to another one.

- Mutation1: an object is randomly taken from one bin and it is stored in another randomly chosen one only if the available space is large enough. If all the bins have been visited and the storage has not been possible, the object is not inserted.
- Mutation2: an object is randomly chosen from the bin with most available space, and it is stored in another randomly chosen bin only if there is free space. If all the bins have been visited and the storage has not been possible, the object is not inserted.
- Mutation3: an object is randomly chosen from the bin with most available space, and it is stored in the empties remaining bin only if the available space is large enough. If all the bins have been tried, and the storage has not been possible, the object is inserted in a new bin in the lower left corner.
- Mutation4: two objects are randomly taken from different bins and are swaped only if there are free space in the bins.

The selection of agents is carried out by applying tournaments using Pareto-dominance relations [5]. The crossover operator works by taking two random agents (A1, A2) as parents, and creating a child agent (CH) by considering bins of both parents. In particular, CH takes the fullest bin of A1, plus the bins of A2, but discarding the objects already taken from A1 in order not to duplicate objects.

Finally, a new local optimizer is also considered with the aim of reducing the number of bins. This task takes the most occupied bin and tests each available space to determine whether or not an object from the remaining bins can fit.

3. EXPERIMENTAL RESULTS

A set of instances proposed by Berkey and Wang [8] have been used to compare the algorithms. A total of six classes with 20 instances each are randomly generated to determine the performance of the multi-objectives memetic algorithms. The weight γ_i of each piece randomly generated in different ranges, has been added to the benchmark set, as table 1 shows. For each instance, there are 500 items to be packed.

The performance of the multi-objective memetic algorithm (MOMA-2DBPP) has been compared with other algorithms, using the

Class	1	2	3	4	5	6
h_i, w_i	[0,100]	[0,25]	[0,50]	[0,75]	[25,75]	[25,50]
γ_i	[0,20] for instances 1-10 of each class; and [0,100] for instances 11-20					

Table 1: Test benchmarks generated for solving the MO-2DBPP.

test instances with 500 pieces described above. The memetic algorithm was executed with a stop criterion of 1000 generations and a population size of 500 agents.

To compare the different fronts, we use a coverage metric [9]. The coverage $C(A,B)$ computes the relative number of points in set B dominated by the points in set A.

$$C(A,B) = \frac{|\{b \in B \mid \exists a \in A : a \prec b\}|}{|B|} \tag{3}$$

To show the good performance of the algorithm MOMA-2D-BPP, it was compared with a recent evolutionary multi-objective particle swarm optimization algorithm called MOEPSO [6]. Figure 2 shows the Pareto fronts generated by these algorithms for a selected set of instances. It can be observed that most of the solutions of the non-dominated sets obtained by MOMA-2DBPP are below those obtained by MOEPSO, i.e. MOMA-2DBPP obtains better approximations to the true (unknown) Pareto-optimal front, although MOEPSO obtains more extreme solutions in some test instances.

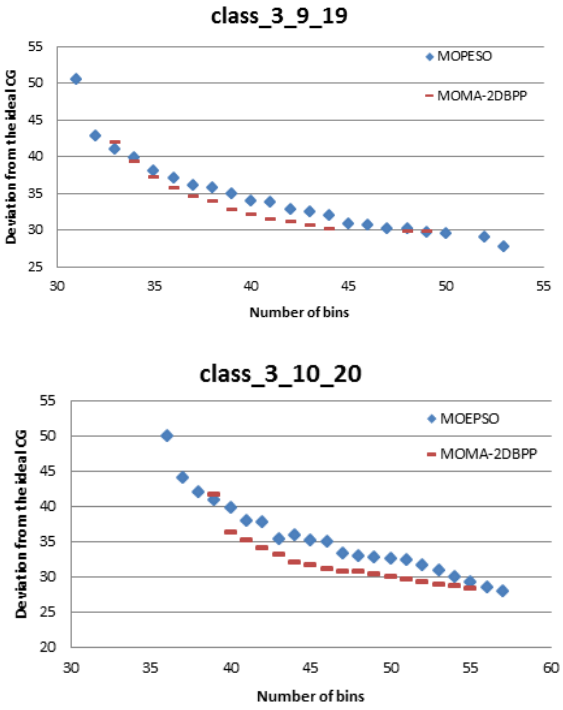


Figure 2: *Pareto front of MOMA-2DBPP and MOEPSO.*

Table 2 shows a comparison of both algorithms for previous instances. The coverage metric has been used to compare the Pareto fronts generated by each algorithm. MOMA-2DBPP algorithm achieves better results than MOEPSO for the two instances, since the coverage metric of the memetic algorithm is higher than MOEPSO in both instances which reinforces the previous conclusions obtained from the graphics displayed above.

	class_3_9_19		class_3_10_20	
	MOEPSO	MOMA	MOEPSO	MOMA
MOEPSO	-	0.20	-	0.05
MOMA	0.52	-	0.77	-

Table 2: Comparison between MOEPSO and MOMA-2BPP in terms of coverage metric.

4. CONCLUSION

This paper presents a memetic algorithm that aims to improve the performance of other published algorithms when solving single-objective and multi-objective formulations of the two-dimensional bin-packing problem with rotations. The memetic algorithm here implemented uses several search operators specifically designed to solve this problem. The multi-objective implementation, MOMA-2DBPP is compared with a multi-objective particle swarm optimization algorithm, MOEPSO. Results obtained in the multi-objective formulation show the good behavior of MOMA-2DBPP, which obtains better results than MOEPSO in terms of coverage metric. The results obtained by the memetic algorithm of this complex problem reinforce the previous conclusions of other authors about the good performance of this meta-heuristic to solve NP-hard optimization problems. Future research should be focused on extending the memetic algorithm for the three-dimensional variants of bin-packing [10], which also have many practical applications in real problems. Despite that, the load balancing in two dimensions can be applied to real world problems, where height does not influence, for instance the storage of pallets.

4.1. Acknowledgements

This work has been financed by the Spanish Ministry of Innovation and Science (TIN2008-01117) and the Excellence Project of Junta de Andalucía (P07-TIC02988), in part financed by the European Regional Development Fund (ERDF).

5. REFERENCES

[1] H. L. Ong, M. J. Magazine, and T. S. Wee, "Probabilistic analysis of bin packing heuristics," *OPERATIONS RESEARCH*, vol. 32, no. 5, pp. 983–998, 1984. [Online]. Available: http://or.journal.informs.org/cgi/content/abstract/32/5/983

[2] E. Hopper and B. C. H. Turton, "An empirical investigation of meta-heuristic and heuristic algorithms for a 2d packing problem," *European Journal of Operational Research*, vol. 128, no. 1, pp. 34 – 57, 2001. [Online]. Available: http://www.sciencedirect.com/science/article/B6VCT-41Y1XYH-3/2/73392e0f11c162878430f67e02d8349d

[3] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness (Series of Books in the Mathematical Sciences)*, first edition ed. W. H. Freeman & Co Ltd, January 1979. [Online]. Available: http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/0716710455

[4] P. J. F. Carlos M. Fonseca, "Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization," pp. 416–423, 1993.

[5] D. E. Goldberg, "Genetic algorithms in search, optimization and machine learning," 1989.

[6] D. Liu, K. Tan, C. Goh, and W. Ho, "On solving multiobjective bin packing problems using particle swarm optimization," in *Evolutionary Computation, 2006. CEC 2006. IEEE Congress on*, 2006, pp. 2095 –2102.

[7] C. F. V. Grunert da Fonseca, "The attainment-function approach to stochastic multiobjective optimizer assessment and comparison," in *Experimental Methods for the Analysis of Optimization Algorithms*, T. Bartz-Beielstein, Ed. Springer, [2010 to apperar].

[8] J. O. Berkey and P. Y. Wang, "Two-dimensional finite bin-packing algorithms," *The Journal of theOperational Research Society*, vol. 38, no. 5, pp. 423–429, May, 1987.

[9] E. Zitzler, "Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications," Ph.D. dissertation, ETH Zurich, Switzerland, 1999. [Online]. Available: http://www.tik.ethz.ch/~sop/publications/

[10] A. Lodi, S. Martello, and D. Vigo, "Heuristic algorithms for the three-dimensional bin packing problem," *European Journal of Operational Research*, vol. 141, no. 2, pp. 410–420, September 2002. [Online]. Available: http://ideas.repec.org/a/eee/ejores/v141y2002i2p410-420.html

A recursive partitioning approach for generating unconstrained two-dimensional non-guillotine cutting patterns

Ernesto G. Birgin * Rafael D. Lobato * Reinaldo Morabito †

* Department of Computer Science, Institute of Mathematics and Statistics, University of São Paulo
Rua do Matão 1010, Cidade Universitária, 05508-090 São Paulo, SP, Brazil
{egbirgin,lobato}@ime.usp.br

† Department of Production Engineering, Federal University of São Carlos
Via Washington Luiz km. 235, 13565-905, São Carlos, SP, Brazil
morabito@ufscar.br

ABSTRACT

In this study, a dynamic programming approach to deal with the unconstrained two-dimensional non-guillotine cutting problem is presented. The method extends the recently introduced recursive partitioning approach for the manufacturer's pallet loading problem. The approach involves two phases and uses bounds based on unconstrained two-staged and non-staged guillotine cutting. The method is able to find the optimal cutting pattern of a large number of problem instances of moderate sizes known in the literature and a counterexample for which the approach fails to find known optimal solutions was not found. For the instances that the required computer runtime is excessive, the approach is combined with simple heuristics to reduce its running time. Detailed numerical experiments show the reliability of the method.

Keywords: Cutting and packing, Two-dimensional non-guillotine cutting pattern, Dynamic programming, Recursive approach, Distributor's pallet loading problem

1. INTRODUCTION

In the present paper, we study the generation of *two-dimensional non-guillotine cutting (or packing) patterns*, also referred by some authors as two-dimensional knapsack problem or two-dimensional distributor's pallet loading. This problem is classified as 2/B/O/ according to Dyckhoff's typology of cutting and packing problems [1], and as two-dimensional rectangular Single Large Object Packing Problem (SLOPP) based on Waescher et al.'s typology [2]. Besides the inherent complexity of this problem (it is NP-hard [3]), we are also motivated by its practical relevance in different industrial and logistics settings, such as in the cutting of steel and glass stock plates into required sizes, the cutting of wood sheets and textile materials to make ordered pieces, the loading of different items on the pallet surface or the loading of different pallets on the truck or container floor, the cutting of cardboards into boxes, the placing of advertisements on the pages of newspapers and magazines, the positioning of components on chips when designing integrated circuit, among others.

Given a large rectangle of length L and width W (i.e. a stock plate), and a set of rectangular pieces grouped into m different types of length l_i, width w_i and value v_i, $i = 1,\ldots,m$ (i.e. the ordered items), the problem is to find a cutting (packing) pattern which maximizes the sum of the values of the pieces cut (packed). The cutting pattern is referred as two-dimensional since it involves two relevant dimensions, the lengths and widths of the plate and pieces. A feasible two-dimensional pattern for the problem is one in which the pieces placed into the plate do not overlap each other, they have to be entirely inside the plate, and each piece must have one edge parallel to one edge of the plate (i.e., an orthogonal pattern). In this paper we assume that there are no imposed lower or upper bounds on the number of times that each type of piece can be cut from the plate; therefore, the two-dimensional pattern is called *unconstrained*.

Without loss of generality, we also assume that the cuts are infinitely thin (otherwise we consider that the saw thickness was added to L, W, l_i, w_i), the orientation of the pieces is fixed (i.e., a piece of size (l_i,w_i) is different from a piece of size (w_i,l_i) if $l_i \neq w_i$) and that L, W, l_i, w_i are positive integers. We note that if the 90°-rotation is allowed for cutting or packing the piece type i of size (l_i,w_i), this situation can be handled by simply considering a fictitious piece type $m+i$ of size (w_i,l_i) in the list of ordered items, since the pattern is unconstrained. Depending on the values v_i, the pattern is called *unweighted*, if $v_i = \gamma l_i w_i$ for $i = 1,\ldots,m$ and $\gamma > 0$ (i.e., proportional to the area of the piece), or *weighted*, otherwise. Moreover, we assume that the unconstrained two-dimensional cutting pattern is *non-guillotine* as it is not limited by the guillotine type cuts imposed by some cutting machines.

In the present paper we extend a *Recursive Partitioning Approach* presented in [4] for the manufacturer's pallet loading to deal with the unconstrained two-dimensional orthogonal non-guillotine cutting (unweighted and weighted, without and with piece rotation). This Recursive Partitioning Approach combines refined versions of both the *Recursive Five-block Heuristic* presented in [5, 6] and the *L-approach* for cutting rectangles from larger rectangles and L-shaped pieces presented in [7, 8]). This combined approach also uses bounds based on unconstrained two-staged and non-staged guillotine cutting patterns. The approach was able to find an optimal solution of a large number of problem instances of moderate sizes known in the literature and we were unable to find an instance for which the approach fails to find a known or proved optimal solution. For the instances that the required computer runtimes were excessive, we combined the approach with simple heuristics to reduce its running time.

2. DESCRIPTION OF THE ALGORITHM

The Recursive Partitioning Algorithm presented here is an extension of the algorithm described in [4] for the manufacturer's pallet loading problem. It has basically two phases: in phase 1 it applies a recursive five-block heuristic based on the procedure presented in [5] and in phase 2 it uses an L-approach based on a dynamic programming recursive formula presented in [7, 8]. Firstly, phase 1 is executed and, if a certificate of optimality is not provided by the

Recursive Five-block Heuristic, then phase 2 is executed. Additionally, information obtained in phase 1 is used in phase 2 in at least two ways, according to [4]. If an optimal solution was already found for a subproblem in phase 1, it is not solved again in phase 2, improving the performance of phase 2. Moreover, having the information obtained in phase 1 at hand, phase 2 is often able to obtain better lower bounds for its subproblems than the ones provided by homogeneous cuttings, therefore improving the performance of phase 2. These two phases are detailed in the sequel.

2.1. Phase 1

In phase 1, the Recursive Five-block Heuristic divides a rectangle into five (or less) smaller rectangles in a way that is called *first-order non-guillotine cut* [9]. Figure 1 illustrates this kind of cut represented by a quadruple (x_1, x_2, y_1, y_2), such that $0 \leq x_1 \leq x_2 \leq L$ and $0 \leq y_1 \leq y_2 \leq W$. This cut determines five subrectangles $(L_1, W_1), \ldots, (L_5, W_5)$ such that $L_1 = x_1$, $W_1 = W - y_1$, $L_2 = L - x_1$, $W_2 = W - y_2$, $L_3 = x_2 - x_1$, $W_3 = y_2 - y_1$, $L_4 = x_2$, $W_4 = y_1$, $L_5 = L - x_2$ and $W_5 = y_2$. Each rectangle is recursively cut unless the (sub)problem related to this rectangle has already been solved.

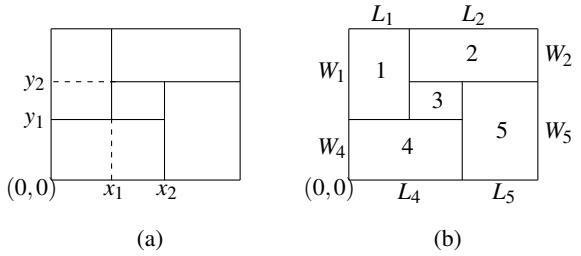


Figure 1: Representation of a first-order non-guillotine cut.

2.2. Phase 2

Phase 2 of the Recursive Partitioning Approach applies the L-approach [7, 8, 4] which is based on the computation of a dynamic programming recursive formula [7]. This procedure divides a rectangle or an L-shaped piece into two L-shaped pieces. An L-shaped piece is determined by a quadruple (X, Y, x, y), with $X \geq x$ and $Y \geq y$, and is defined as the topological closure of the rectangle whose diagonal goes from $(0,0)$ to (X,Y) minus the rectangle whose diagonal goes from (x,y) to (X,Y). Figure 2 depicts the nine possible divisions [4] of a rectangle or an L-shaped piece into two L-shaped pieces.

2.3. Heuristics for large problems

The generation of all patterns by the Recursive Partitioning Approach may be prohibitive for large instances. Moreover, the amount of memory required by these algorithms may not be available. For this reason, we propose heuristics that reduce both the time and memory requirements of the algorithms. These procedures, however, may lead to a loss of quality of the solution found. Since the time and memory complexities of generating all possible cuttings highly depends on the sizes of the integer conic combinations and raster points sets, we can significantly reduce time and memory requirements in two ways: (i) by limiting the search depth of the recursions; and (ii) by replacing the integer conic combinations and raster points sets by smaller sets.

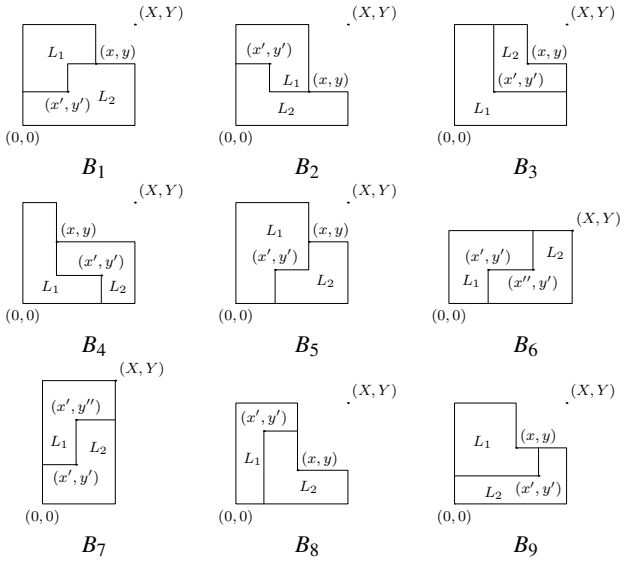


Figure 2: Subdivisions of an L-shaped piece into two L-shaped pieces.

3. NUMERICAL EXPERIMENTS

We implemented the Recursive Partitioning Approach and its heuristic counterpart for the unconstrained two-dimensional non-guillotine cutting problem. The algorithms were coded in C/C++ language. The computer implementation of the algorithms as well as the data sets used in our experiments and the solutions found are publicly available for benchmarking purposes at [10]. In the numerical experiments, we considered 95 problem instances found in the literature. Extensive numerical experiments evaluating the proposed method can be found in [11], where the whole material of the present extended abstract is present in detail.

4. CONCLUDING REMARKS

While a large number of studies in the literature have considered staged and non-staged two-dimensional guillotine cutting problems, much less studies have considered two-dimensional non-guillotine cutting problems (constrained and unconstrained), and only a few of them have proposed exact methods to generate non-guillotine patterns. Moreover, most of the approaches (exact and heuristic) for non-guillotine cutting (or packing) were developed for the constrained problem, which can be more interesting for certain practical applications with relatively low demands of the ordered items. However, part of these methods may not perform well when solving the unconstrained problem. On the other hand, the unconstrained problem is particularly interesting for cutting stock applications with large-scale production and weakly heterogeneous items, in which the problem plays the role of a column generation procedure.

This study presented a Recursive Partitioning Approach to generate unconstrained two-dimensional non-guillotine cutting (or packing) patterns. The approach was able to find the optimal solution of a large number of moderate-sized instances known in the literature and we were unable to find a counterexample for which the approach fails to find a known optimal solution. To cope with large instances, we combined the approach with simple heuristics to reduce its computational efforts. For moderate-sized instances, both the five-block and the L-Algorithm phases of the approach seem to be promising alternatives for obtaining reasonably good or opti-

mal non-guillotine solutions under affordable computer runtimes, whereas for larger instances, the guillotine or the five-block phase may be preferable, depending on the definition of an acceptable time limit. An interesting perspective for future research is to extend the Recursive Partitioning Approach to deal with constrained two-dimensional non-guillotine cutting.

5. REFERENCES

[1] H. Dyckhoff, "A typology of cutting and packing problems," *European Journal of Operational Research*, vol. 44, pp. 145–159, 1990.

[2] G. Wäescher, H. Haußner, and H. Schumann, "An improved typology of cutting and packing problems," *European Journal of Operational Research*, vol. 183, pp. 1109–1130, 2007.

[3] J. E. Beasley, "A population heuristic for constrained two-dimensional non-guillotine cutting," *European Journal of Operational Research*, vol. 156, pp. 601–627, 2004.

[4] E. G. Birgin, R. D. Lobato, and R. Morabito, "An effective recursive partitioning approach for the packing of identical rectangles in a rectangle," *Journal of the Operational Research Society*, vol. 61, pp. 306–320, 2010.

[5] R. Morabito and S. Morales, "A simple and effective recursive procedure for the manufacturer's pallet loading problem," *Journal of the Operational Research Society*, vol. 49, pp. 819–828, 1998.

[6] ——, "Erratum to 'A simple and effective recursive procedure for the manufacturer's pallet loading problem'," *Journal of the Operational Research Society*, vol. 50, pp. 876–876, 1999.

[7] L. Lins, S. Lins, and R. Morabito, "An L-approach for packing (l,w)-rectangles into rectangular and L-shaped pieces," *Journal of the Operational Research Society*, vol. 54, pp. 777–789, 2003.

[8] E. G. Birgin, R. Morabito, and F. H. Nishihara, "A note on an L-approach for solving the manufacturer's pallet loading problem," *Journal of the Operational Research Society*, vol. 56, pp. 1448–1451, 2005.

[9] M. Arenales and R. Morabito, "An and/or-graph approach to the solution of two-dimensional non-guillotine cutting problems," *European Journal of Operational Research*, vol. 84, pp. 599–617, 1995.

[10] "http://www.ime.usp.br/∼egbirgin/packing/."

[11] E. G. Birgin, R. D. Lobato, and R. Morabito, "Generating unconstrained two-dimensional non-guillotine cutting patterns by a recursive partitioning algorithm," *Journal of the Operational Research Society*, 2011, to appear.

A Complete Search Method For Relaxed Traveling Tournament Problem

Filipe Brandão * João Pedro Pedroso * †

* Faculdade de Ciências, Universidade do Porto
Rua do Campo Alegre, 4169-007 Porto, Portugal
fdabrandao@dcc.fc.up.pt jpp@fc.up.pt

† INESC Porto
Rua Dr. Roberto Frias 378, 4200-465 Porto, Portugal

ABSTRACT

The Traveling Tournament Problem (TTP) is a sports scheduling problem that includes two major issues in creating timetables: home/away pattern feasibility and travel distance. In this problem the schedule must be compact: every team plays in every time slot. However, there are some sports leagues that have both home/away pattern restrictions and distance limits, but do not require a compact schedule. In such schedules, one or more teams can have a bye in any time slot. This leads us to a variant of the problem: the Relaxed Traveling Tournament Problem (RTTP). We present a complete search method to solve this problem based on branch-and-bound, metaheuristics and dynamic programming.

Keywords: Complete search, Dynamic programming, Metaheuristics, Branch-and-bound

1. INTRODUCTION

The advances in modeling the combinatorial structure of sports schedules and their solution, together with the increasing practical requirements for schedules by real sports leagues has increased the interest in computational methods for creating them.

The key issues for constructing a schedule are travel distance and home/away pattern restrictions. While teams wish to reduce the total amount they travel, they are also concerned with more traditional issues with respect with home and away patterns.

The Traveling Tournament Problem (TTP) abstracts the key issues in creating a schedule that combines home/away pattern constraints and travel distance minimization. Either home/away pattern constraints and travel distance minimization are reasonably easy to solve, but the combination of them makes this problem very difficult. This problem was proposed in [1].

In TTP the schedule must be compact: every team plays in every time slot; however, there are some sports leagues that have both home/away pattern restrictions and distance limits, but do not require a compact schedule. This leads us to a new problem: the Relaxed Traveling Tournament Problem. This variant of the TTP was proposed by Renjun Bao and Michael Trick [2]. As in this variant the schedule is not compact, teams have byes (i.e., slots where they do not play) in their schedule. The objective is to minimize the travel distance, and the teams are allowed to have a fixed number K of byes.

2. THE TRAVELING TOURNAMENT PROBLEM

In the Traveling Tournament Problem, there is a even number n of teams, each with a home venue. The teams wish to play a round robin tournament, whereby each team will play against every other team twice, once at each team's home venue. This means that $2(n-1)$ slots, or time periods, are required to play a double round robin tournament. There are exactly $2(n-1)$ time slots available to play these games, so every team plays in every time slot. Associated with a TTP instance is a n by n distance matrix D, where D_{ij} is the distance between the venues of team i and team j.

Each team begins at its home site and travels to play its games at the chosen venues. At the end of the schedule each team then returns (if necessary) to its home site.

Consecutive games for a team constitute a road trip; consecutive home games are a home stand. The length of a road trip or home stand is the number of opponents played (not the travel distance).

The TTP is defined as follows:

Input: n, the number of teams; D, an n by n symmetrical distance matrix; l, u integer parameters.

Output: A double round robin tournament on the n teams such that:

- the length of every home stand and road trip is between l and u inclusive;
- games between the same opponents cannot happen in consecutive time slots, which is called no repeater constraint;
- the total distance traveled by the teams is minimized.

The parameters l and u define the trade-off between distance and pattern considerations. For $l = 1$ and $u = n-1$, a team may take a trip equivalent to a traveling salesman tour. For small u, teams must return home often, so the distance traveled will increase. Usually $l = 1$ and $u = 3$, which means that each team cannot play more than three consecutive home games or three consecutive road games.

The solution of the TTP has proven to be a computational difficult challenge. For many years, the six-team instance NL6, available in [3], was the largest instance solved to a provable optimum. In 2008, NL8 was solved; NL10 was solved in 2009. This leaves twelve teams as the next unsolved instance, which is a significantly small league size for such a simple problem description.

3. THE RELAXED TRAVELING TOURNAMENT PROBLEM

The goal in the TTP is to find a compact schedule: the number of time slots is equal to the number of games each team plays. This forces every team to play in every time slot. However, there are some sports leagues that have both home/away pattern restrictions and distance limits but do not require a compact schedule. In such schedules, one or more teams can have a bye in any time slot. This leads us to the Relaxed Traveling Tournament Problem (RTTP).

In this variant of the TTP, instead of fixing the schedule length to be $2(n-1)$, we let the schedule length be $2(n-1)+K$ for some integer $K \geq 0$. For a given K, the problem is called K-RTTP. For $K = 0$, the RTTP is just the TTP. For $K > 0$, each team has K slots in which it does not play.

Byes are ignored in determining the length of a homestand or road-trip, and in determining whether a repeater has occured. This allows that TTP's solutions are feasible for the K-RTTP for every $K \geq 0$ (in fact, K_1-RTTP's solutions are feasible for K_2-RTTP if $K_1 \leq K_2$).

4. SOLUTION METHODOLOGY

For solving the RTTP one has to deal both with feasibility concerns (the home and away pattern) and optimization concerns (the travel distance); this combination makes this problem very difficult to solve to a provable optimal.

One of the most successful methods of solving the TTP is an algorithm which combines an iterative deepening algorithm [4] with depth-first branch-and-bound [5]. Other approaches include a simulated annealing metaheuristic [6], representing the problem with hard and soft constraints, and exploring both feasible and infeasible schedules based on a large neighborhood.

Our solution methodology for RTTP is a complete search-method, putting in place several tools: branch-and-bound (the main method), metaheuristics (for trying to improve bounds), and dynamic programming (to compute lower bounds quickly). The way we combined these tools is described below in Algorithm 1.

So far, the largest instance solved to a provable optimal was NL4; our method allowed us to solve NL6 very quickly and NL8. For larger instances, the method was unable to reach solutions better than the best known solutions for the TTP.

Algorithm 1: Hybrid RTTP-Solver

```
1:  UB ← ∞
2:  S ←[empty schedule]
3:  while not empty(S) do
4:      u ← pop(S)
5:      if final(u) then
6:          v ← hill-climbing(u)
7:          if cost(v) < UB then
8:              UB ← cost(v)
9:          end if
10:     else if cost(u)+ILB(u) < UB then
11:         for all v ∈ branch(u) do
12:             push(S, v)
13:         end for
14:     end if
15: end while
```

4.1. Branch-and-bound

If solutions for the RTTP are generated team by team (i.e., fix all the games of a team before moving to other team), it becomes very difficult to check all the constraints of the problem. E.g., when we fix a game for a team, we are also fixing a game for another team (the first's opponent) in the same round; however we can not apply, for example, the restriction of home/away pattern to the opponent team, due to not having information about previous games.

Therefore, solutions are generated round by round: all the games of one round are fixed before moving to the subsequent round. The advantage of this order is that we can verify restrictions earlier, avoiding the exploration of significant parts of the branch-and-bound tree.

To enumerate solutions we use the following method

1. start at the first round;
2. for each team, if a game is not scheduled yet, pick each possible opponent, and try to schedule a game;
3. after trying all opponents, try to use a bye;
4. when the schedule for the current round is complete, repeat this process in the following round, until completing the schedule.

For trimming off non-optimal candidates from the branch-and-bound tree, we use the current cost plus the Independent Lower Bound (ILB) for the remaining games of each team, as described below.

4.2. Independent Lower Bound and Dynamic Programming

If we calculate the optimal schedule (that minimizes travel distance) for one team without taking into account the other teams' schedule, we have a lower bound to the distance traveled by that team. The sum over the n teams of the distances associated with their independent optimal schedule provides a simple but strong lower bound. This is called Independent Lower Bound (ILB) as was first proposed in [7].

To calculate this lower bound, we need to know: the team, the current location, the number of remaining home games, the list of remaining away games, the current number of consecutive home/away games. This information can be used as the state in dynamic programming. Exploiting some symmetries, a small table suffices for holding this information; e.g., a 108MB table is enough for the twelve teams problem NL12, and it can be computed very quickly.

4.3. Metaheuristics

Everytime we find a new solution inside the branch-and-bound tree, we apply a hill climbing metaheuristic to try to improve bounds. When a local optimum is reached, random perturbations are applied to the solution; this perturbation and hill climbing process is repeated a number of times (100, in our experiment).

To generate the neighbours for the current solution, we use three from the five transformations proposed in [6]. These movements are:

- $SwapHomes(T_i, T_j)$: Given two teams, their home/away roles in the two scheduled games between them are swapped;
- $SwapRounds(r_k, r_l)$: This move swaps rounds r_k and r_l;
- $SwapTeams(T_i, T_j)$: This move simply swaps the schedule of teams T_i and T_j.

Whenever applying a move leads to an invalid solution, the schedule is discarded. These three moves are not sufficient for exploring the entire search space and, as a consequence, they lead to suboptimal solutions; however, they can lead to better solutions, thereby improving the upper bound.

The use of this metaheuristic to improve bounds is particularly important in big instances, such as NL8, where it allows us to quickly find good solutions sooner, and thus pruning more effectively the branch-and-bound tree. Small instances, such as NL6, can be solved without this component, as in this case the search tree (using only the ILB) is relatively small.

5. COMPUTATIONAL RESULTS

The method proposed in this paper was tested on a subset of the benchmark instances available in [3]. The results obtained are reported in Table 1. The previous best known solutions are reported in Table 2. For the NL8 with two byes, the solution for $K = 1$ was used as initial upper bound ($\star$); for NL8 with with three byes, the previous ($K = 2$) solution provided the initial upper bound ($\star\star$). CPU times were obtained with a (sequential) implementation in the C programming language, in a Quad-Core Intel Xeon at 2.66 GHz, running Mac OS X 10.6.6.

Name	# teams	K	ILB	Optimal Solution	Time
NL4	4	1	8044	8160	0s
NL4	4	2	8044	8160	0s
NL4	4	3	8044	8044	0s
NL6	6	1	22557	23124	10s
NL6	6	2	22557	22557	1s
NL8	8	1	38670	39128	44h
NL8	8	2	38670	38761	208h($\star$)
NL8	8	3	38670	38670	92h($\star\star$)

Table 1: *Results for NL Instances. ILB is the independent lower bound at the root node.*

Name	# teams	K	Solution	Optimal Solution
NL4	4	1	8160	8160
NL4	4	2	8160	8160
NL4	4	3	8044	8044
NL6	6	1	23791	23124

Table 2: *Previous results for NL Instances from Bao and Trick [2].*

6. CONCLUSIONS

The solution of Traveling Tournament Problem has proved to be a computational difficult challenge. The combination of home/away pattern constraints and travel distance minimization makes this problem very difficult. Its relaxed version (RTTP) seems to be even harder to solve to a provable optimum. To tackle this problem, we combined different methods: branch-and-bound, dynamic programming and metaheuristics. These were combined in a careful computer implementation, allowing us to solve to optimality some of the previously open instances.

7. REFERENCES

[1] K. Easton, G. Nemhauser, and M. Trick, "The traveling tournament problem description and benchmarks," 2001.

[2] R. Bao, "Time relaxed round robin tournament and the NBA scheduling problem," Master's thesis, Cleveland State University, 2006.

[3] M. Trick, "Challenge traveling tournament instances," 2011, (accessed January 29, 2011). [Online]. Available: http://mat.gsia.cmu.edu/TOURN

[4] R. E. Korf, "Depth-first iterative-deepening : An optimal admissible tree search," *Artificial Intelligence*, vol. 27, no. 1, pp. 97 – 109, 1985. [Online]. Available: http://www.sciencedirect.com/science/article/B6TYF-47X1JH4-G/2/656a3c8f0a14e8d6ca73a9a996faebfe

[5] D. C. Uthus, P. J. Riddle, and H. W. Guesgen, "Dfs* and the traveling tournament problem," in *CPAIOR*, ser. Lecture Notes in Computer Science, W. J. van Hoeve and J. N. Hooker, Eds., vol. 5547. Springer, 2009, pp. 279–293.

[6] A. Anagnostopoulos, L. Michel, P. V. Hentenryck, and Y. Vergados, "A simulated annealing approach to the traveling tournament problem," *J. of Scheduling*, vol. 9, pp. 177–193, April 2006. [Online]. Available: http://portal.acm.org/citation.cfm?id=1127684.1127697

[7] K. Easton, G. L. Nemhauser, and M. A. Trick, "Solving the travelling tournament problem: A combined integer programming and constraint programming approach," in *PATAT*, ser. Lecture Notes in Computer Science, E. K. Burke and P. D. Causmaecker, Eds., vol. 2740. Springer, 2002, pp. 100–112.

A Hybrid Algorithm for Minimizing Earliness-tardiness Penalties in Parallel Machines

Fulgencia Villa * Ramon Alvarez-Valdes † Jose M. Tamarit †

* Polytechnic University of Valencia
Dept. Applied Statistics and Operations Research and Quality
mfuvilju@eio.upv.es

† University of Valencia
Dept. Statistics and Operations Research
{ramon.alvarez, jose.tamarit}@uv.es

ABSTRACT

We consider the problem of scheduling a set of jobs on a set of identical parallel machines where the objective is to minimize the total weighted earliness and tardiness with respect to a common due date. We propose a hybrid heuristic algorithm, combining priority rules for assigning jobs to machines, local search and Path Relinking, with exact procedures for solving the one-machine subproblems. These exact procedures have been developed by our group in a previous study. The algorithm is compared with the best reported results on the same instances in order to assess the efficiency of the proposed strategy.

Keywords: Scheduling, Earliness-tardiness, Metaheuristics

1. INTRODUCTION

In Just-In-Time scheduling, not only tardiness but also earliness are penalized. Tardy jobs, completed after their due date, result in customer discontent, contract penalties, loss of sales and loss of reputation, but early jobs also have non-desirable effects such as inventory carrying costs, the opportunity cost of the money invested in inventory, storage and insurance costs, and product deterioration. Therefore, criteria involving both earliness and tardiness costs are receiving increased attention in machine scheduling research. In this paper we consider the problem of scheduling a set of jobs on a set of identical parallel machines where the objective is to minimize the total weighted earliness and tardiness with respect to a common due date. In practice, problems with a common due date appear when a set of components are produced to be assembled in a later phase or when a set of products have to be sent together to a client.

The problem can be defined as follows. There are n jobs to be processed on a set of m identical parallel machines, all of them with the same due date d. For each job i, the processing time p_i, the penalty per period of earliness α_i, and the penalty per period of tardiness β_i, are known. No preemption is allowed, all the jobs are available at time zero and the machine is continuously available for work. If we denote the completion time of job i by C_i, the objective is

$$min \sum_i^n \alpha_i E_i + \beta_i T_i,$$

where $E_i = max\{d - C_i, 0\}$ and $T_i = max\{C_i - d, 0\}$.

When dealing with this objective function, two cases can be distinguished. We consider a problem as *non-restrictive* if the optimal cost cannot decrease with extensions to the common due date. In this case we say that the due date is *non-restrictive* (d^l), that is, long enough to allow as many jobs as required to be processed in the interval $(0,d)$. In the *restrictive* case the due date, d^r, affects the optimal schedule because not all the required jobs fit into the interval $(0,d)$. According to the classification system by Graham et al. [1], the problem can be denoted as $P|d_i = d^r|\sum_i(\alpha_i E_i + \beta_i T_i)$. The problem is strongly NP-hard because the basic problem $P||\sum_i w_i C_i$, which is already NP-hard, is a particular case.

The non-restrictive case has been studied by Hall [2] and Sundaraghavan and Ahmed [3]. Chen and Powell [4] proposed a column generation algorithm for $P|d_i = d^l|\sum_i((\alpha_i E_i + \beta_i T_i)$, optimally solving instances of up to 60 jobs. More recently, Rios-Solis y Sourd [5] have studied the restrictive case, developing heuristics based on the efficient exploration of an exponential-size neighborhood. An extensive computational study, using new and existing instances, shows the good performance of the proposed procedures. Kedad-Sidhoum et al. [6] have developed a lower bound and a local search heuristic for the case with distinct due dutes, but their procedures can obviously be appplied to the case of a common due date.

2. SOLVING THE ONE-MACHINE PROBLEM

The one-machine problem has been extensively studied. From previous studies we know that there is always an optimal solution satisfying three conditions:

1 An optimal schedule does not contain any idle time between consecutive jobs.

2 The optimal schedule is V-shaped around the common due date. Jobs completed before or on the due date are scheduled in non-increasing order of p_i/α_i, and jobs starting on or after the due date are scheduled in non-decreasing order of p_i/β_i.

3 In the optimal schedule, either the first job starts at time zero or there is a job finishing on the due date.

According to property 3, we can classify the instances into two categories: those for which the optimal solution has a job finishing on the due date and those where the optimal solution starts at time zero. If both conditions hold for a given instance, it is classified into the first category. We have developed a different quadratic model for each class of problems [7].

2.1. Model 1: Problems in which a job ends on the due date

$$\min \quad \sum_i \alpha_i b_i \sum_{j>i,inB} b_j p_j + \sum_i \beta_i a_i \sum_{j\leq i,inA} a_j p_j \tag{1}$$

$$s.t. \quad \sum_{i=1}^{n} b_i p_i \leq d \tag{2}$$

$$a_i + b_i = 1 \qquad \forall i = 1,2,\ldots,n \tag{3}$$

$$a_i, b_i \in \{0,1\} \qquad \forall i = 1,2,\ldots,n \tag{4}$$

$$b_i = \begin{cases} 1, & \text{if } i \text{ finishes on or before } d \\ 0, & \text{otherwise} \end{cases} \qquad \forall i = 1,2,..,n$$

$$a_i = \begin{cases} 1, & \text{if } i \text{ begins on or after } d \\ 0, & \text{otherwise} \end{cases} \qquad \forall i = 1,2,..,n$$

In this model, as there is always a job finishing on d, all jobs are classified as jobs finishing on or before d (the jobs in set B), and jobs starting on or after d (the jobs in set A). Variables a_i and b_i define whether each job i belongs to A or B. Obviously, $a_i = 1 - b_i$, and constraints (3) are redundant. We only keep both for the clarity of the model. Once the jobs are classified, their relative position in A and B is determined by property 2. Therefore, the order required in the objective function is known. We take advantage of this property by building two ordered lists: the B-order, by non-increasing order of p_i/α_i, and the A-order, non-decreasing order of p_i/β_i. In expression (1), the notation "$j > i$, inB" makes reference to the B-order and "$j \leq i$, inA" makes reference to the A-order. The contribution to the objective function of the jobs in B and A is given by the first and second terms of expression (1). Constraint (2) ensures that all the jobs being processed before d fit into the interval $(0,d)$.

2.2. Model 2: Problems with a job starting at time zero

$$\min \quad \sum_i \alpha_i b_i (d - \sum_{j\leq i,inB} b_j p_j) + \sum_i \beta_i a_i (T - d - \sum_{j>i,inA} a_j p_j) + \sum_i (1 - b_i - a_i)\beta_i (T - d - \sum_j a_j p_j) \tag{5}$$

$$s.t. \quad \sum_{i=1}^{n} b_i p_i \leq d \tag{6}$$

$$\sum_{i=1}^{n} a_i p_i \leq T - d \tag{7}$$

$$a_i + b_i \leq 1 \qquad \forall i = 1,\ldots,n \tag{8}$$

$$\sum_i (a_i + b_i) \geq n - 1 \tag{9}$$

$$a_i, b_i \in \{0,1\} \qquad \forall i = 1,\ldots,n \tag{10}$$

We use the same variables a_i and b_i from the previous model, but in this case a straddling job can appear, starting before d and finishing after d. Therefore, we can have $a_i = b_i = 0$ for at most one job and constraints (8) are no longer equalities as they were in Model 1. Constraints (9) ensure that, apart from the possible straddling job, all the other jobs must belong to B or A. Constraint (6) guarantees that the processing time of jobs in B cannot exceed d. Similarly, constraint (7) ensures that jobs in A do not exceed $T - d$. As in this model the sequence starts at time 0 and no idle time is allowed (by Property 1), it ends at time $T = \sum_i p_i$. Constraints (8) and (9) hold with equality if there is no straddling job.

The objective function is calculated in a different way. The contribution of the jobs in B (the first term in the expression (5)) is computed from time 0; the contribution of the jobs in A (the second term in the expression (5)) is computed from the end of the sequence at time $T = \sum_i p_i$, and the contribution of the straddling job appears in the third term.

The computational results obtained with these two models on a large set of test instances from the literature show that Model 1 is extremely fast, even for very large problems. On the contrary, Model 2 is much slower and for instances with more than 20 jobs obtaining the optimal solution in a reasonable time cannot be guaranteed.

3. A HYBRID HEURISTIC ALGORITHM

We propose a 4-phase algorithmic scheme. In Phase 1, several heuristic rules produce assignments of jobs to machines. In Phase 2, the one-machine problems are solved by using Models 1 and 2. Phase 3 is a local search and Phase 4 is a Path Relinking procedure.

- **Phase 1: Assignment of jobs to machines**
  We use two strategies:
  1. Strategy 1
     - Order the whole set of jobs according to a priority rule: Non-increasing p_j/β_j; $p_j\beta_j/\alpha_j$; $p_j\beta_j$; p_j.
     - For the next job in the ordered list, choose the machine to which the job is assigned, according to a criterion: Next machine; Machine with the lowest sum of processing times; Machine in which adding a job produces a minimum increase in cost.
  2. Strategy 2
     - Select a subset of *early* jobs (jobs we consider candidates for set B on a machine). That can be done in several ways: solving a one-machine problem with all the jobs and a due date equal to $m * d$, or ordering the sets by some criterion favouring jobs which should be early (such as non-increasing β_j/α_j or β_j^2/α_j) and selecting the jobs in order until the sum of processing times exceeds $m * d$. The remaining jobs are considered *tardy*.
     - The list of *early* (*tardy*) jobs is ordered by non-decreasing p_j/α_l (p_j/β_j) and each job is assigned in order to the machine with the minimum total processing time of the jobs already assigned.

  Many different assignment strategies can be developed by combining the priority criteria listed above. We implemented and compared them in a preliminary computational study over a reduced set of 288 instances. As expected, none of them always produced the best results and we decided to keep the 10 best rules, taking into account not only their individual performance but also their complementarity, that is, their ability to produce good results for instances difficult to solve for other rules. Therefore, the result of Phase 1 is a set of 10 assignments which are carried over to the subsequent phases of the process.
- **Phase 2: Solving the one-machine subproblems**
  According to the computational experience with Models 1 and 2, we use the following strategy:
  - For instances with up to 20 jobs per machine solve the subproblem with both Models 1 and 2, and keep the best solution obtained.

  - For instances with more than 20 jobs per machines use only Model 1.

  Models 1 and 2 are solved using CPLEX 11.0. As the objective function is non-convex, we could have previously used a convexification procedure. However, our results show that the internal convexification strategy of CPLEX is very efficient and therefore we use CPLEX directly.

- **Phase 3: Local Search**

  We use two simple moves in order to improve the solutions obtained in Phases 1 and 2. As the procedures of Phase 2 produce the optimal (or near-optimal) sequence of the jobs assigned to each machine, these moves are designed to change the assignment of jobs to machines.

  - Insertion of jobs: Extract a job from its assigned machine and assign it to the machine on which it produces the minimum cost increase.
  - Interchange of sublists: We consider two sublists of consecutive tardy jobs on different machines. If the starting time of the first sublist is earlier than the starting time of the second sublist and the sum of its tardiness penalties is also lower than the sum of the tardiness penalties on the second sublist, exchanging sublists will decrease the total cost of the solution.

- **Phase 4: Path Relinking**

  - Elite Set: The 10 solutions obtained in Phase 3
  - Combination of solutions

    We take each pair of solutions of the Elite Set and consider one of them in turn as the Initial Solution and the other as the Guiding Solution.

    * Order the machines of the Initial Solution in such a way that the first machine will be the machine with more jobs in common with the first machine of the Guiding Solution and repeat the process for the remaining machines.
    * Take the next machine k on the ordered list of the initial solution S_i and compare it with machine k of the guiding solution S_g. Let T_{ik} be the set of jobs assigned to machine k in S_i and let T_{gk} be the set of jobs in machine k in S_g. Build the sets $J_{In} = T_{gk} \ / T_{ik}$, $J_{Out} = T_{ik} \ / T_{gk}$
    * Take the jobs in J_{In} to insert them into T_{ik} and the jobs in J_{Out} to eliminate them from T_{ik} and insert them into the machine where they are in S_g. For each insertion, consider the three possibilities: insert into B (early), into A (tardy), or make it the straddling job, and choose the alternative of minimum cost.

4. COMPUTATIONAL RESULTS

We have used the test instances generated by Rios-Solis and Sourd [5], kindly provided by the authors, as well as the best known solutions fo each instance, obtained by the heuristic proposed in [6]. There are four sets of instances, differing in the way the processing times and the penalties have been generated. The number of jobs varies between 10 and 200, the machines between 2 and 8, and three types of due dates (more or less restrictive) are used. Each combination of these factors is replicated 10 times, producing 3360 instances. In our study, we are currently using only one instance for each combination of factors, excluding those of 200 jobs, and therefore we deal with a set of 288 instances which can be seen as representative of the whole set.

The overall average percentage deviation of the solutions obtained in Phases 1 and 2 from the best known solution is 0.33 %, indicating that the constructive procedure which combines priority assignment rules with the exact solution of subproblems produces good quality results. However, if we look at the detailed results by number of machines, we can see that as the number of machines increases, the solutions worsen. Therefore, the assignment of jobs to machines has to be improved if better solutions are to be obtained, which is the purpose of Phases 3 and 4. The average deviation of the solutions is now -0.063 %. Detailed results by the number of jobs and machines and by the strength of the due date appear in Table 1.

Jobs	**10**	**20**	**50**	**100**	**125**	**150**
	-0.14	-0.42	0.15	0.04	-0.01	-0.01
Machines	**2**	**4**	**6**	**8**		
	-0.24	-0.20	0.001	0.19		
Due date	**0.2**	**0.4**	**0.6**			
tightness	-0.05	-0.12	-0.01			

Table 1: Average percentage deviations from the best known solution

5. CONCLUSIONS AND FUTURE WORK

The results obtained so far are encouraging. The combination of these four phases allows us to obtain improved solutions for quite a difficult problem. However, several questions still need to be addressed. First, the use of exact models for solving the one-machine subproblems. These models are currently applied to the job assignments provided by simple priority rules and would perhaps be more usefully applied to improved job assignments obtained by first applying a local search to the results of the priority rules. Second, more aggressive moves can be added to the Local Search in order to change the job assignments more substantially. Third, the current version of the Path Relinking is quite simple. More complex procedures, such as Dynamic or Evolutive Path Relinking could be implemented.

6. ACKNOWLEDGEMENTS

We would like to thank Yasmine Rios-Solis and Francis Sourd for providing us with their instances and results.

This study has been partially supported by the Spanish Ministry of Science and Technology, DPI2008-02700, cofinanced by FEDER funds.

7. REFERENCES

[1] Graham, R., E. Lawler, J.K. Lenstra and A.H.G. Rinnooy Kan. Optimization and approximation in deterministic sequencing and scheduling: a survey *Annals of Discrete Mathematics*, 5:287-326, 1979.

[2] Hall, N. Single and multi-processor for minimizing completion time variance. *Naval Research Logistics Quarterly* 33:49-54, 1986.

[3] Sundaraghavan, P., M. Ahmed. Minimizing the sum of absolute lateness in single machine and multimachine scheduling. *Naval Research Logistics Quarterly* 31:325-333, 1984.

[4] Chen, Z., W. Powell. A column generation based decomposition algorithm for a parallel machine just in time scheduling problem. *European Journal of Operational Research*, 116:220-232, 1999.

[5] Rios-Solis Y.A., F. Sourd. Exponential neigborhood search for a parallel machine scheduling problem, *Computers and Operations Research*, 35:1697-1712, 2008.

[6] Kedad-Sidhoum, S., Rios-Solis Y.A., F. Sourd. Lower bounds for the earliness-tardiness problem on parallel machines with distinct due dates, *European Journal of Operational Research*, 189:1305-1316, 2008.

[7] Alvarez-Valdes R., J.M. Tamarit and F. Villa. Optimal and approximate solutions for the problem of minimizing weighted earliness-tardiness on a single machine with a common due date. *TOP, in press*, DOI 10.1007/s11750-010-0163-7, 2010.

A hybrid algorithm combining heuristics with Monte Carlo simulation to solve the Stochastic Flow Shop Problem

Esteban Peruyero * Angel A. Juan * Daniel Riera *

* Open University of Catalonia
Barcelona, 08018, SPAIN
ajuanp@gmail.com

ABSTRACT

In this paper a hybrid simulation-based algorithm is proposed for the Stochastic Flow Shop Problem. The main idea of the methodology is to transform the stochastic problem into a deterministic problem and then apply simulation. To achieve this goal we use Monte Carlo simulation and a modified version of the well-known NEH heuristic. This approach aims to provide flexibility and simplicity due to the fact that it is not constrained by any previous assumption and relies in well-tested heuristics.

Keywords: Scheduling, Monte-Carlo simulation, Heuristics, Randomized algorithm

1. INTRODUCTION

The Flow Shop Problem (FSP) is a well-known scheduling problem in which a set of independent jobs have to be sequentially executed (processed) by a set of machines. In this scenario, the processing time of each job in each machine is a known constant value. The classical FSP goal is to determine a sequence of jobs minimizing the total makespan, which is the time difference between the start and finish of processing all the jobs in all the machines (Figure 1).

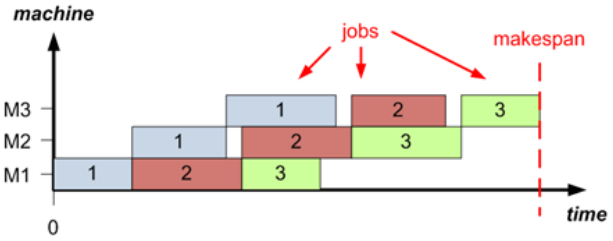


Figure 1: A graphical representation of the FSP

The Stochastic Flow Shop Problem (SFSP) can be seen as a generalization of the FSP. In this non-deterministic version of the Flow Shop Problem, the processing time of each job in each machine is not a constant value, but instead it is a random variable which follows a given probability distribution. Therefore, in this scenario the goal uses to be minimizing the expected makespan, which is not the same as the expected total processing time. The study of the SFSP is within the current popularity of introducing randomness into combinatorial optimization problems. It allows to describe new problems in more realistic scenarios where uncertainty is present.

It is important to remark the FSP as a relevant topic for current research. As it happened with other combinatorial optimization problems, a large number of different approaches and methodologies have been developed to deal with the FSP. These approaches range from pure optimization methods (such as linear and integer programming), which allow to solve small-sized problems, to approximate methods such as heuristics and metaheuristics, which can find near-optimal solutions for medium- and large-sized problems. Although the usual goal is to minimize the makespan, other goals could also be considered, e.g. to minimize the total processing time. Moreover, some of these methodologies are able to provide a set of near-optimal solutions from which the decision-maker can choose according to his/her specific utility function. The situation is quite different in the case of the SFSP: to the best of our knowledge, there is a lack of efficient and flexible methodologies able to provide near-optimal solutions to the stochastic version of the FSP. Moreover, most of the existing approaches are quite theoretical and make use of restrictive assumptions on the probability distributions that model job processing times.

2. BASIC NOTATION AND ASSUMPTIONS

The Stochastic Flow Shop Problem (SFSP) is a scheduling problem that can be formally described as follows: a set J of n independent jobs have to be processed by a set M of m independent machines. Each job $i \in J$ requires a stochastic processing time, p_{ij}, in every machine $j \in M$. This stochastic processing time is a random variable following a certain distribution, e.g. log-normal, exponential, weibull, etc. The goal is to find a sequence for processing the jobs so that a given criterion is optimized. The most commonly used criterion is the minimization of the expected completion time or expected makespan, denoted by $E[C_{max}]$. In addition, it is also assumed that:

- All jobs are processed by all machines in the same order.
- There is unlimited storage between the machines, and non-preemption.
- Machines are always available for processing jobs, but each machine can process only one job at a time.
- A job cannot be processed more than once for each machine.
- Job processing times are independent random variables.

At this point, it is interesting to notice that our approach does not require to assume any particular distribution for the random variables that model processing times. In a practical situation, the specific distributions to be employed will have to be fitted from historical data (observations) using a statistical software. In most existing approaches, however, it is frequently assumed that these processing times will follow a normal or exponential distribution. This assumption is, in our opinion, quite unrealistic and restrictive. For instance, it is unlikely that positive processing times can be conveniently modeled throughout a normal distribution, since any normal distribution includes negative values .

3. STATE OF THE ART AND RELATED WORK

The FSP is a NP-complete problem [1]. Many heuristics and metaheuristics have been proposed in order to solve the FSP due to the impossibility of finding, in reasonable times, exact solutions for most medium- and large-sized instances. Some of the first publications on FSP are those of Johnson [2] and Makino[3]. These authors presented approaches for solving small problems, e.g. problems with only two machines and two jobs. Campbell et al. [4] built a heuristic for the FSP with more than two machines. The NEH algorithm is considered by most researchers as one of the best performing heuristics for solving the FSP. It was introduced by Nawaz et al. [5]. Later, Tailard [6] reduced the NEH complexity by introducing a data structure to avoid the calculation of the makespan. Ruiz and Stützle [7] proposed the Iterated Greedy (IG) algorithm for the FSP built on a two-step methodology. In our opinion, this is one of the best algorithms developed so far to solve the FSP, since it combines simplicity with an outstanding performance.

Many works have focused on the importance of considering uncertainty in real-world problems, particularly in those related to scheduling issues. Thus, Al-Fawzan[8]analyzes the Resource Constrained Project Scheduling Problem (RCPSP) by focusing on makespan reduction and robustness. Jensen[9] also introduces the concepts of neighborhood-based robustness and tardiness minimization. Ke [10] proposes a mathematical model for achieving a formal specification of the Project Scheduling Problem. Allaoui [11] studied makespan minimization and robustness related to the SFSP, suggesting how to measure the robustness. Proactive and reactive scheduling are also characterized in his work. On the one hand, an example of reactive scheduling can be found on Honkomp et al. [12], where performance is evaluated using several methodologies. On the other hand, robustness in proactive scheduling is analyzed in Ghezail et al. [13], who propose a graphical representation of the solution in order to evaluate obtained schedules. As the concept of minimum makespan from FSP is not representative for the stochastic problem, Dodin [14] proposes an optimality index to study the efficiency of the SFSP solutions. The boundaries of the expected makespan are also analyzed mathematically. A theoretical analysis of performance evaluation based on markovian models is performed in Gourgand et al. [15], where a method to compute expected time for a sequence using performance evaluation is proposed. A study of the impact of introducing different types of buffering among jobs is also provided in this work. On the other hand, Integer and linear programming have been employed together with probability distributions to represent the problem in Janak et al. [16].

Simulation has been applied in Juan et al. [17] to solve the FSP. In this work, the NEH algorithm is randomized using a biased probability distribution. Thus, their approach is somewhat similar to a GRASP-like methodology. Simulation-based approaches for the SFSP have mainly focused on performance evaluation, as in Gougard et al. [18]. Similarly, Dodin [14] performs simulations as a way to validate his empirical analysis on the makespan boundaries. Finally, Honkomp et al. [12] also make use of simulation techniques in their approach for reactive scheduling.

In a recent work, Juan et al. [19] describe the application of simulation techniques to solve the Vehicle Routing Problem with Stochastic Demands (VRPSD). The VRPSD is a variation of the classical Vehicle Routing Problem where customer demands are not known in advance. These demands are random variables following some probability distributions. The authors propose to transform the original stochastic problem into a set of related deterministic problems, which are then solved using an efficient algorithm introduced in a previous work [20]. As it will be discussed in more detail next, this paper proposes a similar approach for solving the SFSP.

4. PROPOSED METHODOLOGY

The main idea behind our simulation-based approach is to transform the initial SFSP instance into a FSP instance and then to obtain a set of near-optimal solutions for the deterministic problem by using an efficient FSP algorithm. Notice that, by construction, these FSP solutions are also feasible solutions of the original SFSP instance. Then, simulation is used to determine which solution, among the best-found deterministic ones, shows a lower expected makespan when considering stochastic times. This strategy assumes that a strong correlationship exists between near-optimal solutions for the FSP and near-optimal solutions for the SFSP. Put in other words, good solutions for the FSP are likely to represent good solutions for the SFSP. Notice, however, that not necessarily the best-found FSP solution will become the best-found SFSP solution, since its resulting makespan might be quite sensitive to variations in the processing times. The transformation step is achieved by simply considering the expected value of each processing time as a constant value. Since any FSP solution will be also a feasible SFSP solution, it is possible to use Monte Carlo simulation to obtain estimates for the expected makespan. That is, we obtain these estimates by iteratively reproducing the stochastic behaviour of the processing times in the sequence of jobs given by the FSP solution. Of course, this simulation process will take as many iterations as necessary to obtain accurate estimates. If necessary, variance reduction techniques could be employed in order to reduce the number of iterations to run. Figure 2 shows the flow chart diagram of our approach, which is described next in detail:

1. Consider a SFSP instance defined by a set J of jobs and a set M of machines with random processing times, p_{ij}, for each job $i \in J$ in each machine $j \in M$.
2. For each random processing time p_{ij}, consider its expected or mean value $p^*_{ij} = E\left[p_{ij}\right]$.
3. Let FSP* be the non-stochastic problem associated with the processing times p^*_{ij}, $\forall i \in J, j \in M$.
4. Using any efficient algorithm (e.g. [7, 17]), obtain a set S of n near-optimal solutions for the FSP*.
5. For each $s_k \in S, k = 1, 2, \ldots n$, consider the sequence of jobs in s_k and then start a Monte Carlo simulation in order to estimate the expected makespan associated with this sequence of jobs. Notice that for each s_k, random observations from each p_{ij} ($i \in J, j \in M$) are iteratively generated while maintaining the sequence of jobs provided by s_k.
6. Return the sequence of jobs (solution) which provides the lowest expected makespan.

5. CONTRIBUTION OF OUR APPROACH

The idea of solving a stochastic combinatorial optimization problem through solving one related deterministic problem and then applying simulation is not new (see [19]). However, to the best of our knowledge, this is the first time this approach has been used to solve the SFSP. In fact, most of the SFSP research to date has focused on theoretical aspects of stochastic scheduling. By contrast, the proposed method provides a relatively simple and flexible approach to the SFSP, which in our opinion offers some valuable benefits. In particular, our approach suggests a more practical perspective which is able to deal with more realistic scenarios: by integrating Monte Carlo simulation in our methodology, it is possible to naturally consider any probabilistic distribution for modeling the random job processing times.

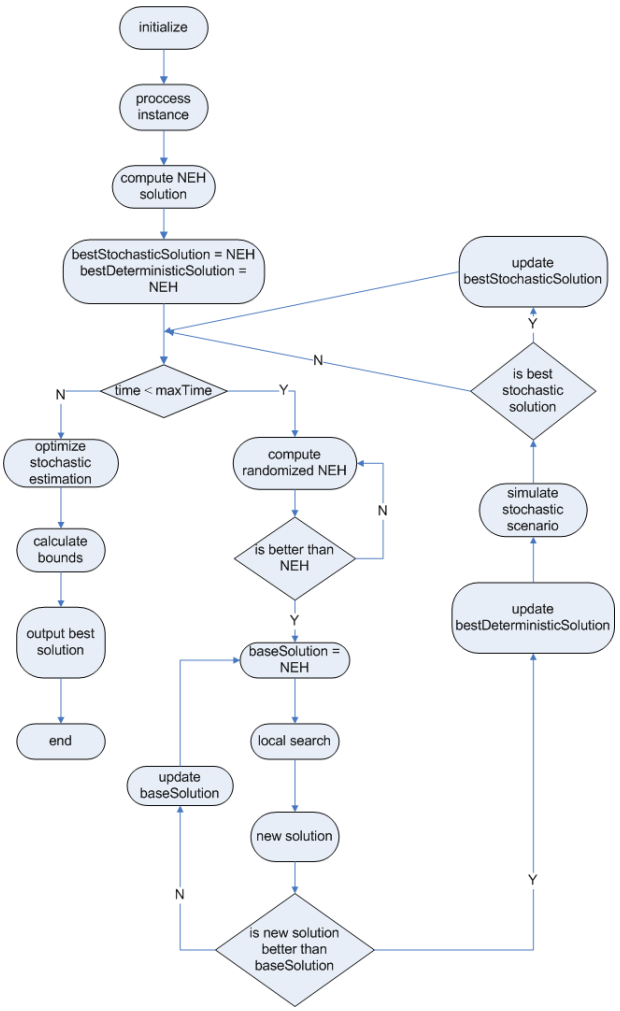


Figure 2: Flow chart of the proposed algorithm

Thus, as far as we know, the presented methodology offers some unique advantages over other existing SFSP approaches. To be specific: (a) the methodology is valid for any statistical distribution with a known mean, both theoretical -e.g. Normal, Log-normal, Weibull, Gamma, etc.- or experimental; and (b) the methodology reduces the complexity of solving the SFSP -where no efficient methods are known yet- to solving the FSP, where mature and extensively tested algorithm have been developed already. All in all, the credibility and utility of the provided solution is increased. Notice also that, being based on simulation, the methodology can be easily extended to consider a different distribution for each job-machine processing time, possible dependencies among these times, etc. Moreover, the methodology can be applied to SFSP instances of any size as far as there exists efficient FSP metaheuristics able to solve those instances. In summary, the benefits provided by our methodology can be summarized in two propierties: simplicity and flexibility.

6. CONCLUSIONS

In this paper we have presented a hybrid approach for solving the Stochastic Flow Shop Problem. The methodology combines Monte Carlo simulation with well tested algorithms for the Flow Shop Problem. The basic idea of our approach is to transform the initial stochastic problem into a related deterministic problem, then obtain a set of alternative solutions for this latter problem using any efficient algorithm, and finally use simulation to verify which of these solutions offers the lowest expected makespan. This approach does not require any previous assumption and is valid for any probability distribution.

7. ACKNOWLEDGEMENTS

This work has been partially supported by the Spanish Ministry of Science and Innovation (TRA2010-21644-C03). This work has been developed in the context of the CYTED-IN3-HAROSA Network (`http://dpcs.uoc.edu`).

8. REFERENCES

[1] A. H. R. Kan, *Machine scheduling problems: Classification, complexity and computations*. Nijhoff (The Hague), 1996.

[2] S. M. Johnson, "Optimal two- and three-stage production schedules with setup times included. naval research logistics," *Naval Research Logistics Quarterly*, no. 1, pp. 61–68, 1954.

[3] T. Makino, "On a scheduling problem," *Operations Research Society of Japan*, vol. 8, pp. 32–44, 1965.

[4] H. G. Campbell, R. A. Dudek, and M. L. Smith, "A heuristic algorithm for the n job, m machine sequencing problem," *Management Science*, vol. 23, no. 16, pp. B630–B637, 1973.

[5] M. Nawaz, E. Enscore, and I. Ham, "A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem," *Omega*, vol. 11, no. 1, pp. 91–95, 1983.

[6] E. Taillard, "Some efficient heuristic methods for the flow shop sequencing problem," *European Journal of Operational Research*, vol. 47, no. 1, pp. 65–74, 1990.

[7] R. Ruiz and T. Stützle, "A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem," *European Journal of Operational Research*, vol. 177, pp. 2033–2049, 2007.

[8] M. A. Al-Fawzan and M. Haouari, "A bi-objective model for robust resource-constrained project scheduling," *International Journal of Production Economics*, vol. 96, no. 2, pp. 175–187, 2005.

[9] M. T. Jensen, "Improving robustness and flexibility of tardiness and total flow-time job shops using robustness measures," *Applied Soft Computing*, vol. 1, no. 1, pp. 35–52, 2001.

[10] H. Ke and B. Liu, "Project scheduling problem with stochastic activity duration times," *Applied Mathematics and Computation*, vol. 168, no. 1, pp. 342–353, 2005.

[11] H. Allaoui, S. Lamouri, and M. Lebbar, "A robustness framework for a stochastic hybrid flow shop to minimize the makespan," in *International Conference on Service Systems and Service Management*, 2006, pp. 1097–1102.

[12] S. Honkomp, L. Mockus, and G. Reklaitis, "Robust scheduling with processing time uncertainty," *Computers & Chemical Engineering*, vol. 21, no. Supplement 1, pp. S1055–S1060, 1997.

[13] F. Ghezail, H. Pierreval, and S. Hajri-Gabouj, "Analysis of robustness in proactive scheduling: A graphical approach," *Computers & Industrial Engineering*, vol. 58, no. 2, pp. 193–198, 2010.

[14] B. Dodin, "Determining the optimal sequences and the distributional properties of their completion times in stochastic flow shops," *Computers & Operations Research*, vol. 23, no. 9, pp. 829–843, 1996.

[15] M. Gourgand, N. Grangeon, and S. Norre, "Markovian analysis for performance evaluation and scheduling in m machine stochastic flow-shop with buffers of any capacity," *European Journal of Operational Research*, vol. 161, no. 1, pp. 126–147, 2005.

[16] S. L. Janak, X. Lin, and C. A. Floudas, "A new robust optimization approach for scheduling under uncertainty: Uncertainty with known probability distribution," *Computers & Chemical Engineering*, vol. 31, no. 3, pp. 171–195, 2007.

[17] A. Juan, R. Ruiz, H. Lourenço, M. Mateo, and D. Ionescu, "A simulation-based approach for solving the flowshop problem," in *Proceedings of the 2010 Winter Simulation Conference. Baltimore, Maryland, USA.*, 2010, pp. 3384–3395.

[18] M. Gourgand, N. Grangeon, and S. Norre, "A contribution to the stochastic flow shop scheduling problem," *European Journal of Operational Research*, vol. 151, no. 2, pp. 415–433, 2003.

[19] A. Juan, J. Faulin, S. Grasman, D. Riera, J. Marull, and C. Mendez, "Using safety stocks and simulation to solve the vehicle routing problem with stochastic demands," *Transportation Research Part C*, 2010, doi:10.1016/j.trc.2010.09.007.

[20] A. Juan, J. Faulin, J. Jorba, D. Riera, D. Masip, and B. Barrios, "On the use of monte carlo simulation, cache and splitting techniques to improve the clarke and wright savings heuristics," *Journal of the Operational Research Society*, 2010, doi:10.1057/jors.2010.29.

A Simulation-based algorithm for solving the Vehicle Routing Problem with Stochastic Demands

Angel Juan * Javier Faulin † Daniel Riera * Jose Caceres * Scott Grasman ‡

* Open University of Catalonia - IN3
Barcelona, Spain
{ajuanp, drierat, jcaceresc}@uoc.edu

† Public University of Navarre
Pamplona, Spain
javier.faulin@unavarra.es

‡ Missouri University of Science & Technology
Rolla, MO, USA
grasmans@mst.edu

ABSTRACT

This paper proposes a flexible solution methodology for solving the Vehicle Routing Problem with Stochastic Demands (VRPSD). The logic behind this methodology is to transform the issue of solving a given VRPSD instance into an issue of solving a small set of Capacitated Vehicle Routing Problem (CVRP) instances. Thus, our approach takes advantage of the fact that extremely efficient metaheuristics for the CVRP already exists. The CVRP instances are obtained from the original VRPSD instance by assigning different values to the level of safety stocks that routed vehicles must employ to deal with unexpected demands. The methodology also makes use of Monte Carlo Simulation (MCS) to obtain estimates of the expected costs associated with corrective routing actions (recourse actions) after a vehicle runs out of load before completing its route.

Keywords: Metaheuristics, Routing, Scheduling

1. INTRODUCTION

The Vehicle Routing Problem with Stochastic Demands (VRPSD) is a well-known NP-hard problem in which a set of customers with random demands must be served by a fleet of homogeneous vehicles departing from a depot, which initially holds all available resources. There are some tangible costs associated with the distribution of these resources from the depot to the customers. In particular, it is usual for the model to explicitly consider costs due to moving a vehicle from one node -customer or depot- to another. These costs are often related to the total distance traveled, but they can also include other factors such as number of vehicles employed, service times for each customer, etc. The classical goal here consists of determining the optimal solution (set of routes) that minimizes those tangible costs subject to the following constraints: (i) all routes begin and end at the depot; (ii) each vehicle has a maximum load capacity, which is considered to be the same for all vehicles; (iii) all (stochastic) customer demands must be satisfied; (iv) each customer is supplied by a single vehicle; and (v) a vehicle cannot stop twice at the same customer without incurring in a penalty cost.

Notice that the main difference between the Capacitated Vehicle Routing Problem (CVRP) and the VRPSD is that in the former all customer demands are known in advance, while in the latter the actual demand of each customer has a stochastic nature, i.e., its statistical distribution is known beforehand, but its exact value is revealed only when the vehicle reaches the customer. For the CVRP, a large set of efficient optimization methods, heuristics and metaheuristics have been already developed ([1]). However, this is not yet the case for the VRPSD, which is a more complex problem due to the uncertainty introduced by the random behavior of customer demands. Therefore, as suggested by Novoa and Storer [2], there is a real necessity for developing more efficient and flexible approaches for the VRPSD. On one hand, these approaches should be efficient in the sense that they should provide optimal or near-optimal solutions to small and medium VRPSD instances in reasonable times. On the other hand, they should be flexible in the sense that no further assumptions need to be made concerning the random variables used to model customer demands, e.g., these variables should not be assumed to be discrete neither to follow any particular distribution. To the best of our knowledge, most of the existing approaches to the VRPSD do not satisfy this flexibility requirement.

The random behavior of customer demands could cause an expected feasible solution to become infeasible if the final demand of any route exceeds the actual vehicle capacity. This situation is referred to as "route failure", and when it occurs, some corrective actions must be introduced to obtain a new feasible solution. For example, after a route failure, the associated vehicle might be forced to return to the depot in order to reload and resume the distribution at the last visited customer. Our methodology proposes the construction of solutions with a low probability of suffering route failures. This is basically attained by constructing routes in which the associated expected demand will be somewhat lower than the vehicle capacity. Particularly, the idea is to keep a certain amount of surplus vehicle capacity (safety stock or buffer) while designing the routes so that if the final routes' demands exceed their expected values up to a certain limit, they can be satisfied without incurring a route failure.

2. BASIC NOTATION

The Stochastic Vehicle Routing Problem (SVRP) is a family of well-known vehicle routing problems characterized by the randomness of at least one of their parameters or structural variables [3]. This uncertainty is usually modeled by means of suitable random variables which, in most cases, are assumed to be independent. The Vehicle Routing Problem with Stochastic Demands

(VRPSD) is among the most popular routing problems within the SVRP family. There are two other classical problems belonging to that family: the Vehicle Routing Problem with Stochastic Customers (VRPSC) which was solved by Gendreau et al. [4] using an adapted Tabu Search, and the Vehicle Routing Problem with Stochastic Times (VRPST), but their applications are rather limited in comparison with the VRPSD, which is described in detail next.

Consider a complete network constituted by $n+1$ nodes, $V = \{0,1,2,\ldots,n\}$, where node 0 symbolizes the central depot and $V^* = V\backslash\{0\}$ is the set of nodes or vertices representing the n customers. The costs associated with traveling from node i to node j are denoted by $c(i,j)\ \forall i,j \in V$, where the following assumptions hold true: (i) $c(i,j) = c(j,i)$ (i.e., costs are usually assumed to be symmetric, although this assumption could be relaxed if necessary); (ii) $c(i,i) = 0$, and (iii) $c(i,j) \leq c(i,k) + c(k,j)\ \forall k \in V$ (i.e., the triangle inequality is satisfied). These costs are usually expressed in terms of traveled distances, traveling plus service time or a combination of both. Let the maximum capacity of each vehicle be $VMC >> \max_{i\in V^*}\{D_i\}$, where $D_i \geq 0\ \forall i \in V^*$ are the independent random variables that describe customer demands -it is assumed that the depot has zero demand. This capacity constraint implies that the demand random value never will be greater than the VMC value, which allows us an adequate performance of our procedure. For each customer, the exact value of its demand is not known beforehand but it is only revealed once the vehicle visits. No further assumptions are made on these random variables other than that they follow a well-known theoretical or empirical probability distribution -either discrete or continuous- with existing mean denoted by $E[D_i]$. In this context, the classical goal is to find a feasible solution (set of routes) that minimizes the expected delivery costs while satisfying all customer demands and vehicle capacity constraints. Even when these are the most typical restrictions, other constraints and factors are sometimes considered, e.g., maximum number of vehicles, maximum allowable costs for a route, costs associated with each delivery, time windows for visiting each customer, solution attractiveness or balance, environmental costs, and other externalities.

3. OUR SIMULATION-BASED APPROACH

Our approach is inspired by the following facts: (a) the VRPSD can be seen as a generalization of the CVRP or, to be more specific, the CVRP is just a VRPSD with constant demands –random demands with zero variance–; and (b) while the VRPSD is yet an emerging research area, extremely efficient metaheuristics do already exists for solving the CVRP. Thus, one key idea behind our approach is to transform the issue of solving a given VRPSD instance into a new issue which consists of solving several "conservative" CVRP instances, each characterized by a specific risk (probability) of suffering route failures. The term conservative refers here to the fact that only a certain percentage of the vehicle total capacity will be considered as available during the routing design phase. In other words, part of the total vehicle capacity will be reserved for attending possible "emergencies" caused by under-estimated random demands during the actual distribution (routing execution) phase. This part can be considered as a safety stock since it reflects the level of extra stock that is maintained to buffer against possible route failures. Next, the specific steps of our methodology are described in detail:

1. Consider a VRPSD instance defined by a set of customers with stochastic demands, where each demand is a random variable following a given statistical distribution –either theoretical or empirical as long as its mean exists.

2. Set a value k for the percentage of the maximum vehicle capacity that will be used as safety stock during the routing design stage.

3. Consider the CVRP(k) defined by: (a) the reduced total vehicle capacity, and (b) the deterministic demands given by the expected value of the real stochastic demands.

4. Solve the CVRP(k) by using any efficient CVRP methodology. Notice that the solution of this CVRP(k) is also an aprioristic solution for the original VRPSD. Moreover, it will be a feasible VRPSD solution as long as there will be no route failure, i.e., as long as the extra demand that might be originated during execution time in each route does not exceed the vehicle reserve capacity or safety stock. Notice also that the cost given by this solution can be considered as a base or fixed cost of the VRPSD solution, i.e., the cost of the VRPSD in case that no route failures occur. Chances are that some route failures occur during the execution phase. If so, corrective actions -such as returning to the depot for a reload before resuming distribution- and their corresponding variable costs will need to be considered. Therefore, the total costs of the corresponding VRPSD solution will be the sum of the CVRP(k) fixed costs and the variable costs due to the corrective actions.

5. Using the solution obtained in the previous step, estimate the expected (average) costs due to possible failures in each route. This can be done by using Monte Carlo simulation, i.e., random demands are generated and whenever a route failure occurs (or just before it happens), a corrective policy is applied and its associated costs are registered. In the experimental section of this paper, every time a route fails we consider the costs of a round-trip from the current customer to the depot; but, since we are using simulation, other alternative policies and costs could also be considered in a natural way. After iterating this process for some hundred/thousand times, a random sample of observations regarding these variable costs are obtained and an estimate for its expected value can be calculated.

6. Depending on the total costs associated with the solutions already obtained, repeat the process from Step 1 with a new value of k -i.e., explore different scenarios to check how different levels of safety stock affect the expected total cost of the VRPSD solution.

7. Finally, provide a sorted list with the best VRPSD solutions found so far as well as their corresponding properties: fixed costs, expected variable costs, and expected total costs.

4. EXPERIMENTAL RESULTS AND DISCUSSION

In the CVRP literature, there exists a classical set of very well-known benchmarks commonly used to test their algorithm. However, as noticed by Bianchi et al. [5], there are no commonly used benchmarks in the VRPSD literature and, therefore, each paper presents a different set of randomly generated benchmarks. Thus, we decided to employ a natural generalization of several classical CVRP instances by using stochastic demands instead of constant ones. So, for each instance, while we decided to keep all node coordinates and vehicle capacities, we changed d_i, the deterministic demands of client i ($\forall i \in \{1,2,\ldots,\#nodes-1\}$) to stochastic demands D_i following an exponential distribution with $E[D_i] = d_i$.

For each instance, a total of 16 scenarios were simultaneously executed using a cluster of 16 personal computers Intel®Core™2 Quad Q8200 at 2.33GHz and 2GB RAM. The 16 scenarios were obtained by varying the available vehicle capacity (i.e., the complementary of the safety-stocks level) from 100% to 85% during the routing-design stage. Table 1 shows the complete results obtained for all 55 classical instances we generalized and tested.

The first column in Table 1 contains the name of each instance, which includes the number of nodes and also the number of routes of the 'standard' solution, e.g. B-n78-k10 is an instance of class B with 78 nodes and able to be solved with a 10-route solution.

Columns 2 to 4 are related to solutions obtained by our algorithm when a 100 % of the vehicle maximum capacity is considered during the design stage. Notice that this strategy always provides pseudo-optimal solutions in terms of fixed costs (Column 2), since they can be directly compared with the CVRP best-known solution. However, since no safety stock is used, there is a chance that these solutions can suffer from route failures. In turn, route failures might imply high expected variable costs (estimated in Column 3 by Monte Carlo simulation), thus increasing the total expected costs, which is estimated in Column 4. Here is where using safety stocks can be of value: by not necessarily using all vehicle maximum capacity during the design stage, some route failures can be avoided. Hopefully, this might lead to new solutions with slightly higher fixed costs but also with lower expected variable costs. At the end, these alternative solutions might present lower total expected costs, which are the ones to be minimized. On the one hand, columns 5 to 9 show the results obtained with our algorithm. Notice that fixed costs in Column 7 are always higher or equal to those in Column 2. However, total expected costs in Column 9 are always lower or equal to those in Column 4. Notice also that sometimes the best-found strategy (for this set of benchmarks) is to use a 100 % of the vehicle maximum capacity (i.e. no safety stocks at all) when designing the routes (Column 5).

5. CONCLUDING REMARKS

We have presented a hybrid approach to solving the Vehicle Routing Problem with Stochastic Demands (VRPSD). The approach combines Monte Carlo simulation with well-tested metaheuristics for the Capacitated Vehicle Routing Problem (CVRP). One of the basic ideas of our methodology is to consider a vehicle capacity lower than the actual maximum vehicle capacity when designing VRPSD solutions. This way, this capacity surplus or safety stocks can be used when necessary to cover route failures without having to assume the usually high costs involved in vehicle restock trips. Another important idea is to transform the VRPSD instance to a limited set of CVRP instances -each of them defined by a given safety-stocks level-, to which efficient solving methods can be applied. Our approach provides the decision-maker with a set of alternative solutions, each of them characterized by their total estimated costs, leaving to him/her the responsibility of selecting the specific solution to be implemented according to his/her utility function. Although other previous works have proposed to benefit from the relationship between the VRPSD and the CVRP, they usually require hard assumptions that are not always satisfied in realistic scenarios. On the contrary, our approach relaxes most of these assumptions and, therefore, it allows for considering more realistic customer demand scenarios. Thus, for example, our approach can be used to solve CVRPSD instances with hundreds of nodes in a reasonable time and, even more important, it is valid for virtually any statistical distribution –the one that best fits historical data on customer demands.

6. ACKNOWLEDGEMENTS

This work has been partially supported by the Spanish Ministry of Science and Innovation (TRA2010-21644-C03) and by the Navarrese and Catalan Governments (IIQ13172.RI1-CTP09-R2, 2009 CTP 00007 and Jerónimo de Ayanz network). This work has been developed in the context of the CYTED-IN3-HAROSA Network (http://dpcs.uoc.edu).

7. REFERENCES

[1] G. Laporte, "What you should know about the vehicle routing problem," *Naval Research Logistics*, vol. 54, pp. 811–819, 2007.

[2] C. Novoa and R. Storer, "An approximate dynamic programming approach for the vehicle routing problem with stochastic demands," *European Journal of Operational Research*, no. 196, pp. 509–515, 2009.

[3] C. Bastian and A. R. Kan, "The stochastic vehicle routing problem revisited," *European Journal of Operations Research*, vol. 56, pp. 407–412, 2000.

[4] M. Gendreau, G. Laporte, and R. SÈguin, "A tabu search heuristic for the vehicle routing problem with stochastic demands," *Operations Research*, vol. 44(3), pp. 469–477, 1996.

[5] L. Bianchi, M. Birattari, M. Chiarandini, M. Mastrolilli, L. Paquete, O. Rossi-Doria, and T. Schiavinotto, "Hybrid metaheuristics for the vehicle routing problem with stochastic demands," *Journal of Mathematical Modelling and Algorithms*, vol. 5, pp. 91–110, 2006.

	Using 100% of the Capacity			Using a Percentage P of the Capacity						
Instance	Fixed	Variable	Total (1)	P	Routes	Fixed	Variable	Total (2)	Time (s)	Gap (1) - (2)
A-n32-k5	787.08	179.49	966.57	100%	5	787.08	179.49	966.57	1	0.00%
A-n33-k5	662.11	159.77	821.88	97%	5	676.10	135.80	811.90	1	1.21%
A-n33-k6	742.69	162.45	905.14	100%	6	742.69	162.45	905.14	1	0.00%
A-n37-k5	672.47	134.43	806.89	97%	5	692.53	109.47	802.00	1	0.61%
A-n38-k5	733.95	157.48	891.43	93%	6	761.25	117.97	879.22	1	1.37%
A-n39-k6	835.25	178.10	1,013.35	94%	6	842.92	150.35	993.27	1	1.98%
A-n45-k6	944.88	254.68	1,199.55	94%	7	979.31	197.70	1,177.01	1	1.88%
A-n45-k7	1,154.39	325.68	1,480.07	100%	7	1,154.39	325.68	1,480.07	2	0.00%
A-n55-k9	1,074.96	304.33	1,379.28	100%	9	1,074.96	304.33	1,379.28	1	0.00%
A-n60-k9	1,362.19	395.42	1,757.61	100%	9	1,362.19	395.42	1,757.61	2	0.00%
A-n61-k9	1,040.31	288.01	1,328.32	95%	10	1,073.86	241.57	1,315.43	1	0.97%
A-n63-k9	1,632.19	518.31	2,150.50	100%	9	1,632.19	518.31	2,150.50	4	0.00%
A-n65-k9	1,184.95	341.43	1,526.37	99%	10	1,213.73	304.73	1,518.46	1	0.52%
A-n80-k10	1,773.79	548.84	2,322.63	100%	10	1,773.79	548.84	2,322.63	7	0.00%
B-n31-k5	676.09	169.46	845.54	95%	5	680.98	158.07	839.05	1	0.77%
B-n35-k5	958.89	267.77	1,226.66	99%	5	978.51	239.61	1,218.12	3	0.70%
B-n39-k5	553.20	142.48	695.68	100%	5	553.20	142.48	695.68	1	0.00%
B-n41-k6	834.92	248.30	1,083.22	96%	7	856.76	224.13	1,080.89	1	0.22%
B-n45-k5	754.23	146.48	900.71	100%	5	754.23	146.48	900.71	1	0.00%
B-n50-k7	744.23	202.85	947.07	93%	7	754.26	186.11	940.37	1	0.71%
B-n52-k7	754.38	204.83	959.21	92%	7	771.02	164.87	935.88	1	2.43%
B-n56-k7	716.42	211.94	928.36	88%	8	757.68	140.32	898.00	1	3.27%
B-n57-k9	1,602.28	559.89	2,162.17	96%	9	1,623.27	515.53	2,138.80	1	1.08%
B-n64-k9	868.40	277.39	1,145.79	100%	9	868.40	277.39	1,145.79	10	0.00%
B-n67-k10	1,039.46	316.59	1,356.05	100%	10	1,039.46	316.59	1,356.05	1	0.00%
B-n68-k9	1,283.16	442.17	1,725.33	97%	9	1,303.09	388.54	1,691.63	8	1.95%
B-n78-k10	1,245.82	367.24	1,613.06	98%	10	1,252.38	357.03	1,609.41	9	0.23%

Table 1: Results for instances A and B using exponentially distributed demands with $E[D_i] = d_i$

Vehicle routing for mixed solid waste collection - comparing alternative hierarchical formulations

Teresa Bianchi-Aguiar * Maria Antónia Carravilla * José F. Oliveira *

* INESC–Porto, Faculdade de Engenharia, Universidade do Porto
Rua Dr. Roberto Frias s/n, 4200-465 Porto, Portugal
{mtbaguiar, mac, jfo}@fe.up.pt

ABSTRACT

The aim of this paper is to present and compare alternative hierarchical formulations for the periodic vehicle routing problem for solid waste collection. The solution of this problem is a one–week plan of daily routes for the transportation of mixed solid waste from containers to disposal facilities, taking into consideration the frequency of collection of each container within the planning horizon, the road network and the resources available. The objective is to minimize operation costs.

The real-world case that supported this study was the collection of mixed solid waste in Ponte de Lima, a municipality in the north of Portugal, and the problem was modelled as a Periodic Vehicle Routing Problem (PVRP) with the additional constraint that routes must pass through one of the alternative disposal facilities before returning to the depot.

Based on this real case scenario, we propose a framework of MIP models with three hierarchical approaches besides the monolithic model. The hierarchical approaches are identified by the aggregation of the decisions in each level: (1) assign and route together; (2) assign days first - assign vehicles and route second; (3) assign first - route second and (4) assign days first - assign vehicles second - route third. Some new estimates for downstream constraints were developed and integrated in upstream levels in order to guarantee feasibility.

Keywords: Waste collection, Hierarchical formulations, Periodic vehicle routing

1. INTRODUCTION

The costs of the collection of solid waste range between 40 and 60% of a community's solid waste management system expenditures [1]. An efficient management of the solid waste collection can therefore generate significant savings while ensuring hygiene patterns and satisfaction of the inhabitants, besides all the other advantages common to the efficient management of transportation systems.

This work is based on a real case concerning Ponte de Lima, a municipality in the north of Portugal. The municipality manages the collection of the mixed waste generated in Ponte de Lima and guarantees its transport to disposal facilities. The main objective of the work done with the municipality was the reduction of the collection costs, that are highly dependent of the distance traveled by the vehicles. The resources such as the number and location of the depots and containers, the number of vehicles and staff, as well as the collection frequency of the containers in each parish were already fixed.

The output of the study should therefore be the visiting calendar of each container within the weekly planning horizon, considering the constrains of the collection frequency, and the plan of the routes for each vehicle and day, with the additional constraint that the routes must go through a disposal facility to unload the waste before returning to the depot. Problems with these characteristics are modeled in the literature as Periodic Vehicle Routing Problems (PVRP), a variant of the Vehicle Routing Problem (VRP).

The PVRP is known to be an NP-hard problem and the additional constraints that had to be included to adapt the model to the real situation of Ponte de Lima made the resolution even more challenging. In order to be able to solve the real problem we built a framework with three hierarchical approaches, which we have tested along with the monolithic model. The hierarchical approaches are identified by the aggregation of the decisions in each level: (1) assign and route together; (2) assign days first - assign vehicles and route second; (3) assign first - route second and (4) assign days first - assign vehicles second - route third. Some estimates of downstream constraints were developed and added in upstream levels in order to guarantee feasibility. We compared the results obtained with the MIP formulations developed for the approaches and with the current practice of the municipality.

The remainder of this paper is organized as follows: in section 2, a brief review of the relevant literature is presented. The problem is described in section 3 and in section 4 the hierarchical framework as well as the developed formulations are presented. In section 5 the results obtained are described and the approaches compared. Conclusions are drawn in section 6.

2. LITERATURE REVIEW

Routing problems have been widely treated in the literature because of their high complexity and practical relevance. The Traveling Salesman Problem (TSP) is the most discussed routing problem and consists in determining a minimum distance route that begins in a given location, passes through all the other locations (customers) and returns to the initial location [2]. In the Vehicle Routing Problem (VRP), a fleet of vehicles with known capacity is available to visit customers which have a known demand. The objective is to design routes for the vehicles at minimal total cost, guaranteeing that all the customers are served and that the capacity of the vehicles is not exceeded [3]. This problem adds to the TSP the decision of which customers assign to which vehicles.

The Periodic Vehicle Routing Problem (PVRP) is an extension of the VRP where customers must be visited with pre-defined frequencies over an extended period. The additional component of the problem consists in the assignment of one visiting calendar from a given set to each customer. The overall objective is to assign routes to the vehicles for each day of the planning horizon that minimize the total travel cost. The visiting calendar of each client must be met and routes are subject to vehicle capacity and route duration constraints. This problem was formally introduced in 1974 by Beltrami and Bodin as a generalization of the VRP, precisely in an application of municipal waste collection [4].

Russel and Igo called the PVRP an "Assignment Routing Problem" and mentioned the difficulties of choosing a calendar for each customer together with solving the routing problem [4]. To deal with the complexity and large scale nature of the problem, several authors consider the PVRP as a multilevel problem:

1. In the first level, a calendar is selected for each customer. In this way, it is decided which customers are visited on each day of the planning horizon;
2. In the second level, and for each day of the planning horizon, customers are assigned to the vehicles available in that day;
3. Finally, in the third level, a route is designed for each combination of day and vehicle.

Note that in the VRP, only the last two decisions need to be made and over a single day only. Being the VRP an NP-hard problem, the PVRP is therefore at least as difficult [5].

A significant body of work has been evolving, with multiple variants, formulations and solution methods applied to the PVRP. Three important variants of the PVRP are mostly addressed in the literature: the PVRP with time window constraints – PVRPTW [6], with service choice – PVRP-SC [7], with multiple depots – MDPVRP [8] and with intermediate facilities – PVRP-IF [9]. In this last variant, capacity replenishment is possible at different points along the route. As far as formulations are concerned, the most used one is the 4-index formulation from Christofides and Beasley, based on the VRP 3-index formulation from Golden et al [4]. Other formulations have been emerging, considering only the assignment problems [10, 11, 12]. More recently, alternative modeling approaches have been emerging, such as the Set Partitioning (SP) [13]. For instances of realistic size, the problem has been solved mostly with heuristics and metaheuristics and in sequential phases. Two-phase solution methods are more commonly found (a survey on solution methods can be found in [4]).

In [14], Ball states that solving an hierarchical problem is more than solving a set of distinct problems. It is necessary to guarantee feasibility in the downstream levels by including approximate measurements of lower level constraints in upstream levels. In the PVRP, this means that in the assignment problems it is necessary to guarantee that the number of customers assigned to a vehicle in a day neither exceeds its capacity nor leads to subproblems where it is not possible to create any route without exceeding its maximum duration. Whereas vehicle capacity constraints have already appeared in assignment problems, approximate measurements of route duration have not been covered so far.

To conclude, and concerning waste collection, this practical application has already been studied in the literature, not only concerning mixed but also separate waste [15, 16, 5, 17, 18].

3. PROBLEM DEFINITION

The municipality of Ponte de Lima owns and operates a fleet of 5 vehicles with different capacities for the mixed-waste collection. These vehicles are parked in a garage in a central parish – Arca. The 994 mixed-waste containers are non-uniformly distributed over Ponte de Lima and the waste is periodically collected and transported to disposal facilities, where afterwards it is whether dumped in a controlled environment or transformed. The filling rates of the containers are highly dependent on the density of both the containers and the inhabitants of the region. They also depend on the collection frequency imposed. The collection is performed 6 days a week. Figure 1 shows the location of the two existing disposal facilities and the depot as well as the collection frequency of the containers within each parish.

Currently the plans are monthly hand-made, without assuring that the collection frequency matches the frequencies defined for each parish.

3.1. Objective

Different filling rates led the municipality to establish different frequencies of collection for the containers. Therefore, for a given planning horizon, a set of routes is required for each vehicle as well as a visiting schedule for each container. Each route should consist of an ordered list of visiting sites that ends on a disposal facility to deposit the waste after collection. The lowest frequency for a container is one visit in a week, which suggests a collection plan of one week.

The objective is to minimize collection costs, which are essentially dependent on the distance traveled by the vehicles. Routes are constrained by vehicle capacity and work shift duration. Each container should be visited as many times per week as its frequency and the visiting days should be distributed as uniformly as possible through the period.

4. A FRAMEWORK OF ALTERNATIVE HIERARCHICAL FORMULATIONS

The problem described in section 3 can be formulated as a Periodic Vehicle Routing Problem. An additional constraint is observed though: routes must pass through a disposal facility to unload the waste before returning to the depot.

The decomposition of highly complex optimization problems into subproblems, hierarchically solved, is a well-known strategy in the literature (e.g. [11, 14]). Not only the problem becomes more efficiently solvable, but it is also taken into account that, in the context of real-world applications, these complex problems arise under broader decision making contexts, with decisions made by different actors and with different time horizon scopes. Therefore, it does make sense to break down the problem into subproblems, not loosing sight from the hierarchical relationships among them. On the other hand there is the well-known fact that solving until optimality a sequence of subproblems does not guarantee optimality for the overall problem resolution. However, given the size of real-world applications, the global optimum would be out of reach. An additional advantage of hierarchical approaches is the possibility of considering different optimization criteria at each level [11].

Bearing this in mind, in figure 2 we propose a framework of decomposition processes for the PVRP, based on different aggregations of the three decisions involved in the problem and identified in section 2. In fact, the PVRP is too difficult to be solved directly by exact methods when considering instances of realistic size. All the subproblems identified are smaller and more amenable to rapid solutions.

The approaches are:

1. Deciding at the same time which customers will be served in each day of the week, by which vehicle, and in which sequence (assign and route together);
2. Deciding first which customers will be served in each day of the week, and afterwards by which vehicle and in which sequence (assign days first - assign vehicles and route second);
3. Deciding at the same time which customers will be served in each day of the week and by which vehicle, and afterwards in which sequence (assign first - route second);
4. Deciding first which customers will be served in each day of the week, then by which vehicle, and finally in which

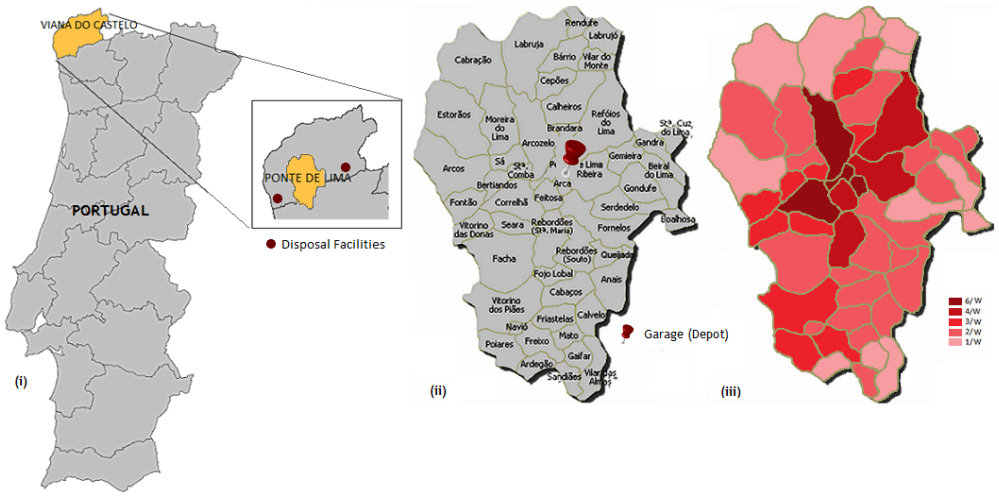


Figure 1: Ponte de Lima Collection System: (i) Disposal Facilities, (ii) Depot, (iii) Collection frequency in each parish

sequence (assign days first - assign vehicles second - route third).

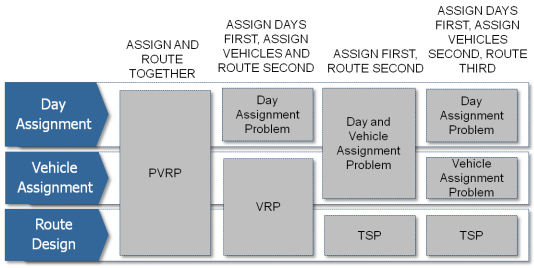


Figure 2: Alternative Decomposition Approaches to the PVRP

The first levels correspond to assignment problems whereas the last level of each approach corresponds to a routing problem. The complexity of the routing problems decrease from the first to the last approach but the number of times that a routing problem is solved increases. For instance, to solve the problem of the case study, in the approach 2 the VRP is solved 6 times, whereas in approaches 3 and 4 the TSP is solved a maximum of 30 times.

Some authors proposed approaches complementary to cluster first - route second, namely route first - cluster second. However, as stated in [14], these approaches do not perform as well from a computational perspective.

To build the framework, different formulations from the literature were put together, and divided by type of approach. All the problems identified in the framework were formulated taking into consideration the practical application features and the formulations scattered before. As far as routing is concerned, the traditional two (TSP) and three (VRP) index formulations were considered because of their greater flexibility in incorporating additional features [3]. To eliminate subtours, a transit load constraint was used instead of the traditional Dantzig-Fulkerson-Johnson subtour elimination constraint [2, 3, 19]. This constraint is a 4–index version of the generalized Miller-Tucker-Zemlin subtour elimination constraints. Concerning the assignment problems, our formulations include some new developments to prevent infeasibility in the downstream levels. An estimation of route duration is proposed in order to prevent that the routes exceed maximum duration. To the best of our knowledge, this is the first time that this constraint is addressed in upper levels. In what concerns vehicle capacity, we have introduced a slack parameter in the corresponding constraint of the upper levels. Finally, the experience with the case study instance allowed some adjustments in the parameters of the models.

5. COMPUTATIONAL RESULTS

The alternative approaches, and corresponding MIP formulations, were evaluated with the case study instance, whose characteristics were described in section 3. The results were compared in terms of objective function value, total execution time and average gap between the integer solution and the lower bound found by CPLEX in each sub-problem (Gap). Additionally, the number of routes and the duration of the longest route were recorded. The total number of variables and constraints of the models generated to solve each level were also analyzed.

All hierarchical approaches presented a reduction of more than 70% on both the number of variables and on the number of constraints, when compared with the monolithic model. It is important to bear in mind that these numbers depend not only on the instance but also on the running conditions because the number of variables and constraints of the lower levels are influenced by the results (concrete decision variable values) of upper levels' problems.

When tested with the case study instance, the monolithic model of approach 1 did not achieve any solution within the time limit. This confirms, also for this case study, the difficulty of the problem which was precisely the reason that has led several authors to consider the PVRP as a multilevel problem and the motivation for this work.

The best results were obtained with approach 2 (assign days first - assign vehicles and route second), not only concerning total distance but also the number of routes. Interestingly, this was the approach with higher gaps in its two levels. In fact, the overall solution quality is mostly influenced by routing decisions as these decisions directly influence total distance and the duration of the routes. By assigning vehicles together with the routing activity we are giving the model freedom to explore a wider solution space based on correct estimates of distances and times.

In spite of achieving optimal solutions on the routing problems and having the lowest gap in the first level, approach 3 (assign days and vehicles first - route second) had the worst global performance. In fact, the problem of assigning days and vehicles still has a considerable dimension, with three times more constraints than the other two equivalent hierarchical approaches.

At last, approach 4 (assign days first - assign vehicles second -

route third) performed in second. This is the only approach with three levels and it was the one generating the smallest number of variables and constraints, which suggests that its problems are simpler and more efficiently solvable.

The fundamental reason for decomposing a problem is that the overall problem is too difficult to be solved monolithically. Thus, it is essential that individual problems are efficiently solvable. On the other hand, when increasing the number of levels we are restricting more and more the solution space. These facts, supported by the results obtained, raise once more the question of the trade–off between the number of decompositions and the difficulty of the resulting problems. Also important is the ability to estimate accurately the distance measure in the upper levels. In fact, this measure evaluates the solutions and should remain as close as possible to the original objective function.

Improved route plans were obtained, not only concerning total distance run by the vehicles but also the number of routes. Besides the reduction in operational costs, an improved service level is expected, since the frequency of collection is guaranteed and the space between consecutive visits to each container is balanced. Moreover, the work shift duration is not exceeded. These were problems faced by the municipality with its current plans.

6. CONCLUSIONS

In this paper, motivated by a real case scenario of a waste collection problem, we proposed a framework of MIP models with a monolithic model and three hierarchical approaches to the Periodic Vehicle Routing Problem. The hierarchical approaches were identified by the aggregation of the decision variables in each level: (1) assign and route together; (2) assign days first - assign vehicles and route second; (3) assign first - route second and (4) assign days first - assign vehicles second - route third. Estimates of downstream constraints were also developed and added at the upper levels in order to guarantee feasibility at the lower levels: maximum duration of routes and maximum load capacity of vehicles.

The hierarchical approach (2), assign days first - assign vehicles and route second, led to better results considering either the total distance traveled or the total number of routes. The hierarchical resolution raised two important points: the trade–off between the number of decompositions and the difficulty of the resulting subproblem and the importance of an accurate estimation of the distance of the routes in the upper levels.

In what concerns our case study, our models were able to obtain better results when compared to the current practice in the municipality. An improved service level is also expected, since the frequency of collection is guaranteed and the space between consecutive visits to each container is balanced, moreover, the work shift duration is not exceeded. These were problems faced by the municipality with its current plans.

As future work, the framework can be extended to take into account multiple depots (MDPVRP). For urban areas with minor distances between collection points, the possibility of returning to collection activity after disposal can also be incorporated (PVRP-IF). Other MIP formulations might be developed for the subproblems, with alternative distance estimates or considering different approaches to the subproblems. Another area of future research is the incorporation of other optimization criteria.

7. REFERENCES

[1] J. Pichtel, *Waste management practices: municipal, hazardous, and industrial*. Taylor & Francis, 2005.

[2] T. Oncan, I. K. Altinel, and G. Laporte, "A comparative analysis of several asymmetric traveling salesman problem formulations," *Computers and Operations Research*, vol. 36, no. 3, pp. 637–654, 2009.

[3] P. Toth and D. Vigo, *The vehicle routing problem*, ser. SIAM monographs on discrete mathematics and applications. Society for Industrial and Applied Mathematics, 2002.

[4] P. M. Francis, K. R. Smilowitz, and M. Tzur, *The period vehicle routing problem and its extensions*, ser. Operations Research/Computer Science Interfaces Series, B. Golden, S. Raghavan, and E. Wasil, Eds. Springer US, 2008, vol. 43.

[5] J. Teixeira, A. Antunes, and J. Desousa, "Recyclable waste collection planning–a case study," *European Journal of Operational Research*, vol. 158, no. 3, pp. 543–554, Nov. 2004.

[6] J. François Cordeau, G. Laporte, and A. Mercier, "A unified tabu search heuristic for vehicle routing problems with time windows," *The Journal of the Operational Research Society*, vol. 52, no. 8, pp. 928–936, 2001.

[7] P. Francis, K. Smilowitz, and M. Tzur, "The period vehicle routing problem with service choice," *Transportation Science*, vol. 40, no. 4, pp. 439–454, 2006.

[8] E. Hadjiconstantinou and R. Baldacci, "A multi-depot period vehicle routing problem arising in the utilities sector," *The Journal of the Operational Research Society*, vol. 49, no. 12, pp. 1239–1248, 1998.

[9] E. Angelelli and M. G. Speranza, "The periodic vehicle routing problem with intermediate facilities," *European Journal of Operational Research*, vol. 137, no. 2, pp. 233–247, 2002.

[10] C. C. R. Tan and J. E. Beasley, "A heuristic algorithm for the period vehicle routing problem," *Omega*, vol. 12, no. 5, pp. 497–504, 1984.

[11] M. Mourgaya and F. Vanderbeck, "Column generation based heuristic for tactical planning in multi-period vehicle routing," *European Journal of Operational Research*, vol. 183, no. 3, pp. 1028–1041, 2007.

[12] B. M. Baker and J. Sheasby, "Extensions to the generalised assignment heuristic for vehicle routing," *European Journal of Operational Research*, vol. 119, no. 1, pp. 147–157, 1999.

[13] R. Baldacci, E. Bartolini, A. Mingozzi, and R. Roberti, "An exact solution framework for a broad class of vehicle routing problems," *Computational Management Science*, vol. 7, pp. 229–268, 2010.

[14] M. O. Ball, "Heuristics based on mathematical programming," *Surveys in Operations Research and Management Science*, vol. 16, no. 1, pp. 21–38, 2011.

[15] D. Tung, "Vehicle routing-scheduling for waste collection in Hanoi," *European Journal of Operational Research*, vol. 125, no. 3, pp. 449–468, Sep. 2000.

[16] E. Angelelli and M. G. Speranza, "The application of a vehicle routing model to a waste-collection problem: two case studies," *The Journal of the Operational Research Society*, vol. 53, no. 9, pp. 944–952, 2002.

[17] A. C. Matos and R. C. Oliveira, "An experimental study of the ant colony system for the period vehicle routing problem," *Ant Colony, Optimization and Swarm Intelligence*, vol. 3172, pp. 1–29, 2004.

[18] T. R. P. Ramos and R. C. Oliveira, "Delimitation of service areas in reverse logistics networks with multiple depots," *Journal of the Operational Research Society*, pp. 1–13, Jun. 2010.

[19] I. Kara, G. Laporte, and T. Bektas, "A note on the lifted Miller-Tucker-Zemlin subtour elimination constraints for the capacitated vehicle routing problem," *European Journal of Operational Research*, vol. 158, no. 3, pp. 793–795, Nov. 2004.

Branch and Cut and Price for the Time Dependent Vehicle Routing Problem with Time Windows

Said Dabia [*] Stefan Røpke [†] Tom Van Woensel [*] Ton De Kok [*]

[*] Eindhoven University of Technology, School of Industrial Engineering
Eindhoven, The Netherlands
{s.dabia, t.v.woensel, a.g.d.kok}@tue.nl

[†] Denmark University of Technology, Department of Transport
Copenhagen, Denmark
sr@transport.dtu.dk

ABSTRACT

In this paper, we consider the Time-Dependent Vehicle Routing Problem with Time Windows (TDVRPTW). Travel times are time-dependent (e.g. due to road congestion), meaning that depending on the departure time from a customer a different travel time is incurred. Because of time-dependency, vehicles' dispatch times from the depot are crucial as road congestion might be avoided. Due to its complexity, all existing solutions to the TDVRPTW are based on (meta-) heuristics and no exact methods are known for this problem. In this paper, we propose the first exact method to solve the TDVRPTW. The MIP formulation is decomposed into a master problem that is solved by means of column generation, and a pricing problem. To insure integrality, the resulting algorithm is embedded in a Branch and Cut framework. We aim to determine the set of routes with the least total travel time. Furthermore, for each vehicle, the best dispatch time from the depot is calculated.

Keywords: Vehicle routing problem, Column generation, Time-dependent travel times, Branch and cut

1. INTRODUCTION

The vehicle routing problem with time windows (VRPTW) concerns the determination of a set of routes starting and ending at a depot, in which the demand of a set of geographically scattered customers is fulfilled. Each route is traversed by a vehicle with a fixed and finite capacity, and each customer must be visited exactly once. The total demand delivered in each route should not exceed the vehicle's capacity. At customers time windows are imposed, meaning that service at a customer is only allowed to start within its time window. The solution to the VRPTW consists of the set of routes with the least traveled distance.

Due to its practical relevance, the VRPTW has been extensively studied in the literature. Consequently, many (meta-) heuristics and exact methods have been successfully developed to solve it. However, most of the existing models are time-independent, meaning that a vehicle is assumed to travel with constant speed throughout its operating period. Because of road congestion, vehicles hardly travel with constant speed. Obviously, solutions derived from time-independent models to the VRPTW could be infeasible when implemented in real-life. In fact, in real-life road congestion results in tremendous delays. Consequently, it is unlikely that a vehicle respects customers' time windows. Therefore, it is highly important to consider time-dependent travel times when dealing with the VRPTW.

In this paper, we consider the time-dependent vehicle routing problem with time windows (TDVRPTW). We take road congestion into account by assuming time-dependent travel times: depending on the departure time at a customer a different travel time is incurred. We divide the planning horizon into time zones (e.g. morning, afternoon, etc.) where a different speed is associated with each of these zones. The resulting stepwise speed function is translated into travel time functions that satisfy the First-In First-Out (FIFO) principle. Because of the time-dependency, the vehicles' dispatch times from the depot are crucial. In fact, a later dispatch time from the depot might result in a reduced travel time as congestion might be avoided. In this paper, we aim to determine the set of routes with the least total travel time. Furthermore, for each vehicle, the best dispatch time from the depot is calculated.

Despite numerous publications dealing with the vehicle routing problem, very few addressed the inherent time-dependent nature of this problem. Additionally, to our knowledge, all existing algorithms are based on (meta-) heuristics, and no exact approach has been provided for the TDVRPTW. In this paper, we solve the TDVRPTW exactly. We use the flow arc formulation of the VRPTW which is decomposed into a master problem (set partitioning problem) and a pricing problem. While the master problem remains unchanged, compared to that of the VRPTW (as time-dependency is implicitly included in the set of feasible solutions) the pricing problem is translated into a time-dependent elementary shortest path problem with resource constraints (TDESPPRC), where time windows and capacity are the constrained resources. The relaxation of the master problem is solved by means of column generation. To guarantee integrality, the resulting column generation algorithm is embedded in a branch-and-bound framework. Furthermore, in each node, we use cutting planes in the pricing problem to obtain better lower bounds and hence reduce the size of branching trees. This results in a branch-and-cut-and-price (BCP) algorithm. Time-dependency in travel times increases the complexity of the pricing problem. In fact, the set of feasible solutions increases as the cost of a generated column (*i.e.* route) does not depend only on the visited customers, but also on the vehicles' dispatch time from the depot. The pricing problem in case of the VRPTW is usually solved by means of a labeling algorithm. However, the labeling algorithm designed for the VRPTW is incapable to deal with time-dependency in travel times and needs to be adapted. In this paper, we develop a time-dependent labeling (TDL) algorithm such that in each label the arrival time function (*i.e.* function of the departure time from the depot) of the corresponding partial path is stored. the TDL generates columns that have negative reduced cost together with their best dispatch time from the depot. To accelerate the BCP algorithm, two heuristics based on the TDL algorithm are designed to quickly find columns with negative reduced cost. Moreover, new dominance criteria are introduced to discard labels that do not

lead to routes in the final optimal solution. Furthermore, we relax the pricing problem by allowing non-elementary paths. The resulting pricing problem is a time-dependent shortest path problem with resource constraints (TDSPPRC). Although the TDSPPRC results in worse lower bounds, it is easier to solve and integrality is still guaranteed by branch-and-bound. Moreover, TDSPPRC should work well for instances with tight time windows. Over the last decades, BCP proved to be the most successful exact method for the VRPTW. Hence, our choice for a BCP framework to solve the TDVRPTW is well motivated.

The main contributions of this paper are summarized as follows. First, we present an exact method for the TDVRPTW. We propose a branch-and-cut-and price algorithm to determine the set of routes with the least total travel time. Contrary to the VRPTW, the pricing problem is translated into a TDESPPRC and solved by a time-dependent labeling algorithm. Second, we capture road congestion by incorporating time-dependent travel times. Because of time dependency, vehicles' dispatch times from the depot are crucial. In this paper, dispatch times from the depot are also optimized. In the literature as well as in practice, dispatch time optimization is approached as a post-processing step, *i.e.* given the routes, the optimal dispatch times are determined. In this paper, the scheduling (dispatch time optimization) and routing are simultaneously performed.

2. LITERATURE REVIEW

An abundant number of publications is devoted to the vehicle routing problem (see [1], [2], and [3] for good reviews). Specifically, the VRPTW has been extensively studied. For good reviews on the VRPTW, the reader is referred to [4], and [5]. The majority of these publications assume a time-independent environment where vehicles travel with a constant speed throughout their operating period. Perceiving that vehicles operate in a stochastic and dynamic environment, more researchers moved their effort towards the optimization of the time-dependent vehicle routing problems. Nevertheless, literature on this subject remains scarce.

In the context of dynamic vehicle routing, we mention the work of [6], [7] and [8] where a probabilistic analysis of the vehicle routing problem with stochastic demand and service time is provided. [9], [10] and [11] tackle the vehicle routing problem where vehicles' travel time depends on the time of the day, and [12] considers a time-dependent traveling salesman problem. Time-dependent travel times has been modeled by dividing the planning horizon into a number of zones, where a different speed is associated with each of these time zones (see [11] and [13]). In [14], traffic congestion is captured using a queuing approach. [9] and [12] models travel time using stepwise function, such that different time zones are assigned different travel times. [15] emphasized that modeling travel times as such leads to the undesired effect of passing. That is, a later start time might lead to an earlier arrival time. As in [11], we consider travel time functions that adhere to the FIFO principle. Such travel time functions does not allow passing.

While several successful (meta-) heuristics and exact algorithms have been developed to solve the VRPTW, algorithms designed to deal with the TDVRPTW are somewhat limited to (meta-) heuristics. In fact, most of the existing algorithms are based on tabu search ([11], [14], [13] and [16]). In [9] mixed integer linear formulations the time-dependent vehicle routing problem are presented and several heuristics based on nearest neighbor and cutting planes are provided. [17] proposes an algorithm based on a multi ant colony system and [18] presents a genetic algorithm. In [19] a local search algorithm for the TDVRPTW is developed and a dynamic programming is embedded in the local search to determine the optimal starting for each route. [20] considers a multi-criteria routing problem, they propose an approach based on the decomposition of the problem into a sequence of elementary itinerary subproblems that are solved by means of dynamic programming. [12] presents a restricted dynamic programming for the time-dependent traveling salesman problem. In each iteration of the dynamic programming, only a subset with a predefined size and consisting of the best solutions is kept and used to compute solutions in the next iteration. [21] emphasizes the difficulty of implementing route improvement procedures in case of time-dependent travel times and proposes efficient ways to deal with that issue. In this paper, we attempt to solve the TDVRPTW to optimality using column generation. To the best of our knowledge, this is the first time an exact method for the TDVRPTW is presented.

Column generation has been successfully implemented for the VRPTW. For an overview of column generation algorithms, the reader is referred to [22]. in the context of the VRPTW, [23] designed an efficient column generation algorithm where they applied subtour elimination constraints and 2-path cuts. This has been improved by [24] by applying k-path cuts. [25] proposes a column generation algorithm by applying subset-row inequalities to the master problem (set partitioning). Although, adding subset-row inequalities to the master problem increases the complexity of the pricing problem, [25] shows that better lower bounds can be obtained from the linear relaxation of the master problem. To accelerate the pricing problem solution, [26] proposes a tabu search heuristic for the ESPPRC. Furthermore, elmentarity is relaxed for a subset of nodes and generalized k-inequalities are introduced. Recently, [27] introduce a new route relaxation, called ng-route, used to solve the pricing problem. Their framework proves to be very effective in solving difficult instances of the VRPTW with wide time windows. [15] argued that existing algorithms for the VRPTW fail to solve the TDVRPTW. One major drawback of the existing algorithms is the incapability to deal with the dynamic nature of travel times. Therefore, existing algorithms for the VRPTW can not be applied to the TDVRPTW without a radical modification of their structure. In this paper, a branch-and-cut-and-price framework is modified such that time-dependent travel times can be incorporated.

3. PROBLEM DESCRIPTION

We consider a graph $G(V,A)$ on which the problem is defined. $V = \{0,1,...,n,n+1\}$ is the set of all nodes such that $V_c = V/\{0,n+1\}$ represents the set of customers that need to be served. Moreover, 0 is the start deport and $n+1$ is the end depot. $A = \{(i,j) : i \neq j \text{ and } i,j \in V\}$ is the set of all arcs between the nodes. Let K be the set of homogeneous vehicles such that each vehicle has a finite capacity Q and q_i demand of customer $i \in V_c$. We assume $q_0 = q_{n+1} = 0$ and $|K|$ is unbounded. Let a_i and b_i be respectively the opening and closing time of node's i time window. At node i, a service time s_i is needed. We denote t_i departure time from node $i \in V$ and $\tau_{ij}(t_i)$ travel time from node i to node j which depend on the departure time at node i.

3.1. Travel Time and Arrival Time Functions

We divide the planning horizon into time zones where a different speed is associated with each of these zones. The resulting stepwise speed function is translated into travel time functions that satisfy the First-In First-Out (FIFO) principle. Usually traffic networks have a morning and an afternoon congestion period. Therefore, we consider speed profiles that have two periods with relatively low speeds. In the rest of the planning horizon, speeds are relatively high. This complies with data collected for a Belgian highway ([28]). Given a partial path P_i starting at the depot 0 and ending at some node i, the arrival time at i depends on the dispatch

time t_0 at the depot. Due to the FIFO property of the travel time functions, a later dispatch at the depot will result in a later arrival at node i. Therefore, if route P_i is unfeasible for some dispatch time t_0 at the depot (*i.e.* time windows are violated), P_i will be unfeasible for any dispatch time at the depot that is later than t_0. Moreover, If we define $\delta_i(t_0)$ as the arrival time function at node i given a dispatch time t_0 at the depot, $\delta_i(t_0)$ will be non-decreasing in t_0. We call the parent node j of node i, the node that is visited directly before node i on route P_i. $\delta_j(t_0)$ is the arrival time at j given a dispatch time t_0 at the depot, and $\tau_{ji}(\delta_j(t_0))$ is the incurred travel time from j to i. Consequently, for every $i \in V$, $\delta_i(t_0)$ is recursively calculated as follows:

$$\delta_0(t_0) = t_0 \quad \text{and} \quad \delta_i(t_0) = \delta_j(t_0) + \tau_{ji}(\delta_j(t_0)) \tag{1}$$

4. COLUMN GENERATION

To derive the set partitioning formulation for the TDVRPTW, we define Ω as the set of feasible paths. A feasible path is defined by the sequence of customers visited along it and the dispatch time at the depot. To each path $p \in \Omega$, we associate the cost c_p which is simply its duration. Hence:

$$c_p = e_p - s_p \tag{2}$$

Where e_p and s_p are respectively the end time and the start time of path p. Furthermore, if y_p is a binary variable that takes the value 1 if and only if the path p is included in the solution, the TDVRPTW is formulated as the following set partitioning problem:

$$\min z_M = \sum_{p \in \Omega} c_p y_p \tag{3}$$

subject to:

$$\sum_{p \in \Omega} a_{ip} y_p = 1 \qquad \forall i \in V \tag{4}$$

$$y_p \in \{0,1\} \quad \forall p \in \Omega. \tag{5}$$

The objective function (3) minimize the duration of the chosen routes. Constraint (4) guarantees that each node is visited only once. Solving the LP-relaxation, resulting from relaxing the integrality constraints of the variables y_p, of the master problem provides a lower bound on its optimal value. The set of feasible paths Ω is usually very large making it hard to solve the LP-relaxation of the master problem. Therefore, we use column generation. In column generation, a restricted master problem is solved by considering only a subset $\Omega' \subseteq \Omega$ of feasible paths. Additional paths with negative reduced cost are generated after solving a pricing subproblem. The pricing problem for the TDVRPTW is (the index k is dropped):

$$\min z_P = \sum_{(i,j) \in A} \overline{\tau}_{ij}(\omega_i) x_{ij} \tag{6}$$

Furthermore, $\overline{\tau}_{ij}(\omega_i) = \tau_{ij}(\omega_i) - \pi_i$ is the arc reduced cost, where π_i is the dual variable associated with servicing node i. In the master problem, π_i results from the constraint corresponding to node i in the set of constraints (4). The objective function of the pricing problem can be expressed as:

$$z_P = e_p - s_p - \sum_{i \in V_c} a_{ip} \pi_i \tag{7}$$

or in the variables x_{ij} as:

$$z_P = e_p - s_p - \sum_{i \in V_c} \left(\pi_i \sum_{j \in \gamma^+(i)} x_{ij} \right) \tag{8}$$

4.1. The Pricing Problem

Solving the pricing problem involves finding columns (*i.e.* routes) with negative reduced cost that improve the objective function of master problem. In case of the TDVRPTW, this corresponds to solving the TDESPPRC and finding paths with negative cost. The TDESPPRC is a generalization of the ESPPRC in which costs are time-dependent. In this paper, we solve the pricing problem by means of a time-dependent labeling (TDL) algorithm which is a modification of the labeling algorithm applied to the ESPPRC. To speed up the TDL algorithm , a bi-directional search is performed in which labels are extended both forward from the depot (*i.e.* node 0) to its successors, and backward from the depot (*i.e.* node n+1) to its predecessors. While forward labels are extended to some fixed time t_m (*e.g.* the middle of the planning horizon) but not further, backward labels are extended to t_m but are allowed to cross t_m. Forward and backward labels are finally merged to construct complete tours. The running time of a labeling algorithm depends on the length of partial paths associated with its labels. A bi-directional search avoids generating long paths and therefore limits running times.

5. COMPUTATIONAL RESULTS

The open source framework COIN is used to solve the linear programming relaxation of the master problem. For our numerical study, we use the well known Solomon's data sets ([29]) that follow a naming convention of $DTm.n$. D is the geographic distribution of the customers which can be R (Random), C (Clustered) or RC (Randomly Clustered). T is the instance type which can be either 1 (instances with tight time windows) or 2 (instances with wide time windows). m denotes the number of the instance and n the number of customers that need to be served. Road congestion is taken into account by assuming that vehicles travel through the network using different speed profiles. We consider speed profiles with two congested periods. Speeds in the rest of the planning horizon (*i.e.* the depot's time window) are relatively high. We consider speed profiles that comply with data from real life. Furthermore, we assume three types of links: fast, normal and slow. Slow links might represent links within the city center, fast links might represent highways and normal links might represent the transition from highways to city centers. Moreover, without loss of generality, we assume that breakpoints are the same for all speed profiles as congestion tends to happen around the same time regardless of the link's type (*e.g.* rush hours).The choice for a link type is done randomly and remains the same for all instances. Our BCP framework is able to solve 75% of the instances with 25 customers, 50% of the instances with 50 customers, and 20% of the instances with 100 customers.

6. REFERENCES

[1] G. Laporte, "The vehicle routing problem: an overview of exact and approximate algorithms," *European Journal of Operational Research*, vol. 59, no. 3, pp. 345–358, 1992.

[2] P. Toth and D. Vigo, *The vehicle Routing Problem*. SIAM Monographs on Discrete Mathematics and Applications. SIAM, Philadelphia, 2002, vol. 9.

[3] G. Laporte, "What you should know about the vehicle routing problem," *Naval Research Logistics*, vol. 54, pp. 811–819, 2007.

[4] O. Bräysy and M. Gendreau, "Vehicle routing problem with time windows, part i: Route construction and local search algorithms," *Transportation Science*, vol. 39, no. 1, pp. 104–118, 2005.

[5] ——, "Vehicle routing problem with time windows, part ii: Metaheuristics," *Transportation Science*, vol. 39, no. 1, pp. 119–139, 2005.

[6] D. J. Bertsimas and D. Simchi-Levi, "A new generation of vehicle routing research: robust algorithms, addressing uncertainty," *Operations Research*, vol. 44, no. 2, pp. 286–304, 1996.

[7] D. J. Bertsimas and G. V. Ryzin, "A stochastic and dynamic vehicle routing problem in the euclidian plane," *Operations Research*, vol. 39, pp. 601–615, 1991.

[8] ——, "Stochastic and dynamic vehicle routing problems in the euclidean plane with multiple capcitated vehicles," *Operations Research*, vol. 41, pp. 60–76, 1993a.

[9] C. Malandraki and R. B. Dial, "A restricted dynamic programming heuristic algorithm for the time dependent traveling salesman problem," *European Journal of Operational Research*, vol. 90, pp. 45–55, 1996.

[10] A. Hill and W. Benton, "Modeling intra-city time-dependent travel speeds for vehicle scheduling problems," *European Journal of Operational Research*, vol. 43, no. 4, pp. 343–351, 1992.

[11] S. Ichoua, M. Gendreau, and J. Y. Potvin, "Vehicle dispatching with time-dependent travel times," *European Journal of Operational Research*, vol. 144, no. 2, pp. 379–396, 2003.

[12] C. Malandraki and M. Daskin, "Time dependent vehicle routing problems: formulations, properties and heuristic algorithms," *Transportation Science*, vol. 26, no. 3, pp. 185–200, 1992.

[13] O. Jabali, T. van Woensel, A. de Kok, C. Lecluyse, and H. Permans, "Time-dependent vehicle routing subject to time delay perturbations," *IIE Transaction*, vol. 41, pp. 1049–1066, 2009.

[14] T. Van Woensel, L. Kerbache, H. Peremans, and N. Vandaele, "Vehicle routing with dynamic travel times: a queueing approach," *European Journal of Operational Research*, vol. 186, no. 3, pp. 990–1007, 2008.

[15] B. Fleischmann, M. Gietz, and S. Gnutzmann, "Time-varying travel times in vehicle routing," *Transportation Science*, vol. 38, no. 2, pp. 160–173, 2004.

[16] W. Maden, R. Eglese, and D. Black, "Vehicle routing and scheduling with time-varying data: A case study," *Journal of the Operational Research Society*, vol. 61, no. 61, pp. 515–522, 2010.

[17] A. F. Donati, R. Montemanni, N. casagrande, A. E. Rizzoli, and L. M. Gambardella, "Time dependent vehicle routing problem with a multi colony system," *Eurorpean Journal of Operational Research*, vol. 185, pp. 1174–1191, 2008.

[18] A. Haghani and S. Jung, "A dynamic vehicle routing problem with time-dependent travel times," *Computers and Operations Research*, vol. 32, pp. 2959–2986, 2005.

[19] H. Hashimoto, M. Yagiura, and T. Ibaraki, "An iterated local search algorithm for the time-dependent vehicle routing problem with time windows," *Discrete Optimization*, vol. 5, pp. 434–456, 2008.

[20] K. N. Androutsopoulos and K. G. Zografos, "Solving the multi-criteria time-dependent routing and scheduling in a multimodal fixed scheduled network," *European Journal of Operational Research*, vol. 192, pp. 18–28, 2009.

[21] H. Tang, "Efficcient implementation of improvement procedures for vehicle routing with time-dependent travel times," *Transportation Research Record*, pp. 66–75, 2008.

[22] M. E. Lübbecke and J. Desrosiers, "Selected topics in column generation," *Operations Research*, vol. 53, no. 6, pp. 1007–1023, 2005.

[23] N. Kohl, J. Desrosiers, O. B. G. Madsen, M. M. Solomon, and F. Soumis, "2-path cuts for the vehicle routing problem with time windows," *Transportation Science*, vol. 33, no. 1, pp. 101–116, 1999.

[24] W. Cook and J. L. Rich, "A parallel cutting plane algorithm for the vehicle routing problem with time windows," *Technical Report TR99-04, Computational and Applied Mathematics, Rice University, Housten, USA*, 1999.

[25] M. Jespen, B. Petersen, S. Spoorendonk, and D. Pisinger, "Subset-row inequalities applied to the vehicle-routing problem with time windows," *Operations Research*, vol. 56, no. 2, pp. 497–511, 2008.

[26] G. Desaulniers, F. Lessard, and A. Hadjar, "Tabu search, partial elementarity, and generalized k-path inequalities for the vehicle routing problem with time windows," *Transportation Science*, vol. 42, no. 3, pp. 387–404, 2008.

[27] R. Baldacci, A. Mingozzi, and R. Roberti, "New route relaxation and pricing strategies for the vehicle routing problem," *Working paper, the university of Bologna*, 2010.

[28] T. Van Woensel and N. Vandaele, "Empirical validation of a queueing approach to uninterrupted traffic flows," *4OR, A Quarterly Journal of Operations Research*, vol. 4, no. 1, pp. 59–72, 2006.

[29] M. M. Solomon, "Algorithms for the vehicle routing and scheduling problems with time window constraints," *Operations Research*, vol. 35, no. 2, pp. 254–265, 1987.

An algorithm based on Iterated Local Search and Set Partitioning for the Vehicle Routing Problem with Time Windows

S. Ribas [*] A. Subramanian [*] I. M. Coelho [*] L. S. Ochi [*] M. J. F. Souza [†]

[*] Universidade Federal Fluminense
Rua Passo da Pátria, 156 - Bloco E - Niterói, Brazil
{sribas, anand, imcoelho, satoru}@ic.uff.br

[†] Universidade Federal de Ouro Preto
Campus Universitário, Morro do Cruzeiro, Ouro Preto, Brazil
marcone@iceb.ufop.br

ABSTRACT

The Vehicle Routing Problem with Time Windows is a well known optimization problem and it has received a lot of attention in operational research literature. This work proposes a hybrid algorithm that combines the Iterated Local Search metaheuristic, the Variable Neighborhood Descent method and an exact Set Partitioning model for solving it. The computational results demonstrate that the proposed hybrid approach is quite competitive, since out of the 56 test problems considered, the algorithm improved the best known solution in 12 cases and equaled the result of another 27.

Keywords: Vehicle Routing Problem with Time Windows, Hybrid Algorithm, Iterated Local Search, Set Partitioning

1. INTRODUCTION

The Vehicle Routing Problem with Time Windows (VRPTW) is a well known optimization problem and it has received a lot of attention in operational research literature. In this problem, a fleet of vehicles must leave the depot, serve customer demands, and return to the depot, at minimum cost, without violating the capacity of the vehicles as well as the time window specified by each customer.

There are two main reasons (operational and theoretical) for investing in research to develop new algorithms for the efficient resolution of this problem. From the practical/operational point of view, the costs related to transporting people or merchandise are generally high, with a tendency to increase, motivated by the actual expansion of commerce of all types [1]. Researchers calculate that 10% to 15% of the final cost of the merchandise commercialized in the world is due to its transport [2]. From the theoretical aspect, since the VRP and most of its variants, including the VRPTW, are NP-hard problems [3], the efficient resolution of these problems represents a challenge for researchers, who, in general, opt for heuristic approaches. The size of this challenge is demonstrated by the great number of articles dealing with this type of problem.

The VRPTW has been dealt with various objectives and, in the present work, the aim is to minimize the total traveling distance which is one of the most commonly found in literature.

Given the complexity of the problem, its resolution using pure exact methods is often an extremely arduous task due the large amount of computational time required. This fact has motivated the development of new heuristic algorithms for solving VRPTW. It is noteworthy to mention that such algorithms aims at finding near-optimal solutions using less computational effort.

The algorithm proposed in this article for solving VRPTW combines the concepts of Iterated Local Search metaheuristic, the Variable Neighborhood Descent method and an exact Set Partitioning model, which periodically determines the best combination of routes generated during the execution of the algorithm.

2. PROPOSED METHODOLOGY

This section explains the proposed hybrid algorithm. Section 2.1 presents the data structure used to represent a VRPTW solution, while Section 2.2 describes the penalty-based function that evaluates a solution for the problem. Next, Section 2.3 demonstrates the procedure used to construct the initial solution; and Section 2.4 describes the used neighborhood structures. Finally, Section 2.5 presents the proposed algorithm.

2.1. Solution representation

A route r is defined by a sequence of integer numbers that corresponds to the identifiers of the customers in r. A solution s contains a set of routes.

2.2. Evaluation function

A solution s is evaluated by the function f, given by the equation (1), which must be minimized:

$$f(s) = \sum_{r \in s} g(r) = \sum_{r \in s} (c(r) + w_l.l(r) + w_e.e(r)) \qquad (1)$$

where: g is a function that evaluates routes; $c(r)$ is the cost of the route r; $l(r)$ corresponds to the lateness time for r; $e(r)$ is the load excess in the route r; w_l and w_e are penalties per unit of delay and excess load, respectively. They were empirically set in $w_l = 200$ and $w_e = 300$.

Notice that when s is feasible, the value given by f will only correspond to the travel cost, since in this case: $l(r) = e(r) = 0, \forall r \in s$.

2.3. Constructive procedure

To obtain an initial solution for the VRPTW, a cheapest insertion method, called *CI-POP*(), that explores the Proximate Optimality Principle [4] was developed. According to this principle, in an optimal sequence of choices, each sub-sequence should also be optimal. It is worth mentioning that although this principle deals

with optimal cases, in the developed algorithm there is no guarantee that the optimal solution will be obtained, or even parts of the optimal solution. Thus, this principle is only employed to generate better initial solutions.

Let $|K|$ be the maximum number of available vehicles. Initially, the constructive algorithm creates $|K|$ empty routes and a list of candidates to be inserted in the set of routes. The idea of the procedure is to iteratively insert each candidate customer in the best location. A local search is periodically performed in the partial solution. More specifically, the parameters of the method were calibrated in such a way that five local searches occur during the construction; for example, if there is a total of 100 customers, the local search is performed for every twenty customers added to the partial solution. In this case, the local search is performed using the RVND (see Section 2.5.2). The procedure terminates when all customers have been added.

2.4. Neighborhood structures

In order to explore the solution space, 10 neighborhood structures are used, where six of these modify two routes at each movement performed (inter-route), while the other four modify only a single route (intra-route). The inter-route neighborhood structures are generated by the following movements: *Shift*$(1,0)$, *Shift*$(2,0)$, *Shift*$(3,0)$, *Swap*$(1,1)$, *Swap*$(2,1)$ and *Swap*$(2,2)$. A movement of the neighborhood structure *Shift*$(k,0)$ involves transferring k adjacent customers from route r_1 to another route r_2; and a movement of the type, *Swap*(k,l), interchanges k adjacent customers from route r_1 to l other adjacent customers from another route r_2.

As for those neighborhood structures that only modify one route at a time, the following movements are used: *Exchange*, *Shift'*(1), *Shift'*(2) and *Shift'*(3). The *Exchange* neighborhood involves the permutation between two customers of the same route and it can be seen as an intra-route version of the *Swap*$(1,1)$ neighborhood. The other three neighborhoods can be considered as intra-route versions of the *Shift*$(1,0)$, *Shift*$(2,0)$ e *Shift*$(3,0)$ neighborhoods, respectively.

2.5. Proposed algorithm

The proposed algorithm, called Intensified Iterated Local Search (IILS-SP), involves the construction of an initial solution according to the procedure presented in Section 2.3, followed by a local search that combines adapted versions of the Iterated Local Search (ILS) and Variable Neighborhood Descent (VND) methods with an exact approach based on the mathematical formulation of the Set Partitioning (SP). The pseudo-code of IILS-SP is presented in Algorithm 1. Let s_0 be an initial solution; s^* the best solution obtained during the procedure execution; s' a perturbed solution; and, s'' a local optimal solution obtained by the application of the RVND to the perturbed solution.

The following sections detail each part of this algorithm.

2.5.1. Intensified Iterated Local Search

Intensified Iterated Local Search is an extension of the *Iterated Local Search* – ILS [5] metaheuristic. ILS explores the solution space by applying perturbations to the current local optimal solution. This metaheuristic starts with the initial solution s_0 and applies a local search to it, obtaining s^*. Next, the method iteratively performs the following steps: (*i*) perturbs the current best solution s^*; (*ii*) obtains a solution s'; and (*iii*) performs a local search in s', obtaining a local optimal s''. If s'' is better than the current best solution s^*, then the method transforms s'' into the new current solution. Otherwise, the method performs another iteration. This procedure is repeated until the stopping criterion is met.

Algorithm 1: IILS-SP()

```
1  s_0 ← CI-POP()
2  s* ← RVND(s_0)
3  repeat
4  |   s' ← Perturbation(s*, history)
5  |   s'' ← RVND(s')
6  |   if AppropriatedMoment(history) then
7  |   |   s'' ← Intensification (s'')
8  |   end
9  |
10 |   s* ← AcceptanceCriterion(s'', s*, history)
11 until stopping criterion not met
12 return s*
```

It is important to emphasize that ILS's success strongly depends on the perturbations performed. This way, the perturbation applied to a given solution should be proportioned in such a way that the resulting modification is sufficient to escape from local optima and to explore different regions of the search space, but keeping some characteristics of the current best solution, in order to avoid a complete random restart in next iterations.

In this work, a perturbation (line 4 of Algorithm 1) consists of applying $p+2$ movements randomly chosen in the neighborhood *Shift*, presented in Section 2.4, where $p \in \{0,1,2,\dots\}$ represents the perturbation level. This way, the greater this value, the greater the number of modifications performed in the solution. Herein, *ILSmax* iterations without improvement are applied in the same perturbation level. When this value is achieved, the perturbation level is increased.

In this case, the local search of the IILS (lines 2 and 5 of Algorithm 1) is performed using the Variable Neighborhood Descent with random neighborhood ordering, denoted by RVND and described in Section 2.5.2.

Finally, the proposed algorithm contains an intensification module (line 7 of Algorithm 1). This module is activated at appropriate moments of the search and invokes a mathematical programming procedure, based on Set Partitioning, to find the optimal set of routes among those generated during the search. More specifically, the partitioning model is applied to the set formed by all the routes belonging to the solutions generated after the local search phase of the IILS algorithm. That is, for each IILS iteration, the routes of the solution s'' (line 5 of Algorithm 1) are added to the set to be partitioned. This is done in such a way that there are no repeated routes in the set, which has an unlimited size.

A description of this module is given in Section 2.5.3.

2.5.2. Variable Neighborhood Descent with random neighborhood ordering

The procedure Variable Neighborhood Descent (VND) [6] involves an exhaustive exploration of the solution space by means of systematic exchanges of the neighborhood structures. During the local search, only the solution that is better than the current best solution is accepted. When a better solution is found, the method restarts the search, beginning with the first neighborhood structure.

The method VND is based on three principles: (*i*) a local optimum for a given neighborhood structure does not necessarily correspond to a local optimum of another neighborhood structure; (*ii*) a global optimum corresponds to a local optimum for all neighborhood structures; and (*iii*) for many problems, the local optimum of

a given neighborhood structure is close to the local optima of other neighborhood structures.

The latter principle, of empirical nature, indicates that a local optimum frequently gives some type of information about the global optimal. This is the case in which local and global optimum share a lot of variables with the same value.

The classical version of VND searches local optimal solutions following a fixed order of neighborhood structures. This strategy is widely applied and the results in literature confirm its efficiency. However, for the results presented in this work, a random order was used to explore the neighborhoods. This strategy is adopted with success in [7]. Here, this strategy is so-called RVND.

2.5.3. Set partitioning model

The intensification phase of the proposed algorithm involves the exact resolution of a Set Partitioning Problem (SPP). Let $\mathscr{R}$ be the subset of routes generated by the IILS-algorithm and let y_j, $\forall j \in \mathscr{R}$, be the binary variables that indicate if the route $j \in \mathscr{R}$ is part of the solution ($y_j = 1$); or not ($y_j = 0$). Each route $j \in \mathscr{R}$ has an associated cost g_j. The parameter m_{ij} is equal to 1 if the customer $i \in N$ is attended by the route $j \in \mathscr{R}$; and 0, otherwise. The mathametical formulation is as follows.

$$Minimize \sum_{j \in \mathscr{R}} g_j y_j \tag{2}$$

$$\sum_{j \in \mathscr{R}} m_{ij} y_j = 1, \forall i \in N \tag{3}$$

$$\sum_{j \in \mathscr{R}} y_j \leq |K| \tag{4}$$

$$y_j \in \{0,1\}, \forall j \in \mathscr{R} \tag{5}$$

The objective of this formulation is to find a set of routes that attend the constraints of the problem with a minimum cost (2). Constraints (3) guarantee that each customer is visited by exactly one route. Constraints (4) ensure that a solution contains up to $|K|$ routes. Constraints (5) define the domain of the variables.

In this work, the SPP model was implemented using ILOG API Concert for C++ and solved by the CPLEX optimizer, version 12.

3. COMPUTATIONAL RESULTS

The proposed algorithm (IILS-SP) was developed in C++ programming language and tested in a computer with an Intel Quad Core 2.4 GHz microprocessor with 8 GB RAM and operational system Ubuntu Linux 9.10 Kernel 2.6.31.

IILS-SP was applied to solve the set of instances proposed by Solomon [8], which is well known in the literature.

For each of the 56 instances, five runs were performed using a 10-minute processing time limit for each run as stopping criterion[1]. The algorithm was empirically calibrated and the parameters were fixed as follows: (*i*) in the construction of an initial solution, as customers are being inserted, five local searches were performed as described in Section 2.3; (*ii*) the number of no-improvement iterations at a given level of perturbation of IILS was fixed as 20; (*iii*) the procedure is iteratively performed according to the *Multi-Start* [9] method, where at each iteration, an initial solution is constructed by a non-deterministic method described in the Section 2.3 and a local search is performed by IILS-SP; and (*iv*) the maximum processing time for each execution of the mathematical solver in the intensification phase was limited to 5 seconds.

[1]The computational results of this research are available at `http://www.decom.ufop.br/sabir/shared/2011alio-vrptw-results.zip`

Table 1: Comparisons between different works that optimize the total distance traveled

Class		Work*					This work
		RT95	CA99	SC00	AL07	OV08	
C1	NV	10.00	10.00	10.00	10.00	10.00	10.00
	TD	**828.38**	**828.38**	**828.38**	**828.38**	**828.38**	**828.38**
C2	NV	3.00	3.00	3.00	3.00	3.00	3.00
	TD	**589.86**	596.63	**589.86**	**589.86**	**589.86**	**589.86**
R1	NV	12.16	12.42	12.08	13.25	13.33	13.17
	TD	1208.50	1233.34	1211.53	1183.38	1186.94	**1181.03**
R2	NV	2.91	3.09	2.82	5.55	5.36	5.36
	TD	961.71	990.99	949.27	899.90	**878.79**	883.10
RC1	NV	11.87	12.00	11.88	12.88	13.25	12.75
	TD	1377.39	1403.74	1361.76	1341.67	1362.44	**1338.54**
RC2	NV	3.37	3.38	3.38	6.50	6.13	6.13
	TD	1119.59	1220.99	1097.63	1015.90	**1004.59**	1009.17
All	CNV	414	420	412	489	488	482
	CTD	57231	58927	56830	55134	55021	**54842**

* RT95 [10], CA99 [11], SC00 [12], AL07 [1] and OV08 [13]

In summary, the best solutions found during the executions by the IILS-SP were: 100% (9/9) tied values for C1; 100% (8/8) tied values for C2; 33.3% (4/12) improved and 41.6% (5/12) tied values for R1; 27.3% (3/11) improved and 9.1% (1/11) tied values for R2; 37.5% (3/8) improved and 37.5% (3/8) tied values RC1; and 25% (2/8) improved and 12.5% (1/8) tied values for RC2. Overall, the values improved in 21.4% (12/56) of the cases, the values tied in 48.2% (27/56) and the values decreased in 30.4% (17/56).

The algorithm proved to be robust, since it presented relatively small gaps. In 80.4% (45/56) of the analyzed instances, gap was less that 1.0%. When this value was improved, the gap was always smaller than 4.16% (as in the R208). These results show that the algorithm produces final solutions with quite little variability in terms of solution quality. In addition, in some cases (R110, R202 e RC105) the proposed algorithm produced better results in average than those found in literature.

Table 1 presents the results of different researches that had as a primary objective the minimization of the total distance traveled. The columns represent the algorithm whereas the lines show the average number of vehicles and the total distance traveled of the best solutions obtained for each class. For each algorithm, the average results with respect to Solomon's benchmarks are reported with respect to number of vehicles ("NV") and total distance ("TD"). CNV and CTD indicate, respectively, the cumulative number of vehicles and cumulative total distance over all the 56 instances. When observing the results of each group separately, the conclusion is that the algorithm values tied with those of the best results found in literature in the cluster groups of C1 and C2, and outperformed them in the groups of R1 and RC1. In the R2 and RC2 groups, although the results were close, they were not able to improve the values of the other groups. Therefore, when considering the overall scenario, IILS-SP outperformed all the others algorithms in terms of solution quality.

To verify the influence of the intensification phase of IILS-SP over its version without this strategy, named ILS, computational experiments were carried out according to Aiex *et al.* [14]. In each experiment, we measure the CPU time to find or improve the target value. For each instance/target pair, the n running times are sorted in increasing order. We associate with the i-th sorted running time $t(i)$ a probability $p(i) = (i-1/2)/n$, and plot the points $z(i) = [t(i), p(i)]$, for $i = 1, ..., n$. Figure 1 illustrates this cumulative probability distribution plot for IILS-SP and ILS algorithms, using the R208 instance and having as target a value 5% far from the best known value.

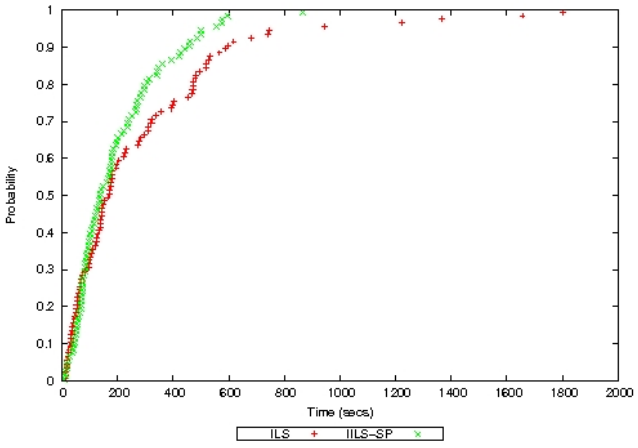


Figure 1: Cumulative probability distribution

This Figure clearly shows that IILS-SP is able to find a cost function value at least as good as the given target value faster than the ILS algorithm.

4. CONCLUSIONS

This paper presents a hybrid algorithm for the Vehicle Routing Problem with Time Windows. The proposed algorithm, so-called IILS-SP, combines the Iterated Local Search metaheuristic, the Variable Neighborhood Descent method and an exact Set Partitioning model that, periodically, performs the best combination of the routes generated along the algorithm. Hence, the IILS-SP combines the flexibility of heuristic methods and the power of mathematical programming.

The IILS-SP was tested in 56 well-known VRPTW instances and the results were compared with the best solutions found in literature. The computational results show that the proposed hybrid approach is quite competitive, since out of the 56 test problems considered, the algorithm improved the best known solution in 12 cases and equaled the result of another 27.

5. ACKNOWLEDGEMENTS

The authors acknowledge CAPES, CNPq and FAPEMIG for supporting the development of this research.

6. REFERENCES

[1] G. B. Alvarenga, G. R. Mateus, and G. de Tomi, "A genetic and set partitioning two-phase approach for the vehicle routing problem with time windows," *Computers and Operations Research*, vol. 34, pp. 1561–1584, 2007.

[2] G. F. King and C. F. Mast, "Excess travel: causes, extent and consequences," *Transportation Research Record*, no. 1111, pp. 126–134, 1997.

[3] J. K. Lenstra and A. H. G. R. Kan, "Complexity of vehicle routing and scheduling problems," *Networks*, vol. 11, no. 2, pp. 221–227, 2006.

[4] M. G. C. Resende and C. C. Ribeiro, "Grasp," in *Search Methodologies*, 2nd ed., E. K. Burke and G. Kendall, Eds. Springer (to appear), 2010, available at: http://www.ic.uff.br/$\sim$celso/artigos/grasp.pdf.

[5] H. R. Lourenco, O. C. Martin, and T. Stutzle, "Iterated local search," in *Handbook of Metaheuristics*, F. Glover and G. Kochenberger, Eds. Boston: Kluwer Academic Publishers, 2003, ch. 11.

[6] N. Mladenovic and P. Hansen, "A variable neighborhood search," *Computers and Operations Research*, vol. 24, pp. 1097–1100, 1997.

[7] A. Subramanian, L. Drummond, C. Bentes, L. Ochi, and R. Farias, "A parallel heuristic for the vehicle routing problem with simultaneous pickup and delivery," *Computers and Operations Research*, vol. 37, pp. 1899–1911, 2010.

[8] M. M. Solomon, "Algorithms for the vehicle routing and scheduling problem with time window contraints," *Operational Research*, vol. 35, pp. 254–265, 1987.

[9] R. Martí, "Multi-start methods," in *Handbook of Metaheuristics*, F. Glover and G. Kochenberger, Eds. Boston: Kluwer Academic Publishers, 2003, ch. 12.

[10] Y. Rochat and E. Taillard, "Probabilistic diversification and intensification in local search for vehicle routing," *Journal of Heuristics*, vol. 1, pp. 147–167, 1995.

[11] Y. Caseau and F. Laburthe, "Heuristics for large constrained vehicle routing problems," *Journal of Heuristics*, vol. 5, pp. 281–303, 1999.

[12] G. Schrimpf, J. Schneider, H. Stamm-Wilbrandt, and G. Dueck, "Record breaking optimization results using the ruin and recreate principle," *Journal of Computational Physics*, vol. 159, pp. 139–171, 2000.

[13] H. de OLIVEIRA and G. Vasconcelos, "A hybrid search method for the vehicle routing problem with time windows," *Annals of Operations Research*, 2008. [Online]. Available: http://dx.doi.org/10.1007/s10479-008-0487-y

[14] R. M. Aiex, M. G. C. Resende, and C. C. Ribeiro, "Probability distribution of solution time in grasp: An experimental investigation," *Journal of Heuristics*, vol. 8, pp. 200–2, 2000.

A medium term short sea fuel oil distribution problem

Agostinho Agra * Marielle Christiansen † Alexandrino Delgado ‡

* Department of Mathematics and CIDMA
University of Aveiro
aagra@ua.pt

† Department of Industrial Economics and Technology Management
Norwegian University of Science and Technology
marielle.christiansen@iot.ntnu.no

‡ Department of Mathematics
University of Cape Verde
alexandrino.delgado@unicv.edu.cv

ABSTRACT

We consider a real inventory routing problem occurring in the archipelago of Cape Verde, where an oil company is responsible for the inventory management of multiple fuel oil products and for the routing of ships between the islands. Inventory management considerations are taken into account at the demand side, but not at the supply side. Demands are assumed to be constant over the time horizon of several months. The objective of the company is to establish a medium term plan that satisfies demand requirements and minimizes the transportation costs. We present a formulation for the problem based on the one given by Christiansen (1999). Since this formulation provides large integrality gaps we discuss different extended formulations and compare them for a time horizon of fifteen days. In order to obtain feasible solutions for time horizons of several months, we construct a rolling horizon heuristic that uses the extended formulation that provided best computational results.

Keywords: Maritime transportation, Inventory, Routing, Extended Formulations

1. INTRODUCTION

We present a real maritime inventory routing problem occurring in the archipelago of Cape Verde. An oil company is responsible for the inventory management of different oil products in several tanks located in the main islands. The inventory management must be coordinated with the routing of ships between the islands in order to prevent shortfalls. We consider a time horizon of six months. This problem can be classified within maritime transportation as a short-sea medium-term inventory-routing problem.

Maritime transportation has received a clear increased interest in the last decade. Christiansen et al. [1] present a review on maritime transportation, and Christiansen and Fagerholt [2] is devoted to maritime inventory routing problems. Combined routing and inventory management within maritime transportation have been present in the literature the last decade only. See [3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16]. Most of these maritime inventory routing articles are based on real problems.

In Cape Verde, fuel oil products are imported and delivered to specific islands and stored in large supply storage tanks. From these islands, fuel oil products are distributed among all the inhabited islands using a small heterogeneous fleet of ships. These products are stored in consumption storage tanks. Some ports have both supply tanks for some products and consumption tanks of other products. Not all the islands consume all the products. In the medium term planning considered here capacities in supply tanks are ignored. However, for the consumption ports the capacity of the tanks for a particular product is usually less than the total demand over the planning horizon for that product making the inventory management an important issue.

Unlike the short term planning case, in the medium term planning the data is typically forecasted. Hence, safety stocks must be considered. We assume the demands are constant over the time horizon. Several important issues taken into account in a short term plan are relaxed or incorporated indirectly in the data. For instance, berth capacities and operating time windows at ports, that are essential in the short term plan, are ignored here. To transport the fuel oil products between the islands, the planners control a small, heterogeneous fleet consisting of two ships. Each ship has a specified load capacity, fixed speed and cost structure. The traveling times are an estimation based on the practical experience and include the travel time, set up times for the coupling and decoupling of pipes, operation times and an additional time to account for delays and waiting times.

The cargo hold of each ship is separated into several cargo tanks. The products cannot be mixed and cleaning operations are required when changing from dirty oil product to clean oil products. Again this issue is more relevant on a short term plan then in a medium term where the quantities transported and the traveling times are estimations. Hence we ignore this issue.

Given the initial stock levels at the demand tanks, the initial position (which can be a point at sea) and the quantities onboard each ship, the inter island distribution plan consists of designing routes and schedules for the fleet of ships including determining the (un)loading quantity of each product at each port. This plan must satisfy the safety stocks of each product at each island, the capacities of the ships and tanks. The transportation cost of the distribution plan is to be minimized.

Following Christiansen [4], we present an initial arc-load flow formulation for the problem. Since this formulation leads to large integrality gaps we discuss how to obtain tighter formulations. Using the (extended) formulation that provided better computational results to solve to optimality instances up to fifteen days we construct a rolling horizon heuristic that allows us to obtain feasible plans for horizons of several months.

2. FORMULATION

Following [4] we present an Arc-Load Flow (ALF) formulation. We divide the formulations in the following parts: routing constraints, loading and discharging constraints, time constraints and inventory constraints. Finally the objective function is presented.

Routing constraints:

Let V denote the set of ships. Each ship $v \in V$ must depart from its initial port position, that can be a point at sea, in the beginning of the planning horizon. The set of ports is denoted by N. Each possible port arrival is denoted by the pair (i,m) representing the m^{th} visit to port i. Direct ship movements (arcs) from port arrival (i,m) to port arrival (j,n) are represented by (i,m,j,n). We define S_A as the set of possible port arrivals (i,m), S_{Av} as the set of possible port arrivals made by ship v, and set S_{Xv} as the set of all possible movements (i,m,j,n) of ship v. We denote by $\underline{\mu}_i$ the minimum number of visits to port i.

For the routing we define the following binary variables: arc flow variables x_{imjnv} that are equal to 1 if ship v sails from port arrival (i,m) directly to port arrival (j,n), and 0 otherwise; variables xo_{imv} that indicate whether ship v sails directly from its initial position to port arrival (i,m) or not; w_{imv} is 1 if ship v visits port i at arrival number (i,m) and 0 otherwise; z_{imv} is equal to 1 if ship v ends its route at port arrival (i,m) and 0 otherwise, and y_{im} indicates whether a ship is visiting port arrival (i,m) or not.

The routing constraints are as follows:

$$\sum_{(j,n)\in S_{Av}} xo_{jnv} = 1, \qquad v \in V, \tag{1}$$

$$w_{imv} - \sum_{(j,n)\in S_{Av}} x_{jnimv} - xo_{imv} = 0, v \in V, (i,m) \in S_{Av}, \tag{2}$$

$$w_{imv} - \sum_{(j,n)\in S_{Av}} x_{imjnv} - z_{imv} = 0, \quad v \in V, (i,m) \in S_{Av}, \tag{3}$$

$$\sum_{v\in V} w_{imv} = y_{im}, \qquad (i,m) \in S_A, \tag{4}$$

$$y_{i(m-1)} - y_{im} \geq 0, \qquad (i,m) \in S_A : m > 1, \tag{5}$$

$$y_{i\underline{\mu}_i} = 1, \qquad i \in N. \tag{6}$$

Equations (1) ensure that each ship departs from its initial port position to some port arrival. Equations (2) and (3) are the flow conservation constraints, ensuring that a ship arriving at a port also leaves that port by either visiting another port or ending its route. Constraints (4) ensure that each port arrival (i,m) is visited at most once. Constraints (5) state that if port i receives the m^{th} visit, then it also must receive the $m-1^{th}$ visit. I Equations (6) guarantee the minimum number of visits at each port. These constraints are not necessary but computational experience showed that these constraints can be very important when good bounds are given for the minimum number of visits to each port.

Loading and discharging:

Let K represent the set of all products. Not all ports consume all products. Parameters J_i^k assume value 1 if port i is a supplier of product k;-1 if port i is a consumer of product k, and 0 if i is neither a consumer nor a supplier of product k. The quantity of product k on board ship v at the beginning of the planning horizon is given by Q_v^k. C_v is the total storage capacity of ship v. The minimum and the maximum discharge quantity of product k are given by $\underline{Q}_{im}^k$ and $\overline{Q}_{im}^k$ respectively.

In order to model the loading and unloading constraints we define the following binary variables: o_{imv}^k is equal to 1 if product k is loaded onto or unloaded from ship v at port arrival (i,m), and 0 otherwise; and the following continuous variables: q_{imv}^k is the amount of product k (un)loaded at port arrival (i,m), l_{imv}^k is the amount of product k onboard ship v when leaving from port arrival (i,m).

The loading and unloading constraints are given by:

$$x_{imjnv}[l_{imv}^k + J_j^k q_{jnv}^k - l_{jnv}^k] = 0, v \in V, (i,m,j,n) \in S_{Xv}, k \in K, \tag{7}$$

$$xo_{i1v}[Q_v^k + J_i^k q_{i1v}^k - l_{i1v}^k] = 0, \quad v \in V, (i,1) \in S_{Av}, k \in K, \tag{8}$$

$$\sum_k l_{imv}^k \leq C_v w_{imv}, \qquad v \in V, (i,m) \in S_{Av}, \tag{9}$$

$$\underline{Q}_{im}^k o_{imv}^k \leq q_{imv}^k \leq \overline{Q}_{im}^k o_{imv}^k, \qquad v \in V, (i,m) \in S_{Av}, \forall k \in K : J_i^k = -1, \tag{10}$$

$$\sum_k o_{imv}^k \geq w_{imv}, \qquad v \in V, (i,m) \in S_{Av}, \tag{11}$$

$$o_{imv}^k \leq w_{imv}, \qquad v \in V, (i,m) \in S_{Av}, k \in K, \tag{12}$$

$$o_{imv}^k \leq \sum_{(j,n)\in S_{Wv}} x_{jnimv}, \qquad v \in V, (i,m) \in S_{Av}, k \in K : J_i^k = -1, \tag{13}$$

Equations (7) ensure that if ship v sails from port arrival (i,m) to port arrival (j,n), then there must be satisfied the equilibrium of the quantity of product k on board each ship. These constraints can be linearized as follows:

$$l_{imv}^k + J_j^k q_{jnv}^k - l_{jnv}^k + C_v x_{imjnv} \leq C_v, \quad v \in V, (i,m,j,n) \in S_{Xv}, k \in K, \tag{14}$$

$$l_{imv}^k + J_j^k q_{jnv}^k - l_{jnv}^k - C_v x_{imjnv} \geq -C_v, v \in V, (i,m,j,n) \in S_{Xv}, k \in K. \tag{15}$$

Constraints (8) are similar to (7), and ensure the equilibrium of the load on board the ship for the first visit. These constraints can be replaced by the following linear constraints:

$$Q_v^k + J_i^k q_{i1v}^k - l_{i1v}^k + C_v xo_{i1v} \leq C_v, \quad v \in V, (i,1) \in S_{Av}, k \in K, \tag{16}$$

$$Q_v^k + J_i^k q_{i1v}^k - l_{i1v}^k - C_v xo_{i1v} \geq -C_v, v \in V, (i,1) \in S_{Av}, k \in K. \tag{17}$$

The ship capacity constraints are given by (9). To prevent undesirable situations such as a ship visiting a port to discharge a very small quantity, constraints (10) impose a lower and upper limits on the unload quantities. Constraints (11) ensure that if ship v visits port arrival (i,m), then at least one product must be (un)loaded. Constraints (12) ensure that if ship v (un)loads one product at visit (i,m), then w_{imv} must be one. Constraints (13) relate the variables o_{imv}^k to x_{jnimv}.

Time constraints:

Since the demand is assumed to be constant during the planning horizon we consider a continuous time model. In order to keep track of the inventory level it is necessary to determine the start and the end times at each port arrival. We define the following parameters: T_{ik}^Q is the time required to load/unload one unit of product k at port i; T_{ijv} is the traveling time between port i and j by ship v; T_{iv}^O indicates the traveling time required by ship v to sail from its initial port position to port i; T_i^B is the minimum interval between the departure of one ship and the next arrival at port i. T is a large constant.

Given the start and end time variables, t_{im} and t_{Eim} at port arrival (i,m), the time constraints can be written as:

$$t_{Eim} = t_{im} + \sum_{v} T^{Q}_{ik} q^{k}_{imv}, \qquad (i,m) \in S_A, \tag{18}$$

$$t_{im} - t_{Ei(m-1)} + T^{B}_{i} y_{i(m+1)} \geq T^{B}_{i}, (i,m) \in S_A : m > 1, \tag{19}$$

$$t_{Eim} + T_{ijv} - t_{jn} \leq T(1 - x_{imjnv}), v \in V, (i,m,j,n) \in S_{Xv}, \tag{20}$$

$$\sum_{v \in V} T^{O}_{iv} xo_{i1v} \leq t_{i1}, \qquad i \in N. \tag{21}$$

Constraints (18) define the end time of service of arrival (i,m). Constraints (19) impose a minimum interval between two consecutive visits at port i. Constraints (20) relate the end time of port visit (i,m) to the start time of port visit (j,n) when ship sails directly from port (i,m) to (j,n). Constraints (21) ensure that start time at port arrival $(i,1)$ occurs after a ship sails from its initial port position to port arrival $(i,1)$.

Inventory constraints:

Inventory constraints are considered for each unloading port i ($J^{k}_{i} = -1$). The demand rate R^{k}_{i} of product k at port i, as well as the minimum $\underline{S}^{k}_{i}$ and maximum $\overline{S}^{k}_{i}$ stock levels of each products k at the consumption ports are given. The parameter $\overline{\mu}_{i}$ denotes the maximum number of visits at port i.

We define the continuous variables s^{k}_{im} and s^{k}_{Eim} indicating the stock level at the start and end of port visit (i,m), respectively. The inventory constraints are as follow:

$$s^{k}_{im} + \sum_{v} q^{k}_{imv} + R^{k}_{i} t_{Eim} - R^{k}_{i} t_{im} - s^{k}_{Eim} = 0, (i,m) \in S_A, k \in K, \tag{22}$$

$$s^{k}_{Eim} + R^{k}_{i} t_{i(m+1)} - R^{k}_{i} t_{Eim} - s^{k}_{i(m+1)} = 0, \quad (i,m) \in S_A, k \in K \tag{23}$$

$$\underline{S}^{k}_{i} \leq s^{k}_{im}, s^{k}_{Eim} \leq \overline{S}^{k}_{i}, \qquad \forall (i,m) \in S_A, k \in K \tag{24}$$

$$\underline{S}^{k}_{i} \leq s^{k}_{Ei\overline{\mu}_{i}} + R^{k}_{i}(T - t_{Ei\overline{\mu}_{i}}) \leq \overline{S}^{k}_{i}, \qquad \forall i \in N, k \in K, \tag{25}$$

Equations (22) calculate the stock level of each product when the service ends at port arrival (i,m). Similarly, equations (23) relate the stock level at the start of port arrival $(i,m+1)$ to the stock level at the end of port visit (i,m). The upper and lower bound on the stock levels are ensured by constraints (24). Constraints (25) ensure that the stock level at the end of the planning horizon is within the stock limits.

Objective function:

The objective is to minimize the total routing cost:

$$\sum_{v \in V} \sum_{(i,m,j,n) \in S_{Xv}} C_{ijv} x_{imjnv} \tag{26}$$

3. EXTENDED FORMULATION

The arc-load flow model provides, in general, large integrality gaps. In order to improve the original formulation, we propose and test, for periods up to 15 days, different reformulations. Next we introduce the formulation that provided best computational results.

Define f^{k}_{imjnv} as the amount of product k that ship v transports from port arrival (i,m) to port arrival (j,n), and fo^{k}_{i1v} as the amount of product k that ship v transports from its initial port position to port arrival $(i,1)$.

Using these additional variables we can provide the following Arc Flow (AF) formulation:
min (26) subject to (1)-(6), (10)-(25), and

$$fo^{k}_{j1v} + \sum_{(i,m) \in S_{Av}} f^{k}_{imjnv} + J^{k}_{j} q^{k}_{jnv} = \sum_{(i,m) \in S_{Av}} f^{k}_{jnimv}, \quad v \in V, (j,n) \in S_{Av}, \forall k \in K \tag{27}$$

$$\sum_{k \in K} f^{k}_{imjnv} \leq C_{v} x_{imjnv}, \quad v \in V, (i,m,j,n) \in S_{Xv} \tag{28}$$

Constraints (27) ensure the equilibrium on the quantity on board the ship, and constraints (28) impose un upper bound of the capacity of ship v.

It can be shown that the AF formulation is stronger than the ALF formulation. For the computational experiments we considered 11 instances based on real data and used a computer with processor Intel Core 2 Duo, CPU 2.2GHz, with 4GB of RAM using the optimization software Xpress Optimizer Version 21.01.00 with Xpress Mosel Version 3.2.0. For a short time horizon of 15 days, the running times were, in average, lower when the ALF formulation was used. Of course, other extended formulations, as multicommodity type formulations that are not presented here, provide best lower bounds but higher average running times. The following table compares the average integrality gaps and average running time obtained with each formulation. Both formulations have been tightened with valid inequalities imposing a minimum number of operations to each port.

Formulation	Gap(%)	Time (seconds)
ALF	31.8	162
AF	28	129

4. ROLLING HORIZON HEURISTIC

Considering a planning horizon of several months, the tested instances become too large to be solved to optimality by a commercial software. To provide feasible solutions we develop a rolling horizon heuristic. The main idea of the rolling horizon heuristics is to split the planning horizon into smaller sub-horizons, and then repeatedly solve limited and tractable mixed integer problem for the shorter sub-horizons.

Rolling horizon heuristics have been used intensively in the past, see for instance the related works [17, 18, 19, 20].

In each iteration k (excepted the first one), the sub-horizon considered is divided into three parts: (i) a frozen part where binary variables are fixed; (ii) a central part (CP_k) where the nature of the variables is kept, and (iii) a forecasting period (FP_k) where binary variables are relaxed. We assume that the central and forecasting periods have equal length. The central period in iteration k becomes the frozen period in iteration $k+1$, and the forecasting period from iteration k becomes part of the central period in iteration $k+1$, see Figure 1. The process is repeated until the whole planning horizon in covered. In each iteration the limited mixed integer problem is solved using the AF formulation. When moving from iteration k to iteration $k+1$ we (a) fix the value of the binary variables, (b) update the initial stock level of each product at each port, (c) calculate the quantity of each product onboard each ship, and (d) update, for each ship, the initial position and the travel time and cost from that position to every port. Computational results are reported.

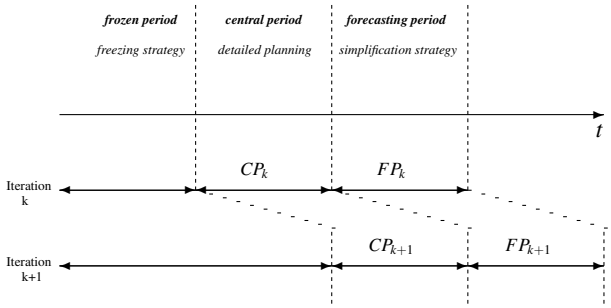


Figure 1: *The rolling horizon heuristic*

5. REFERENCES

[1] M. Christiansen, K. Fagerholt, B. Nygreen, and D. Ronen, "Maritime transportation. In: C. Barnhart, G. Laporte (Eds.)," *Handbook in Operations Research and Management Science*, vol. 14, no. 96, pp. 189–284, 2007.

[2] M. Christiansen and K. Fagerholt, "Maritime inventory routing problem. In: C. Floudas, P. Pardalos,(Eds.)," *Encyclopedia of optimization, second edition. Springer*, pp. 1947–1955, 2009.

[3] F. Al-Khayyal and S. J. Hwang, "Inventory constrained maritime routing and scheduling for multi-commodity liquid bulk, part I: Applications and model," *European Journal of Operational Research*, no. 176, pp. 106–130, 2007.

[4] M. Christiansen, "Decomposition of a combined inventory and time constrained ship routing problem," *Transportation Science*, vol. 33, no. 14, pp. 3–6, February 1999.

[5] M. Christiansen and K. Fagerholt, "Robust ship scheduling with multiple time windows," *Naval Research Logistics*, vol. 49, no. 15, pp. 611–625, February 2002.

[6] M. Christiansen and B. Nygreen, "A method for solving ship routing problems with inventory constraints," *Annals of Operations Research*, no. 22, pp. 357–378, 1998a.

[7] J. Desrosiers, Y. Dumas, M. Solomon, and F. Soumis, "Time constrained routing and scheduling. In: M. O. Ball, T. L. Magnanti, C. L. M., Nemhauser, G. L. (Eds.) ," *Handbooks in Operations Research and Management Science 8, Network Routing. North-Holland, Amsterdam*, pp. 35–139, 1995.

[8] K. Fagerholt, "Ship scheduling with soft time windows - an optimization based approach," *European Journal of Operational Research*, no. 131, pp. 559–571, 2001.

[9] T. Flatberg, H. Haavardtun, O. Kloster, and A. Løkketangen, "Combining exact and heuristic methods for solving a vessel routing problem with inventory constraints and time windows," *Ricerca Operativa*, vol. 29, no. 91, pp. 55–68, 2000.

[10] R. Giesen, J. Muñoz, M. Silva, and M. Leva, "Método de solucíon al problema de ruteo e inventarios de múltiplos produtos para una frota heterogénea de naves," *Ingeniería de Transporte*, vol. 13, no. 1, pp. 31–40, 2007.

[11] R. Grønhaug, M. Christiansen, M. Desaulniers, and G. Desrosiers, "A branch-and-price method for a liquefied natural gas inventory routing problem," *Transportation Science*, vol. 44, no. 3, pp. 400–415, 2010.

[12] S. J. Hwang, "Inventory constrained maritime routing and scheduling for multi-commodity liquid bulk," Ph.D. dissertation, Georgia Institute of Technology, Atlanta, 2005.

[13] D. Ronen, "Marine inventory routing: Shipments planning." *Journal of the Operational Research Society*, vol. 53, no. 1, pp. 108–114, 2002.

[14] H. Sherali, S. Al-Yakoob, and M. Hassan, "Fleet management models and algorithms for an oil-tanker routing and scheduling problem," *IIE Transactions*, vol. 31, no. 5, pp. 395–406, 1999.

[15] M. Stålhane, J. Rakke, H. Moe, R. Andersson, M. Christiansen, and K. Fagerholt, "A construction and improvement heuristic for a liquefied natural gas inventory routing problem," *Submitted to Journal*, 2010.

[16] N. Siswanto, D. Essam, and R. Sarker, "Solving the ship inventory routing and scheduling problem with undedicated compartments," *Computers and Industrial Engineering, DOI: 10.1016/j.cie.2010.06.011*, 2010.

[17] C. Mercé and G. Fontain, "Mip - based heuristics for capacitated lotsizing problems," *International Journal of Productions Economics*, no. 85, pp. 97–111, 2003.

[18] D. Bredström and M. Rönnqvist, "Supply chain optimization in pulp distribution using a rolling horizon solution approach," *NHH Dept. of Finance and Management Science Discussion Paper*, no. 17, December 2006.

[19] J. Rakke, M. Stålhane, C. Moe, M. Christiansen, H. Andersson, K. Fagerholt, and I. Norstad, "A rolling horizon heuristic for creating a liquefied natural gas annual delivery program," *Transportation Research Part C: Emerging Technologies,doi:10.1016/j.trc.2010.09.006*, 2010.

[20] M. Savelsbergh and Jin-Hwa Song, "Inventory routing with continuous moves," *Computers and Operations Research*, vol. 34, no. 6, pp. 1744 – 1763, 2007.

Nash Equilibria in Electricity Markets

Margarida Carvalho * João P. Pedroso * João Saraiva †

* INESC Porto and Faculdade de Ciências, Universidade do Porto
Rua do Campo Alegre, 4169-007 Porto, Portugal
mmsc@inescporto.pt jpp@fc.up.pt

† INESC Porto and Faculdade de Engenharia, Universidade do Porto
Rua Dr. Roberto Frias, 4200 - 465 Porto, Portugal
jsaraiva@fe.up.pt

ABSTRACT

Nash equilibria are solutions for many problems arising in Economics. In a restructured electricity sector, the pool market can be seen as a game where some players, the producers, submit their proposals. The profits of each producer depend on the proposals of the others. So, in this context, the strategies reached by the producers in a Nash equilibria are the best solutions for them. Here, we present our work in the development of techniques that can be used for determining Nash equilibria for this game.

Keywords: Nash Equilibria, Energy Sector, Adjustmet Process, Electricity Markets

1. INTRODUCTION

At the end of the 19^{th} century, electricity started to be generated, transported and distributed through low power networks with small geographic size. The companies were vertically integrated and there was not any competition in this sector. This kind of organization in the electricity market implies that: the consumers could not choose an electricity company to be supplied, the prices were defined in an administrative and sometimes unclear way, planning activities were made with less complexity than today (also because the economic environment was less volatile). Therefore, before the oil crises (1973), the electricity companies easily made forecasts, because the risk or uncertainty were not prior concerns. This situation changed in the beginning of the 70's: high inflation and interest rates made the economic environment more volatile. Adding to this fact, the evolution of technology forced the deregulation of the electric sector and its vertical unbundling. Thus, new companies were established and market mechanisms were implemented in order to generate competition conditions (see [1]).

Nowadays, in this restructured sector, many electricity markets are based on a pool-based auction for the purchase and sale of power. In this work, we apply game theory to this problem, in particular the notion of Nash equilibria. We look at this pool-based auction as a game, in which producers are the players that have to choose a strategy (or proposal) to submit. Here, their goal is to submit one that maximizes the profit. A Nash equilibrium is a set of strategies for each player, where nobody wants to change unilaterally his behaviour(see [2]). So, in the electricity pool market, we are interested in finding such equilibrium strategies for the producers, since it is the best answer that we can give to this non-cooperative game.

There is some literature related to this subject. For example, [3] and [4] study strategic bidding in electricity pool markets, with elastic and inelastic demand, respectively. We assumed almost inelastic demand because in the current markets this is the most realistic assumption: the consumers will pay (almost) anything to meet the demand.

The authors of [5] considered the case of constant, stochastic demand. They used the methodology of [4] to eliminate the bilinear terms of the generation companies' profit maximization problem, using a piecewise linear function and binary variables. They also contributed with a procedure that has the aim of finding all Nash equilibria in pure strategies. There, the proposals' prices and quantities take discrete values, unlike in our work, which focuses on the use of methods to compute Nash equilibria in games with finite strategies. However, as reported by the authors of [6], the discretization of the space of strategies can artificially eliminate some true Nash equilibria and add some equilibria that do not exist in the original game. In [7] it is proposed a fast computation of Nash equilibria in pure strategies by observing their properties; in that work, discretization is not required.

To approach this problem, we present an adjustment process that could be seen as a learning process by companies generating electricity. When this process converges we find a Nash equilibrium in pure strategies.

This extended abstract is organized as follows: Section 2 presents the electricity market model, Section 3 clarifies the concept of Nash equilibrium, explains the developed approach to achieve them and presents an example, Section 4 presents our future work in this problem.

2. ELECTRICITY MARKET MODEL

In the pool market, consumers and producers submit their proposals to buy and sell electricity. Here we assume very inelastic demand, that is characterized by the real constants $m < 0$ and b, in such a way that the demand is represented by almost a vertical segment $y = mx + b$. The generation companies simultaneously submit their proposals, that corresponds to pairs of quantity (MWh) and price (\$/MWh). Let n be the number of the selling proposals. For each hour of the day we have a matrix M that contains all the information about the proposals of the producers and the generation costs. This matrix has the following form, for each row $j \in \{1,2,\ldots,n\}$:

$$M_j = \begin{bmatrix} j & s_j & E_j & \lambda_j & c_j \end{bmatrix}$$

where the proposals are indexed by $j = 1,2,\ldots,n$, s_j is the producer of proposal j, E_j is the proposal quantity in MWh, λ_j is the price in \$/MWh and c_j is the associated marginal cost.

Then the market operator, an independent agent, carries out an economic dispatch, ED, once the price and quantity bids of the producers were submitted. He wants to find which proposals should

be dispatched so that the demand is satisfied and the market clearing price P_d is minimized. The market operator organizes the proposals by ascending order of the prices λ_j and aggregates them, forming the supply curve. Thus the intersection of this curve with the demand segment gives the market clearing price P_d and quantity Q_d, and the proposals that are dispatched, as shown in Figure 1. Therefore, the revenue for each producer i is given by:

$$\Pi_i = \sum_{j \in \{ED: s_j = i\}} \left(P_d - c_j\right) g_j$$

where g_j is the energy produced by $i = s_j$ in the ED. This profit deeply depends on the strategies of the other players, which makes the problem complex.

3. NASH EQUILIBRIA COMPUTATION

Game theory provides important tools in economics. The concept of Nash equilibrium of a game plays a relevant role in this context. Basically, it is a probability distribution over the set of strategies of each player, such that nobody wants to change unilaterally this behaviour. If some player would change his strategy with respect to a Nash equilibrium, his profit would not increase (see [2]).

In our case, the strategies of each player are the proposal prices, so, in a Nash equilibrium, we have the probability of choosing λ_j over the set $[0, b]$, where b is the maximum price at which the consumers buy electricity (see section 2 where the demand is defined). A Nash equilibrium in which each player plays with probability one a certain strategy is called a equilibrium in pure strategies.

The method that we use in this abstract only provides pure Nash equilibria, but we are currently working towards finding mixed Nash equilibria.

3.1. Adjustment Process

In current electricity markets, the producers have to communicate the market operator their proposals for each hour of the following day. We admit that each producer predicts exactly the demand for each hour and knows the technology of his competitors, so that he knows the marginal costs of the others. Our goal is to find the best strategy for each company.

We will apply an adjustment process to find out the Nash equilibria of this non-cooperative game. An adjustment process is an iterative process in which each player adjusts his strategy according to the past iterations. This is a learning process. It is easy to find examples in which this method diverges or has chaotic behaviour, so this process does not always work. However, if a solution is found, then it is a Nash equilibrium.

In this context, we started only with the prices λ_j as decision variables, but it follows immediately how to adapt the process in order to have both prices and quantities as decision variables.

We have used two adjustment processes: the ones described in [8] and in [9]. The first one only uses the data of last iteration to adjust the strategy, while the second uses an estimation based on the all past iterations. After some experiences, we noted that the first method converges faster than the second (in this case) and also it is simpler to describe. Hence, we focus in that one in this work. In [10] a very similar process is presented, but there the decision variables are the quantities.

Our method can be described with the following pseudo code:

1: initialise with an information matrix M, $\varepsilon > 0$, and demand parameters m and b;
2: let S_i be the set of proposals of the producer i and k the number of producers;
3: **repeat**
4: $X \leftarrow$ fourth column of M;
5: **for** $i = 1$ to k **do**
6: $\lambda_{S_i} \leftarrow \arg\max_{\lambda_j, j \in S_i} \sum_{s_j \in ED \cap S_i} \left(P_d - c_j\right) g_j$;
7: update the fourth column of M with λ_j for $j \in S_i$;
8: **end for**
9: $Y \leftarrow$ fourth column of M;
10: $\Delta \leftarrow ||Y - X||$;
11: **until** $\Delta < \varepsilon$

In short, in each step every producer finds the strategy that maximizes his profit assuming that the other players are going to follow the strategy of the previous iteration. The process stops when two iterations are sufficiently close to each other (that means that the current matrix M is a Nash equilibrium, because nobody made a significant change in his behaviour. In fact, when $\Delta = 0$, M is exactly a Nash Equilibrium). It is important to notice that the maximization process, in step six, needs a method able to tackle non-smooth functions, as the profit of the companies is a function with discontinuities.

The most important step in our adjustment process is the maximization of the producers's profits. To solve this problem, we have used the MATLAB implementation of a global optimization method developed by Ismael Vaz and Luís Vicente, see [11]. In this method we only need to evaluate the objective function values resulting from pattern search and particle swarm, so this is exactly what we need in our adjustment process.

3.2. Case Study

In this example, we consider one period of the pool Market, with five producers, each with one generation unit. They want to know the prices to attribute to their proposals so as to maximize their profit. We assume that this is a competitive market, so there is not cooperation between the companies.

The information matrix M is:

$$M = \begin{bmatrix} j & s_j & E_j & \lambda_j & c_j \\ 1 & \texttt{Producer A} & 100 & 0.50 & 0.40 \\ 2 & \texttt{Producer B} & 150 & 0.30 & 0.30 \\ 3 & \texttt{Producer C} & 200 & 0.80 & 0.55 \\ 4 & \texttt{Producer D} & 180 & 0.55 & 0.50 \\ 5 & \texttt{Producer E} & 250 & 0.85 & 0.60 \end{bmatrix} \quad (1)$$

and the demand is modeled by

$$y = \frac{-7}{2000} x + \frac{7}{4}.$$

This initial situation is represented in Figure 1, with $P_d = 0.50$\$ /MWh, $Q_d = 342.8571$MWh and accepted proposals:

$$M_{ED} = \begin{bmatrix} j & s_j & g_j & \lambda_j \\ 2 & \texttt{Producer B} & 150.0000 & 0.30 \\ 1 & \texttt{Producer A} & 100.0000 & 0.50 \\ 4 & \texttt{Producer D} & 92.8571 & 0.55 \end{bmatrix}$$

In our simulation of the pool market, if we have two or more proposals with the same price we accept them in a proportional way. For example, the market clearing price is 30 \$/MWh and we need 200MWh to be allocated; we have three proposals with price 30\$/Mwh and quantities 300MWh, 60MWh and 240MWh; then we accept $\frac{1}{3} = \frac{200}{300+60+240}$ of the quantity of each proposal.

Applying our algorithm to this situation, with $\varepsilon = 10^{-9}$ and a max-

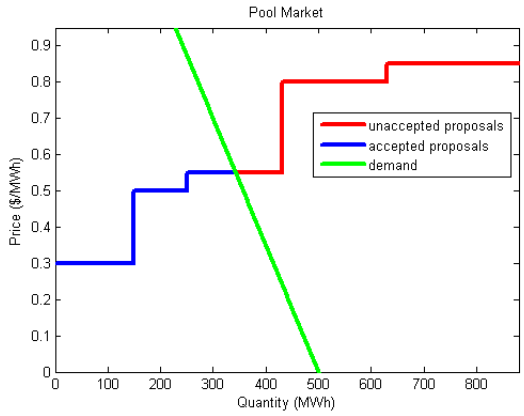


Figure 1: *Pool Market.*

imum of 50 adjustment iterations, we achieve:

$$M = \begin{bmatrix} j & s_j & E_j & \lambda_j & c_j \\ 1 & \texttt{Producer A} & 100 & 0.400000 & 0.40 \\ 2 & \texttt{Producer B} & 150 & 0.300000 & 0.30 \\ 3 & \texttt{Producer C} & 200 & 0.550000 & 0.55 \\ 4 & \texttt{Producer D} & 180 & 0.549999 & 0.50 \\ 5 & \texttt{Producer E} & 250 & 0.600000 & 0.60 \end{bmatrix}$$

in 17 iterations and final $\Delta = 2.44 \times 10^{-10}$. This is a Nash equilibrium as we can see in Figures 2, 3 and 4 :

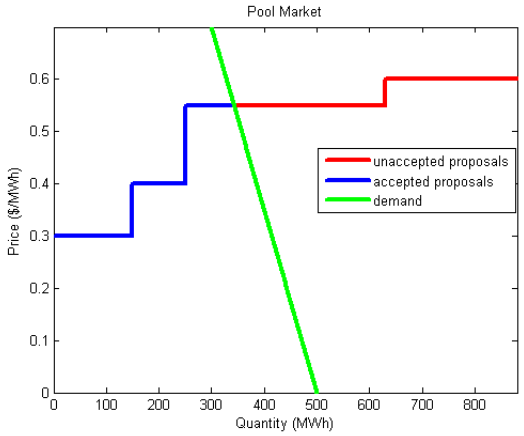


Figure 2: *Pool Market.*

Only the producer D can achieve a larger profit by increasing the price of the proposal up to 0.55 \$/MWh, but if it chooses $\lambda_1 = 0.55$ it would have to divide the sold quantity with producer C. So it chooses a price slightly less than 0.55 \$/MWh. Figure 3 shows that this process starts converging to the Nash equilibrium very fast, despite the existence of some fluctuations in Δ due to the numerical optimization process. The ED of this equilibrium is:

$$M_{ED} = \begin{bmatrix} j & s_j & g_j & \lambda_j \\ 2 & \texttt{Producer B} & 150.0000 & 0.300000 \\ 1 & \texttt{Producer A} & 100.0000 & 0.400000 \\ 4 & \texttt{Producer D} & 92.8571 & 0.549999 \end{bmatrix} \quad (2)$$

with $P_d = 0.549999$\$/MWh and $Q_d = 342.8571$MWh.

It is important to mention that the optimization method PSwarm used (see [11]) is stochastic, so applying again this method could

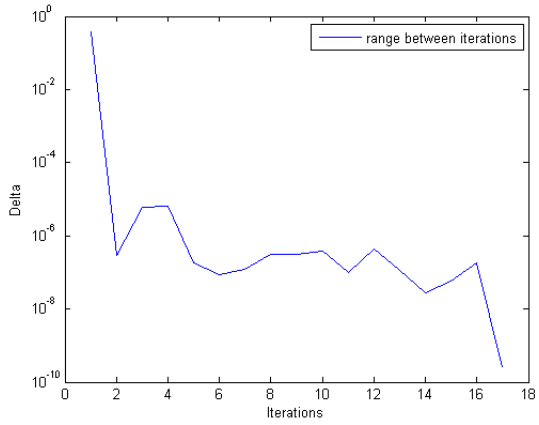


Figure 3: *Evolution of Δ through iterations of the adjustment process.*

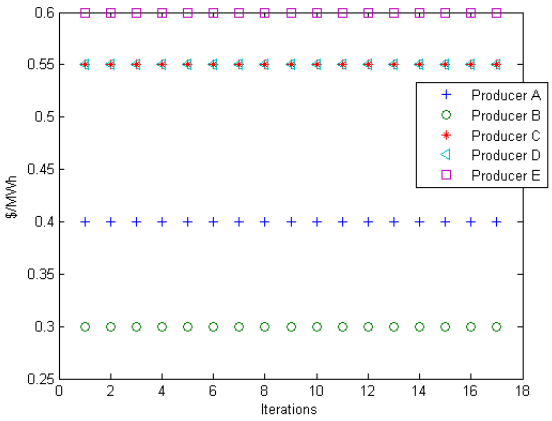


Figure 4: *Evolution of proposals's prices through iterations of the adjustment process.*

give different results. We used the adjustment process in this example several times and observed that the method always found this equilibrium, but the number of iterations used ranged between 3 and 25. Another relevant comment is that changing the initial matrix M can achieve different Nash equilibria. In this example we always found the same Nash equilibrium on pure strategies.

Note that if we modify, for example, bid 4 in Equation (2) for a quantity in the domain $[92.8571, 180]$, we keep having a Nash Equilibrium. Thus we expected that adjusting prices and quantities could lead us to chaotic behaviours and consequently to the non-convergence of the process, because usually there is more than one answer with the same profit for the optimization step.

As a matter of fact, producers can also choose the quantities for their bids, hence we applied to this example a more general adjustment process, where the quantities and prices are decision variables. To make that possible, we need to add to our data, M, m and b, the capacity of each generator. Here, it is assumed that the third column of the initial M is concerned with the maximum capacity of each generator.

Applying to the matrix of Equation (1) the adjustment process with

respect to prices and quantities we achieve with four iterations:

$$M = \begin{bmatrix} j & s_j & E_j & \lambda_j & c_j \\ 1 & \texttt{Producer A} & 100.000 & 0.400000 & 0.40 \\ 2 & \texttt{Producer B} & 150.000 & 0.300000 & 0.30 \\ 3 & \texttt{Producer C} & 200.000 & 0.550000 & 0.55 \\ 4 & \texttt{Producer D} & 92.8571 & 0.500000 & 0.50 \\ 5 & \texttt{Producer E} & 250.000 & 0.600000 & 0.60 \end{bmatrix}$$

and economic dispatch:

$$M_{ED} = \begin{bmatrix} j & s_j & E_j & \lambda_j & c_j \\ 2 & \texttt{Producer B} & 100.000 & 0.400000 & 0.40 \\ 1 & \texttt{Producer A} & 150.000 & 0.300000 & 0.30 \\ 4 & \texttt{Producer D} & 92.8571 & 0.500000 & 0.50 \\ 3 & \texttt{Producer C} & .354 \cdot 10^{-6} & 0.550000 & 0.55 \end{bmatrix}$$

with $\Delta = 0$, so there is no doubt that this represent a Nash equilibrium.

4. FUTURE WORK

The problem of non-convergence of our process when there is more than one answer with the same profit for the optimization step, can possibly be overcomed if we search for equivalence classes (two proposals are in the same class if both lead to the same profit) and use a single representant for each class. In this context, we are studying the convergence conditions that tell us when the adjustment process works successfully. Another reason that could explain the non-convergence of the method is that we may enter a cycle or an orbit that diverges. Here it may be preferable to use the adjustment process based in the estimative of past iterations, because it is more suitable in order to converge.

We are also currently studying the possibility of discretizing the space of strategies to make it finite (see [6]). In that case, there is a large number of available methods to find Nash equilibria (including mixed Nash equilibria). Then, we can adapt the results to the original game through interpolation. We have to be careful in this process because sometimes an equilibrium in discrete games is not one in the original game. An important step in this process is to recognize strategies that are dominated, meaning that they are never adopted and played.

In conclusion, this work addresses important questions arising in the pool market, and can contribute to the development of algorithms to find mixed Nash equilibria where the sets of strategies are continuous.

5. ACKNOWLEDGEMENTS

This work was supported by a INESC Porto fellowship in the setting of the Optimization Interunit Line.

6. REFERENCES

[1] J. Saraiva, J. P. da Silva, and M. P. de Leão, *Mercados de Electricidade - Regulação de Tarifação de Uso das Redes*. FEUP Edições, 2002.

[2] D. Fudenberg and J. Tirole, *Game Theory*, 5th ed. Cambridge, MA: MIT Press, 1996.

[3] D. Pozo, J. Contreras, Ángel Caballero, and A. de Andrés, "Long-term Nash equilibria in electricity markets," *Electric Power Systems Research*, vol. In Press, Corrected Proof, pp. –, 2010. [Online]. Available: http://www.sciencedirect.com/science/article/B6V30-519VV21-1/2/e7fcc2861b27be46806cd9aaf0aed724

[4] M. Pereira, S. Granville, M. Fampa, R. Dix, and L. Barroso, "Strategic bidding under uncertainty: a binary expansion approach," *Power Systems, IEEE Transactions on*, vol. 20, no. 1, pp. 180 – 188, Feb. 2005.

[5] D. Pozo and J. Contreras, "Finding multiple nash equilibria in pool-based markets: A stochastic EPEC approach," *Power Systems, IEEE Transactions on*, vol. PP, no. 99, pp. 1 –9, 2011.

[6] K.-H. Lee and R. Baldick, "Tuning of discretization in bimatrix game approach to power system market analysis," *Power Systems, IEEE Transactions on*, vol. 18, no. 2, pp. 830 – 836, May 2003.

[7] E. Hasan and F. Galiana, "Fast computation of pure strategy nash equilibria in electricity markets cleared by merit order," *Power Systems, IEEE Transactions on*, vol. 25, no. 2, pp. 722 –728, May 2010.

[8] J. P. Pedroso and Y. Smeers, "Equilibria on a game with discrete variables," in *Programs, Proofs, Processes*, F. Ferreira, H. Guerra, E. Mayordomo, and J. Rasga, Eds. Azores, Portugal: Computability in Europe 2010, 2010, pp. 326–335.

[9] L. Thorlund-Petersen, "Iterative computation of Cournot equilibrium," Norwegian School of Economics and Business Administration-, Working Papers, 1988. [Online]. Available: http://econpapers.repec.org/RePEc:fth:norgee:1-88

[10] J. Contreras, M. Klusch, and J. Krawczyk, "Numerical solutions to Nash-Cournot equilibria in coupled constraint electricity markets," *Power Systems, IEEE Transactions on*, vol. 19, no. 1, pp. 195 – 206, Feb. 2004.

[11] A. Vaz and L. Vicente, "A particle swarm pattern search method for bound constrained global optimization," *Journal of Global Optimization*, vol. 39, pp. 197–219, 2007, 10.1007/s10898-007-9133-5. [Online]. Available: http://dx.doi.org/10.1007/s10898-007-9133-5

Application of Combinatorial Optimization in Natural Gas System Operation

Teresa Nogueira *

* Institute of Engineering, Polytechnic Institute of Porto
Rua Dr. Antonio Benardino de Almeida, 431 – 4200-072 Porto, PORTUGAL
tan@isep.ipp.pt

ABSTRACT

The best places to locate the Gas Supply Units on natural gas systems and their optimal allocation to loads are the key factors to organize an efficient upstream gas infrastructure. In this work we use the *P*-median problem to locate the GSUs on a gas network and the transportation problem to assign gas demand nodes to the source facilities. Due to its mathematical structure, the application of *P*-median problem to large networks needs heuristic techniques. This paper presents two Lagrangean heuristics, tested on a realistic network - the primary Iberian natural gas network. Computational results are presented, showing the location arrangement and system total costs.

Keywords: Gas supply units – GSUs, Lagrangean heuristic, P-median problem, Relocation heuristic

1. INTRODUCTION

To comply with natural gas demand growth patterns and Europe import dependency, the Iberian natural gas industry needs to organize an efficient upstream infrastructure [1]. Marine terminals, storage facilities and gas injection points, are the source points of the natural gas system: the Gas Supply Units – GSUs. The location of such infrastructures in gas networks, as well as allocated loads, should be carefully planned in order to minimize overall costs [2].

Most of gas loads are connected to GSUs by pipelines, being the natural gas transported in the gas form at high pressure . Alternatively, when there is no physical pipeline between supply/demand points, gas may be transported by virtual pipeline – gas transported by road trucks in its liquefied form.

The aim of this paper is the presentation of two Lagrangean heuristics to support the decision of GSUs location on a gas network. This location problem studies the best places to locate GSUs on network, minimizing total distances between sources and loads [3].

Once defined, GSUs serve load sites with known gas demands, minimizing combined GSUs location and transport costs. This question is addressed by the transportation problem.

For the location problem, we use the *P*-median problem, that finds the location of a number of *P* facilities (in this case, GSUs), so as to minimize the weighted average distance of the system [4]. Due to its mathematical structure, the *P*-median problem is NP-hard and therefore cannot be solved in polynomial time. So, it is necessary to use heuristics methods for large and realistic *P*-median problems.

In [5], was presented a simple Lagrangean heuristic, by using Lagrangean relaxation and subgradient optimization to solve the dual problem. In this paper we improve the solution by adding the Lagrangean relocation heuristic. This is done by analyzing some changes between medians and non-medians locations. With this exhaustive procedure, we can obtain better solutions, not reached by simple Lagrangean heuristics.

In section two we present the Lagrangean relaxation for *P*-median problems. Section three presents the relocation heuristic, an improvement to the simple Lagrangean heuristic. To conclude about the effectiveness of the Lagrangean relocation heuristic, we compare its computational results to those of the simple Lagrangean approach.

The location modelling presented in this work is applied to the Iberian natural gas system, to find the best GSUs location and their optimal allocation to loads.

The Iberian natural gas network is geographically organised with 65 demand nodes (Fig. 1). Most of these demand points are connected by physical pipelines (red lines in Fig. 1); the others are supplied by road trucks with gas in liquefied form – the virtual pipelines. These virtual pipelines are all the connections between two nodes without a physical pipeline.

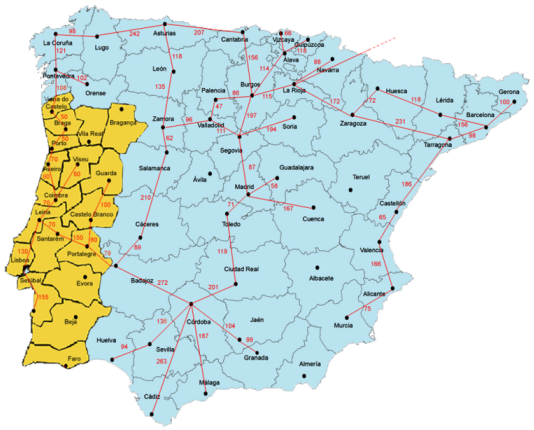


Figure 1: *Iberian natural gas network.*

2. THE LAGRANGEAN APPROACH

To exemplify the application of Lagrangean relaxation to the location *P*-median problem, we will consider the following binary integer programming problem, with *m* GSUs potential sites and *n* demand nodes:

$$Z = \text{Min} \sum_{i=1}^{m} \sum_{j=1}^{n} \alpha . d_{ij} . X_{ij} \qquad (1)$$

subject to:

GSUs Located	Values α/α_v (e/m^3km)	Total Costs (M€)	
		Simple Lagrangean Heuristic	Lagrangean Relocation Heuristic
P = 14	0,015 / 0,018	286.690,7	281.782,2
P = 20	0,019 / 0,022	318.110,3	310.359,3
P = 25	0,024 / 0,027	365.457,4	355.393,9
P = 28	0,027 / 0,030	382.454,2	372.758,9

Table 1: Lagrangean Heuristics Results.

$$\sum_{i=1}^{m} X_{ij} = 1 \qquad j = 1,\ldots,n \tag{2}$$

$$\sum_{i=1}^{m} X_{ii} = P \tag{3}$$

$$X_{ij} \leq X_{ii} \qquad i = 1,..,m; j = 1,..,n \tag{4}$$

$$X_{ij} \in \{0,1\} \qquad i = 1,..,m; j = 1,..,n \tag{5}$$

Where:

$\alpha.[d_{ij}]$, is the symmetric cost [distance] matrix, with $d_{ii} = 0$, $\forall i$; α is the kilometric gas transported cost per gas unit (cubic meter – m^3); $[X_{ij}]$ is the allocation matrix, with $X_{ij} = 1$ if a node i is allocated to node j, and $X_{ij} = 0$, otherwise; $X_{ii} = 1$ if node i has a GSU and $X_{ii} = 0$, otherwise; P is the number of GSUs (medians) to be located.

The objective function (1) minimizes the distance of each pair of nodes in network, weighted by α. Constraints (2) ensure that each node j is allocated to a source node i. Constraint (3) determines the number of GSUs to be located (P). Constraint (4) sets that a demand node j is allocated to a node i, if there is a GSU at node i. Constraint (5) states the integer conditions.

The parameter α assumes the cost value in physical pipelines, if a pipe exists between node i and j. If there is no pipe connection between nodes, the parameter α is replaced by α_v, the cost value in virtual pipelines (usually, α_v is greater than α). These different transport cost values are implicitly assumed in the algorithm and have a great influence on the located GSUs solution.

3. RELOCATION HEURISTIC

The Lagrangean heuristic presented to solve the location problem often gives very good results, but it can be improved with the application of an additional heuristic – the relocation heuristic – which attempts to get closer to the optimal solution than the simple Lagrangean heuristic. The computational results comparing the two Lagrangean approach are presented in this section.

The relocation heuristic starts from the simple Lagrangean results. Then, the P clusters are identified, $C_1, C_2, \ldots C_P$, corresponding to the P medians (GSUs) and their allocated non-medians (gas load nodes). The solution can be improved by searching for new medians, swapping the current medians by non-medians and reallocating the loads. For each swap we analyze the solution achieved by new location and allocation from new source to loads. If the new solution is better, we keep it. The process is repeated until no more improvements are achieved.

In table 1 we can see the behavior of the two implemented Lagrangean heuristics: simple and with relocation. The solution was taken for different values of α and α_v, respectively, kilometric cost of the natural gas unit transported by physical and virtual pipeline. The total costs presented in table I are the sum of GSUs implantation costs and the transport costs.

The increment of P means an increment of medians, so more GSUs installed, thus, the total costs increase for both approaches. As observed in table 1, for each case of $\alpha/\ \alpha_v$ values, the total costs that resulted from relocation heuristic are better (lower costs) than the simple Lagrangean heuristic.

As we increase the α/α_v value transportation costs are increased, so the optimization problem minimizes the solution by adding more GSUs. This is an attempt to minimize transportation cost, but fixed costs have a big influence, so, the result is a higher total cost.

For all solutions of P, we can see in table I the Lagrangean relocation heuristic presents an obvious improvement in system total costs.

4. CONCLUSION

Based on Lagrangean heuristics, this work supports GSUs location decision-make and their optimal allocation to gas demands. The simple Lagrangean heuristic often gives very good results, but not necessarily the optimal ones. To improve this resolution approach, we developed the Lagrangean relocation heuristic, which proved its efficiency in total costs function minimization. Its performance was verified for different location scenarios and with different parameters values.

The developed location model can be applied to any other gas network type with the same good performance.

5. REFERENCES

[1] T. Nogueira, Z. Vale and M. Cordeiro. Natural Gas Market in Europe: Development and Trends. *CAIP2005 – 7th Interamerican Conference on Computer Analysis of Manufacturer Processes*, Vila Real, Portugal, 2005, pp. 115–118.

[2] T. Nogueira, R. Mendes, Z. Vale and J. Cardoso. An Heuristic Approach for Optimal Location of Gas Supply Units in Transportation System. *22nd International Scientific Meeting of Gas Experts*, vol.1, pp. 303-311, May 2007, Opatija, Croatia.

[3] L. Lorena and M. Narciso. Relaxation Heuristics for Assignment Problem. *European Journal of Operational Research*, 91:600–610, 1996.

[4] Z. Drezner and H. Hamacher. Faciility Location: applications and Theory. *1st ed. New York: Springer-Verlag*, 2004, p. 119-143.

[5] T. Nogueira and Z. Vale. Natural Gas System Operation: Lagrangean Optimization Techniques. *IGRC 2008 – International Gas Union Research Conference*, ID 147, Category: Transmission, October 2008, Paris, France.

A Multi-objective EPSO for Distributed energy resources planning

Renan S. Maciel [‡] Mauro Rosa [*] Valdimiro Mirando [*] [†] Antonio Padilha-Feltrin [‡]

[*] INESC Porto
Campus da FEUP, Rua Dr. Roberto Frias 378, Porto, Portugal
{mrosa, vmiranda}@inescporto.pt

[‡] Department of Electrical Engineering, Sao Paulo State University (UNESP)
Ilha Solteira, Brazil
rsmaciel@aluno.feis.unesp.br, padilha@dee.feis.unesp.br

[†] FEUP, Faculty of Engineering of the University of Porto
Porto, Portugal

ABSTRACT

There is an increasing interest in Multi-objective optimization meta-heuristics to solve complex problems in many different areas. In Power Systems, Multi-objective optimization is also under intensive research applied to traditional problems and mainly to the most recent trends such as Distributed Energy Resources integration considering SmartGrids paradigm. Therefore, this paper is proposing a Multi-objective approach to the hybrid EPSO method. The Multi-objective EPSO method, called MEPSO, is applied to a discrete problem of DER impact evaluation on electric distribution network. It was observed, through the several runs, a better performance of MEPSO when compared to the NSGA-II method. Despite of being an initial evaluation, the results encourage to exploit the best of EPSO characteristics in the Multi-objective domain.

Keywords: Multi-objective optimization, Meta-heuristics, EPSO, NSGA-II, DER planning

1. INTRODUCTION

The ever-increasing interest to apply the concepts of Multi-objective optimization (MO) in real-world problems, and the intense exploitation of the meta-heuristics as computational solutions to cope with complex optimization problems are fostering the development of many well known search techniques to the multi-objective domain. In power systems planning area, methodologies based on MO meta-heuristics are under research, especially in the Distributed Energy Resources (DER) integration [1]. In general, in the earlier works, it was used the so-called *classic* MO methods such as Weighted Sum or ε-Constraint based on Genetic Algorithms (GA). As they have presented some limitations [2], new proposals based on Pareto optimality concepts, have being constantly employed. Despite the GA-based methods, such as NSGA-II and SPEA2, being more often used on DER planning, there are different MO techniques based on meta-heuristics (e.g. Simulated Annealing, Tabu Search, PSO [3], [4], and so on) that take advantage of some specific mechanisms of each meta-heuristic.

The Evolutionary Particle Swarm Optimization (EPSO) successfully combines evolutionary strategies with the PSO method. A complete view about performance improvements carried out on EPSO algorithm is reported in the literature, including on power system problems [5], [6].

This work proposes a Multi-objective EPSO called MEPSO, which is applied to a discrete problem of DER integration in electrical distribution networks. First of all, it is thoroughly presented and discussed the MEPSO algorithm. After that, an example followed by some remarks, results and discussions is showed.

2. THE MULTI-OBJECTIVE EPSO PROPOSAL: MEPSO

The EPSO method merges the efficient PSO movement equation and overall structure with evolutionary strategies, namely self-adaptive mutation and an explicit selection procedure [7]. For the MEPSO approach some steps of the EPSO algorithm are preserved, whereas others are strongly changed in order to incorporate MO concepts. Considering the general algorithm of EPSO presented in [8], the mutation and replication procedures were fully preserved in the MEPSO. On the other hand, reproduction, evaluation, and selection steps were remodeled using some of the MO procedures introduced by the NSGA-II method [9]. The general algorithm for MEPSO can be described as follow:

Parameters and variables initialization;
Do *i = 0;*
Do *while i < IterMax;*
  Rank and sort *the swarm based on the concept of dominance;*
  Update *the Pareto List (PL), an external list that keeps the optimal solution set;*
  Assign *the Global Best (Gb) to each particle of the swarm;*
  Do *for each particle:*
    Replicate *the particle;*
    Execute *on the replicated particle:*
      Mutation of the strategic parameters;
    Execute on both original and replicated particles:
      Reproduction (based on PSO movement equation);
      Assign *the Personal best (Pb);*
    Add *the replicated particle to Replica List (RL) that keeps whole set of replicated particles;*
  Combine *the original swarm with the RL;*
  Perform *Selection over the combined list of particles;*
  Do *i = i + 1;*
Print PL.

The *rank* and *sort* of the swarm using the concept of dominance can be performed in different ways. In this paper, the Fast Non-dominated Sort (FNS) algorithm [9] was employed. The non-dominated solutions are the best ranked, belonging to the Pareto front 1. They are followed by the dominated solutions of only one solution in the front 2. The process follows in the same way until the last front.

The PL update consists of the merger between current PL with the front one. Thus, the elimination of the repeated and dominated solutions in the combined set is performed in order to conclude the update process. The Gb assignment in MEPSO was deeply remodeled. The swarm will not have the same Gb assigned for each particle, as occurs with the Star Communication approach [8]. Except for the front 1, where a particle belonging to the front f will receive, as Gb, a solution randomly chosen from the front $f-1$. The Gb for the front 1 are randomly taken from a reduced set of the PL, called Gb List. The solutions for Gb List are chosen aiming to favor diversity, i.e., looking for better exploration of poorly crowded regions of the search space. Hence, the crowding distance metric [9] was used in order to introduce a measure of dispersion of the solutions, into objective function space. The Gb List may sized fixed. In this paper, the five less crowded solutions in PL was chosen. However, another alternative can be a percentage of the PL size [10].

After the particle performs a movement, Pb is assigned. The Pb of a particle is the last non-dominated position visited by the particle itself in its path, until the current iteration.

The selection stage doesn't consist in a simple comparison among the fitness function of a particle and its replicas. The original swarm and the list of replicas are combined and ranked using FNS. Then, it is applied an elitist strategy [9], wherein the swarm of the next iteration is firstly composed by the best ranked solutions accordingly with dominance and afterward using the diversity criterion based on the crowding distance metric.

2.1. Discussion

There are many features in this MEPSO approach that can be changed and tested towards performance improvements.

To compare solutions in MO problems isn't quite trivial like in single objective optimization. Several mechanisms should be defined in order to deal with multiple objectives, to accommodate Pareto optimality concepts, and to obtain a diversified Pareto front solution set. Here it was utilized the dominance-based rank and the crowding distance metric from [9], which corresponds to NSGA-II method improvements over its former version. Thus, other strategies may be matter of investigation.

The Gb assignment structure has huge influence on convergence. In this approach the main idea is to stimulate diversity and exploration of the search space. For front 1 it is intended to intensify the search along the PF. However, changes may be performed on how the Gb is chosen in a front. In this work it is a randomly procedure, or even the whole assignment procedure.

The selection stage has also a high importance. The current proposal exploits elitist procedure from NSGA-II. Nevertheless, sometimes particle's information may be lost since a particle and its replica can be chosen. Tests can be made in order to check the influence of this behavior on the method performance.

In [10] there is an example of MO PSO based on the NSGA-II mechanisms. However, it is different from this latter approach, because it doesn't incorporating the ES of EPSO, in the personal and global best assignment and in elitism.

3. DER INTEGRATION PLANNING PROBLEM

It is widely recognized and reported on literature that high penetration of DER on distribution networks may offers together benefits and negative consequences [11] to the systems. Both positive and negative impacts depend on many technical characteristics such as technology used, size of units, operation and control strategies to deal with DER as well as capacity and placement on network. MO optimization stands for an interesting way to cope with DER integration problem, mainly due to suitable property of combine objectives from different natures over a discrete manner.

In order to show the potentialities of the proposed method, it is performed a simple set of tests involving MEPSO and NSGA-II methods. A simplified model of the problem is assumed. It consists of studying the impact over the network losses and short circuit level accordingly with the position and size of generation units. A fixed number of generators to be connected are defined. Also a single generation and load scenario is used in order to prepare the problem in a discrete manner with a finite number of solutions. The methods codification is detailed in [12] and an example is shown in Fig. 1. Each vector position indicates an available Distributed Generation (DG) unit and the value assumed in each position indicates the node where the generator is connected.

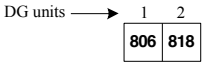


Figure 1: *Codification example of the problem.*

In Fig. 1, for instance, the DG unit "1" is connected on node "806", and the generator unit "2" on node "818".

3.1. Problem formulation

The two objectives to be minimized are real power loss and short circuit level. They could be viewed through the use of some indices, where the tradeoff between both is observed. The indices ILp and ISC3 are written as follows:

1. Total Real Power Losses index (ILp) [13]: in (1) it is evaluated the DG impact over the real power losses by calculating the ratio between the network total real power loss for a DG configuration (Loss^{DG}) and the total real power loss without DG (Loss^{0}).

$$ILp = \frac{Loss^{DG}}{Loss^{0}} \tag{1}$$

1. Three-Phase Short-Circuit Level index (ISC3) [13]: this index, defined in (2), contributes to the DG impacts evaluation concerning the network fault protection strategies.

$$ISC3 = \max_{i=1,NN} \left(\frac{I_{SC}abc_i^{DG}}{I_{SC}abc_i^{0}} \right) \tag{2}$$

where: $I_{SC}abc_i^{DG}$ represents three-phase fault current value in node i for a given DG configuration on the network; $I_{SC}abc_i^{0}$ represents three-phase fault current value in node i for the network without DG; NN is the number of nodes.

Both indices follow the distribution utilities requirements in terms of DG unit connections. In some cases, losses and short-circuit are the most important variables for connecting DG.

The problem formulation is presented in equations (3) to (8).

$$Min\ ILp \tag{3}$$

$$Min\ ISC3 \tag{4}$$

Subject to:

$$0.95 \cdot V_{S/S} \leq \left| V_i^{DG} \right| \leq 1.05 \cdot V_{S/S} \quad (5)$$

$$\left| I_j^{DG} \right| \leq I_j^{\max} \quad (6)$$

$$n_i^{DG} \leq 1 \quad (7)$$

$$N_{network}^{DG} = N_{available}^{DG} \quad (8)$$

where $V_{S/S}$ is rated voltage at the Substation; V_i^{DG} is the voltage at the node i for a given DG configuration; I_j^{DG} is the current through the branch j for a given DG configuration; I_j^{max} is the maximum rated current for branch j; n_i^{DG} is the number of DG units connected in node i; $N_{network}^{DG}$ is the total number of DG units connected in the network; and $N_{available}^{DG}$ is the total number of DG units available.

This formulation guides the distribution companies that own DG units, allowed in some places as a way to invest in the network [14], to take advantage of the connection by means of a tradeoff analysis. The study may also represent a scenario of DER not owned by the utility where the information provided give a portrait of the impact over the network technical performance, then driving a policy of incentive or not the DER connection on certain locations.

Sometimes even the solutions that violate constraints may have relevance in the tradeoff analysis if the gain in some objective justifies the investment on turning feasible solutions, depending on the extent of the violation and the violated constraint. For this reason and in order to observe the performance of the methods for a larger number of PF, the tests will be performed for both constrained (CONS) and unconstrained (UNCONS) problems.

Two radial electric distribution networks with different features are used: the IEEE-34 and IEEE-123 [15]. The DG units to be allocated in the networks were defined in such a way to produce a similar penetration level in both grids. Two generators were chosen to each network: one of rated power of 200 kW and another of 400 kW for the IEEE-123 network; and one of rated power of 100 kW and another of 200 kW for the IEEE-34 network. The information about each network and the search space is shown in Table 1. The substation node is not a candidate node to receive a generator. It is also presented in the last Table 1 column how much MaxEvals represents related to the search space size.

Network	Vs/s (pu)	Nodes (except S/S)	DG units	Solutions in the search space	Max Evals	Max Evals (%)
IEEE-123	1.0	113	2	12656	5000	39.5
IEEE-34	1.05	32	2	992	400	40.3

Table 1: Summary of the networks, tests and search space features.

Finally, the methods performance is compared here by the number of points found in the calculated PF (PF_{calc}) that belongs to the true PF (PF_{true}). For this purpose it is observed the cardinality of the PF_{calc}, the number of dominated solutions (DS) in this set, and also the metric PF ratio (PFR) is used, defined by (4), which gives the percentage of PF_{true} found.

$$PFR = \frac{|PF_{calc} \cap PF_{true}|}{|PF_{true}|} \times 100 \quad (9)$$

4. RESULTS

Table 2 shows the cardinality of the true PF for constrained and unconstrained problems considering both networks.

	IEEE123		IEEE34	
	CONS	UNCONS	CONS	UNCONS
$\lvert PF_{true} \rvert$	29	91	52	76

Table 2: Number of Pareto Front points for different cases.

The results are presented in Tables 3 and 4 for both electric networks considering constrained and unconstrained problem.

	CONS34		UNCONS34	
	NSGA-II	MEPSO	NSGA-II	MEPSO
$\lvert PF_{calc} \rvert$	50	50	67	70
DS	-	-	-	-
PFR	96.2	96.2	88.2	92.1

Table 3: Summary of results for the CONS34 and UNCONS34 test cases.

	CONS123		UNCONS123	
	NSGA-II	MEPSO	NSGA-II	MEPSO
$\lvert PF_{calc} \rvert$	26	28	85	91
DS	-	-	-	-
PFR	89.6	96.6	93.4	100.0

Table 4: Summary of results for the CONS123 and UNCONS123 test cases.

In all test cases the final solution obtained by each method does not have points dominated by the PF_{true}. Then, both methods demonstrated good convergence to the true PF. However, they generally were not able to define the whole PF_{true} set.

Comparing the methods performance, except for CONS 34 test case where both methods had the same PFR, MEPSO found more solutions of the true PF than NSGA-II, keeping always a PFR higher than 90%. Additionally, only MEPSO found the whole true PF for the UNCONS123 test case.

It is important to remark that although MEPSO presented PFR equal or higher than NSGA-II, it does not means that NSGA-II solution set is contained in the MEPSO solution set, as can be seen in the Fig. 2.

The EPSO method presents performance improvements and features that can be exploited in MO. This paper shows in details a multi-objective proposal for EPSO and an example of application in Power System research field, where MO is being increasingly used. The results demonstrate that MEPSO is comparable or even better than NSGA-II, a method largely employed in the proposed problem. MEPSO also preserved a simple framework and a user-friendly parameter setting.

However, MEPSO must be applied to problems with different sizes and features in order to clearly define its behavior.

5. ACKNOWLEDGEMENTS

This work was supported by the FAPESP (grant no. 2006/06758-9), CNPq (grant no. 303741/2009-0) and CAPES (grant no. 0694/09-6).

6. REFERENCES

[1] A. Alarcon-Rodriguez, G. Ault, and S. Galloway. Multi-objective planning of distributed energy resources: a review

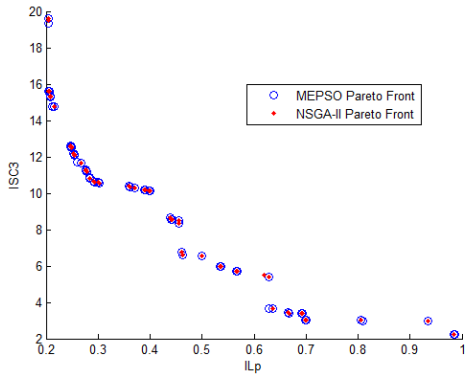


Figure 2: *Pareto Front for MEPSO and NSGA-II methods considering the UNCONS34 test case.*

of the state-of-the-art. *Renewable and sustainable energy reviews*, vol. 14, pp. 1353-1366, 2010.

[2] K. Deb. Multi-objective optimization using evolutionary algorithms. *U.K.: John Wiley & Sons Ltd*, 2004.

[3] M. Reyes-Sierra, and C.A. Coello Coello. Multi-objective particle swarm optimizers: a survey of the state-of-the-art. *International journal of computational intelligence research*, vol. 2, n. 3, pp. 287-308, 2006.

[4] C.A. Coello Coello, G.B. Lamont, D.A.V. Veldhuizen. Evolutionary algorithms for solving multi-objetive problem. *2nd ed., New York: Springer*, 2007.

[5] V. Miranda and N. Fonseca. EPSO – best-of-two-world metaheuristic applied to power systems problems. in *Proc. 2002 IEEE congress on evolutionary computation (CEC)*, pp. 1847-1851.

[6] M. Eghbal, E.E. El-Araby, N. Yorino, and Y. Zoka. Application of metaheuristic methods to reactive power planning: a comparative study for GA, PSO and EPSO. in *Proc. 2007 ISIC – IEEE International conference on systems, man and cybernetics*, pp. 3755-3760.

[7] V. Miranda. Hybrid systems. in *Modern heuristic optimization techniques*, *K. Y. Lee, M. A. El-Sharkawi, Ed. New Jersey: John Wiley & Sons*, 2008, pp. 524-562.

[8] V. Miranda, H. Keko and A.J. Duque. Stochastic star communication topology in evolutionary particle swarm (EPSO). *International journal of computational intelligence research*, vol. 4, n. 2, pp. 105-116, 2008.

[9] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. on Evolutionary Computation*, vol. 6, n. 2, pp. 182-197, 2002.

[10] X. Li. A non-dominated sorting particle swarm optimizer for multiobjective optimization. in *Proc. 2003 Genetic and evolutionary computation conference (GECCO'2003)*, pp. 37-48.

[11] J.A. Peas-Lopes, N. Hatziargyriou, J. Mutale, P. Djapic and N. Jenkins. Integrating distributed generation into electric power systems: a review of drivers, challenges and opportunities. *Electric power systems research*, vol. 77, n. 9, pp. 1189-1203, 2007.

[12] R.S. Maciel and A. Padilha-Feltrin. Distributed generation impact evaluation using a multi-objective Tabu Search. in *Proc. 2009 International conference on intelligent systems applications to power systems*, pp. 37-48.

[13] L.F. Ochoa, A. Padilha-Feltrin, and G. Harrison. Time-series-based maximization of distributed wind power generation integration. *IEEE Trans. on Energy Conversion*, vol. 23, n. 3, pp. 968-974, 2008.

[14] P. Siano, L.F. Ochoa, G.P. Harrison and A. Piccolo. Assessing the strategic benefits of distributed generation ownership for DNOs. *IET Gen., Trans. & Distr.*, vol. 3, n. 3, pp. 225-236, 2009.

[15] W.H. Kersting. Radial distribution test feeders. in *Proc. 2001 IEEE Power Engineering Society Winter Meeting,* vol. 2, pp. 908-912.

On Using Preprocessing Cuts: Identification and Probing Schemes in Stochastic Mixed 0-1 and Combinatorial Optimization

L.F. Escudero * M.A. Garín † M. Merino ‡ G. Pérez ‡

* Dpto. Estadística e Investigación Operativa
Universidad Rey Juan Carlos, Móstoles (Madrid), Spain
laureano.escudero@urjc.es

† Dpto. de Economía Aplicada III
Universidad del País Vasco, Bilbao (Vizcaya), Spain
mariaaraceli.garin@ehu.es

‡ Dpto. de Matemática Aplicada, Estadística e Investigación Operativa
Universidad del País Vasco, Leioa (Vizcaya), Spain
{maria.merino, gloria.perez}@ehu.es

ABSTRACT

We present a Branch and Fix Coordination algorithm for solving medium and large scale multi-stage mixed 0-1 & combinatorial optimization problems under uncertainty. The uncertainty is represented via a nonsymmetric scenario tree. The basic idea consists of explicitly rewriting the *nonanticipativity constraints (NAC)* of the 0-1 and continuous variables in the stages with common information. As a result an assignment of the constraint matrix blocks into independent scenario cluster submodels is performed by a compact representation. This partitioning allows to generate a new information structure to express the *NAC* which link the related clusters, such that the explicit *NAC* linking the submodels together is performed by a splitting variable representation The new algorithm has been implemented in a C++ experimental code that uses the open source optimization engine *COIN-OR*, for solving the auxiliary *LP* and mixed 0-1 submodels. Some computational experience is reported to validate the new proposed approach. We give computational evidence of the model tightening effect that have preprocessing techniques in stochastic integer optimization as well, by using the probing and Gomory and clique cuts identification and appending schemes of the optimization engine of choice.

Keywords: Integer Programming, Mathematical Programming, Stochastic integer optimization

1. INTRODUCTION

Stochastic Optimization is actually one of the most robust tools for decision making. It is broadly used in real-world applications in a wide range of problems from different areas such as finance, scheduling, production planning, industrial engineering, capacity allocation, energy, air traffic, logistics, etc. The integer problems under uncertainty have been studied in [1], [2] and [3], just for citing a few references. An extended bibliography of Stochastic Integer Programming (SIP) has been collected in [4].

It is well known that a mixed 0-1 & combinatorial optimization problem under uncertainty with a finite number of possible future scenarios has a mixed 0-1 *Deterministic Equivalent Model (DEM)*, where the risk of providing a wrong solution is included in the model via a set of representative scenarios. However, as any graph representation of this type of multi-stage models can suggest, the scenario information structuring for this type of problems is more complex than for the approximation made by considering two-stage stochastic mixed 0-1 & combinatorial models. We should point out that the scenario tree in real-life problems is very frequently a nonsymmetric one and then, the traditional splitting variable representation for *the nonanticipativity constraints* (for short, *NAC*), see [1, 5], on the 0-1 and continuous variables does not appear readily accessible to manipulations that are required by the decomposition strategies. A new type of strategies is necessary for solving medium and large scale instances of the problem. The decomposition approaches that appear most promising are based on some forms of branching selection, and scenario cluster partitioning and bounding that definitively use the information about the separability of the problem, see our work in [6].

In full version of this work [7] we present a stochastic mixed 0-1 optimization modeling approach and a parallelizable *Branch and Fix Coordination (BFC)* algorithm for solving general mixed 0-1 & combinatorial optimization problems under uncertainty, where it is represented by nonsymmetric scenario trees. Given the structuring of the scenario clusters, the approach generates independent cluster submodels, then, allowing parallel computation for obtaining lower bounds to the optimal solution value as well as feasible solutions for the problem until getting the optimal one. We present a splitting variable representation with explicit *NAC* for linking the submodels together, and a compact representation for each submodel to treat the implicit *NAC* related to each of the scenario clusters. Then, the algorithm that we propose uses the *Twin Node Family (TNF)* concept, see [6], and it is specially designed for coordinating and reinforcing the branching nodes and the branching 0-1 variable selection strategies at each *Branch-and-Fix (BF)* tree. The nonsymmetric scenario tree which will be partitioned into smaller scenario cluster subtrees. The new proposal is denoted *Nonsymmetric BFC-MS* algorithm. We report some computational experience to validate the new approach by using a testbed of medium and large scale instances.

2. SPLITTING VARIABLE REPRESENTATION IN STOCHASTIC OPTIMIZATION

Let us consider the following multi-stage deterministic mixed 0-1 model

$$
\begin{aligned}
&\min \sum_{t\in\mathscr{T}} a_t x_t + c_t y_t \\
&s.t. \\
&\quad A_1 x_1 + B_1 y_1 = b_1 \\
&\quad A_t' x_{t-1} + A_t x_t + B_t' y_{t-1} + B_t y_t = b_t \quad \forall t \in \mathscr{T} - \{1\} \\
&\quad x_t \in \{0,1\}^{nx_t},\ y_t \in \mathscr{R}^{+ny_t}, \quad \forall t \in \mathscr{T}
\end{aligned} \tag{1}
$$

where $\mathscr{T}$ is the set of stages (without loss of generality, let us consider that a stage is only included by one time period), such that $T = |\mathscr{T}|$, x_t and y_t are the nx_t and ny_t dimensional vectors of the 0-1 and continuous variables, respectively, a_t and c_t are the vectors of the objective function coefficients, and A_t and B_t are the constraint matrices for stage t.

This model can be extended to consider uncertainty in some of the main parameters, in our case, the objective function, the *rhs* and the constraint matrix coefficients. To introduce the uncertainty in the parameters, we will use a scenario analysis approach. A scenario consists of a realization of all random variables in all stages, that is, a path through the scenario tree. In this sense, Ω will denote the set of scenarios, $\omega \in \Omega$ will represent a specific scenario, and w^ω will denote the likelihood or probability assigned by the modeler to scenario ω, such that $\sum_{\omega\in\Omega} w^\omega = 1$. We say that two scenarios belong to the same group in a given stage provided that they have the same realizations of the uncertain parameters up to the stage. Following the *nonanticipativity principle*, see [1, 5], among others, both scenarios should have the same value for the related variables with the time index up to the given stage. Let also $\mathscr{G}$ denote the set of scenario groups (i.e., nodes in the underlying scenario tree), and $\mathscr{G}_t$ denote the subset of scenario groups that belong to stage $t \in \mathscr{T}$, such that $\mathscr{G} = \cup_{t\in\mathscr{T}} \mathscr{G}_t$. Ω_g denotes the set of scenarios in group g, for $g \in \mathscr{G}$.

The *splitting variable* representation of the *DEM* of the full recourse stochastic version related to the multi-stage deterministic problem (1) can be expressed as follows,

$$
\begin{aligned}
&z_{MIP} = \min \sum_{\omega\in\Omega} \sum_{t\in\mathscr{T}} w^\omega \left(a_t^\omega x_t^\omega + c_t^\omega y_t^\omega\right) \\
&s.t. \\
&\quad A_1 x_1^\omega + B_1 y_1^\omega = b_1 \ \ \forall \omega \in \Omega \\
&\quad A_t'^\omega x_{t-1}^\omega + A_t^\omega x_t^\omega + B_t'^\omega y_{t-1}^\omega + B_t^\omega y_t^\omega = b_t^\omega,\ \forall \omega \in \Omega,\ t \geq 2 \\
&\quad x_t^\omega - x_t^{\omega'} = 0,\ \forall \omega, \omega' \in \Omega_g : \omega \neq \omega',\ g \in \mathscr{G}_t,\ t \leq T-1 \\
&\quad y_t^\omega - y_t^{\omega'} = 0,\ \forall \omega, \omega' \in \Omega_g : \omega \neq \omega',\ g \in \mathscr{G}_t,\ t \leq T-1 \\
&\quad x_t^\omega \in \{0,1\}^{nx_t^\omega},\ y_t^\omega \in \mathscr{R}^{+ny_t^\omega},\ \forall \omega \in \Omega,\ t \in \mathscr{T}.
\end{aligned} \tag{2}
$$

Observe that for a given stage t, $A_t'^\omega$ and A_t^ω are the technology and recourse matrices for the x_t variables and $B_t'^\omega$ and B_t^ω are the corresponding ones for the y_t variables. Notice that $x_t^\omega - x_t^{\omega'} = 0$ and $y_t^\omega - y_t^{\omega'} = 0$ are the *NAC*. Finally, nx_t^ω and ny_t^ω denote the dimensions of the vectors of the variables x and y, respectively, related to stage t under scenario ω.

3. SCENARIO CLUSTERING IN SCENARIO TREES

It is clear that the explicit representation of the *NAC* is not required for all pairs of scenarios in order to reduce the dimensions of model. In fact, we can represent implicitly the *NAC* for some pairs of scenarios in order to gain computational efficiency.

Definition 1. *A* scenario cluster *is a set of scenarios whose NAC are implicitly considered in model (2).*

We will decompose the scenario tree into a subset of scenario clusters, where $\mathscr{P} = \{1, ..., q\}$ denotes the set of clusters and $q = |\mathscr{P}|$. Let Ω^p denote the set of scenarios that belongs to a generic cluster p, where $p \in \mathscr{P}$ and $\sum_{p=1}^{q} |\Omega^p| = |\Omega|$. It is clear that the criterion for scenario clustering in the sets, say, $\Omega^1, \ldots, \Omega^q$ is instance dependent. Moreover, we favor the approach that shows higher scenario clustering for greater number of scenario groups in common. In any case, notice that $\Omega^p \bigcap \Omega^{p'} = \emptyset$, $p, p' = 1, \ldots, q$: $p \neq p'$ and $\Omega = \cup_{p=1}^{q} \Omega^p$. Let also $\mathscr{G}^p \subset \mathscr{G}$ denote the set of scenario groups for cluster p, such that $\Omega_g \cap \Omega^p \neq \emptyset$ means that $g \in \mathscr{G}^p$, $\mathscr{G}_t^p = \mathscr{G}_t \cap \mathscr{G}^p$ denotes the set of scenario groups for cluster $p \in \mathscr{P}$ in stage $t \in \mathscr{T}$.

Definition 2. *The* ***break stage*** t^* *is the stage t such that the number of scenario clusters is $q = |\mathscr{G}_{t^*+1}|$, where $t^* + 1 \in \mathscr{T}$. Observe that cluster $p \in \mathscr{P}$ includes the scenarios that belong to group $g \in \mathscr{G}_{t^*+1}$, i.e., $\Omega^p = \Omega_g$.*

Notice that the choice of $t^* = 0$ corresponds to the full model and $t^* = T - 1$ corresponds to the scenario partitioning.

4. COMPUTATIONAL EXPERIENCE

The approach has been implemented in a C++ experimental code. It uses the open source optimization engine *COIN-OR* for solving the *LP* relaxation and mixed 0-1 submodels, in particular, we have used the functions: Clp (LP solver), Cbc (MIP solver), Cgl (Cut generator), Osi, OsiClp, OsiCbc and CoinUtils.

The computational experiments were conducted in a Workstation Debian Linux (kernel v2.6.26 with 64 bits), 2 processors Xeon 5355 (Quad Core with 2x4 cores), 2.664 Ghz and 16 Gb of RAM.

Table 1 gives the dimensions of the *DEM* of the full stochastic model in compact representation for difficult medium and large scale problems. Table 2 gives μ, the mean and σ, standard deviation for dimensions of the cluster submodels; so, we can observe the variability of the nonsymmetric clusters. The headings are as follows: *m*, number of constraints; *nx*, number of 0-1 variables; *ny*, number of continuous variables; *nel*, number of nonzero coefficients in the constraint matrix; and *dens*, constraint matrix density (in %).

Inst.	*m*	*nx*	*ny*	*nel*	*dens*
P1	696	160	376	1550	0.42
P2	1202	530	241	3053	0.33
P3	7282	1878	4152	20818	0.05
P4	16172	4270	9340	53257	0.02
P5	23907	5560	11675	68937	0.02
P6	32914	6672	14010	105854	0.02
P7	2085	450	1155	9105	0.27
P8	4696	1090	2516	9935	0.06
P9	11298	2668	5962	25262	0.03
P10	16870	4600	10430	42015	0.02
P11	31648	7984	17676	83252	0.01
P12	40020	8847	19377	100680	0.01
P13	5256	1176	2904	12861	0.06
P14	11121	2538	6045	27315	0.03
P15	14570	3370	7830	32508	0.02
P16	28176	6584	15008	62934	0.01
P17	45844	10794	24256	102480	0.01
P18	76424	18108	40208	170954	0.00

Table 1: *Testbed problem dimensions*

Table 3 shows some results of our computational experimentation. The headings are as follows: $|\mathscr{P}|$, number of clusters; $|\Omega|$, number of scenarios; $|\mathscr{G}|$, number of scenario groups; Z_{LP}, solution value of the *LP* relaxation of the original *DEM* in compact

Inst.	μ_m (σ_m)	μ_{nx} (σ_{nx})	μ_{nel} (σ_{nel})	μ_{dens} (σ_{dens})
P1	133 (30)	28 (7)	275 (62)	2.31 (0.68)
P2	496 (68)	230 (20)	1227 (171)	0.76 (0.08)
P3	869 (305)	193 (70)	2145 (767)	0.46 (0.20)
P4	1788 (579)	397 (131)	4961 (1617)	0.24 (0.09)
P5	2815 (21)	561 (4)	6953 (51)	0.14 (0.00)
P6	3823 (28)	673 (5)	10675 (78)	0.13 (0.00)
P7	750 (187)	160 (43)	3236 (859)	0.80 (0.20)
P8	643 (191)	138 (41)	1259 (372)	0.49 (0.21)
P9	1241 (537)	269 (117)	2544 (1097)	0.30 (0.18)
P10	2007 (454)	516 (146)	4711 (1333)	0.15 (0.03)
P11	3322 (1208)	729 (266)	7608 (2759)	0.11 (0.05)
P12	3748 (1455)	740 (288)	8423 (3265)	0.12 (0.06)
P13	950 (260)	199 (56)	2171 (610)	0.37 (0.14)
P14	1751 (544)	365 (114)	3930 (1217)	0.20 (0.06)
P15	1973 (617)	423 (133)	4081 (1275)	0.17 (0.09)
P16	3403 (984)	733 (212)	7010 (2025)	0.09 (0.03)
P17	5000 (2216)	1081 (480)	10266 (4549)	0.08 (0.05)
P18	5126 (1967)	824 (317)	8604 (3300)	0.07 (0.03)

Table 2: *Testbed cluster-subproblem dimensions*

representation; Z_0, optimal expected solution value obtained by solving independently the mixed 0-1 cluster submodels; z_{MIP}, optimal solution value of the original *DEM*.We can observe the very good lower bounds Z_0, that can allow to improve the convergence speed of the algorithm.

Inst.	$\lvert\mathscr{P}\rvert$	$\lvert\Omega\rvert$	$\lvert\mathscr{G}\rvert$	Z_{LP}	Z_0	z_{MIP}
P1	6	52	80	4395695	4654305	4654305
P2	3	6	12	75103.6	58589.1	58585.1
P3	10	247	313	5691.3	442336	573848
P4	11	347	427	11601.4	725490	903367
P5	10	1001	1112	4977.8	385471	468277
P6	10	1001	1112	6116.5	540241	653638
P7	3	13	30	20210.9	964395	973038
P8	8	377	545	3156.8	156064	156064
P9	10	1021	1334	3829.5	239683	239683
P10	9	674	920	5757.0	394469	505729
P11	11	1569	1996	5474.1	401435	401435
P12	9	674	920	5757.0	394469	505729
P13	6	208	392	8071.8	371498	372296
P14	7	523	846	6157.3	339381	339381
P15	8	1140	1685	3941.7	212593	212593
P16	9	2372	3292	3521.9	258977	258977
P17	10	4063	5397	2629.0	303900	303900
P18	11	7058	9054	3824.7	318958	318958

Table 3: *Computational results. Stochastic solution*

It is well known that one of the most important contributions to the advancement of the theory and applications of deterministic integer & combinatorial optimization has been the development of the preprocessing techniques for solving large scale instances in affordable computing effort, due to the tightening of the models and, so, reducing the *LP* feasible space without eliminating any feasible integer solution that potentially could become the optimal one. Some of the key ingredients in preprocessing are the probing techniques [8, 9, 10] and schemes for identifying and appending Gomory cuts [11, 12] and clique cuts [13], among other important schemes. So, our algorithm for solving large instances of the mixed integer *DEM* takes benefit from the preprocessing techniques of the optimization engine of choice. They are used for solving the auxiliary mixed integer submodels related to the scenario clusters. The difference in computing time by using preprocessing compared with the alternative that does not use it

		Nonsymmetric BFC-MS			*B&B*	
Inst.	T	$nTNF$	tt	tt_C	tt	tt_C
P1	4	1	0.4	0.3	4000.2	0.8
P2	4	114	198.1	138.8	1304.2	1304.2
P3	4	8	21.8	1.7	41.4	1.7
P4	4	16	171.6	11.7	1530.8	19.4
P5	4	8	162.1	8.8	448.5	13.4
P6	4	10	229.5	8.5	889.7	48.4
P7	5	81	142.5	41.8	188.3	35.9
P8	5	1	2.7	0.9	272.3	6.1
P9	5	1	9.1	1.4	100.0	4.5
P10	5	10	206.6	45.8	7992.7	296.4
P12	5	1	80.2	14.9	12113.1	126.8
P12	5	7	513.8	66.8	3566.2(*)	867.5
P13	6	3	13.2	2.6	1304.2	10.2
P14	6	1	14.2	3.5	—	22.9
P15	6	1	19.7	4.9	7226.3(*)	19.2
P16	6	1	81.2	26.4	628.5(*)	48.5(*)
P17	6	1	152.8	8.7	1897.3	67.3
P18	6	1	377.0	24.1	—	202.9

—: Time limit exceeded (6 hours)
(*): Time for obtaining quasioptimum (0.05)

Table 4: *Nonsymmetric BFC-MS performance vs B&B*

is crucial in solving large scale instances. Table 4 shows the efficiency and stability of the *Nonsymmetric BFC-MS* algorithm proposed in the full version [7] of the paper. The headings are as follows: T, number of stages; $nTNF$, number of *TNF*s; *B&B*, plain use of the Branch-and-Bound procedure for the full model by using the Cbc function of *COIN-OR*; and tt and tt_C, total elapsed time (in seconds) without and with preprocessing. Although other break stages have been considered, we have obtained the best results with the break stage $t^* = 1$ and, then, $q = |\mathscr{G}_2|$ for both the *Nonsymmetric BFC-MS* algorithm and the plain use of the Cbc function of *COIN-OR*.

5. CONCLUSIONS

A modeling approach and an exact Branch-and-Fix Coordination algorithmic framework, so-called *Nonsymmetric BFC-MS*, is being proposed in the full version of the paper for solving multi-stage mixed 0-1 & combinatorial problems under uncertainty in the parameters. The 0-1 and continuous variables can also appear at any stage. The approach treats the uncertainty by scenario cluster analysis, allowing the scenario tree to be nonsymmetric. This last feature has not been considered in the literature that we are aware of. However, in our opinion, it is crucial for solving medium and large scale problems, since the real-life mixed integer optimization problems under uncertainty that, at least, we have encountered have very frequently nonsymmetric scenarios to represent the uncertainty. We can observe (1) the efficiency of using the preprocessing techniques (i.e., probing and Gomory and clique cuts identification and appending schemes) and (2) the astonishing small computing time required by the proposed algorithm, such that it clearly outperforms the plain use of the optimization engine of choice.

6. ACKNOWLEDGEMENTS

This research has been partially supported by the projects ECO2008-00777 ECON from the Ministry of Education and Science, Grupo de Investigación IT-347-10 from the Basque Government, URJC-CM-2008-CET-3703 and RIESGOS CM from

Comunidad de Madrid, and PLANIN MTM2009-14087-C04-01 from Ministry of Science and Innovation, Spain. The full paper is to be submitted for publication in a regular journal.

7. REFERENCES

[1] J. Birge and F. Louveaux, *Introduction to Stochastic Programming*. Springer, 1997.

[2] R. Schultz, "Stochastic programming with integer variables," *Mathematical Programming Ser. B*, vol. 97, pp. 285–309, 2003.

[3] R. Schultz and S. Tiedemann, "Conditional value-at-risk in stochastic programs with mixed integer recourse," *Mathematical Programming Ser. B*, vol. 105, pp. 365–386, 2006.

[4] M. H. van der Vlerk, "Stochastic integer programming bibliography," World Wide Web, http://www.eco.rug.nl/mally/biblio/sip.html, 1996-2007.

[5] R. Rockafellar and R.-B. Wets, "Scenario and policy aggregation in optimisation under uncertainty," *Mathematics of Operations Research*, vol. 16, pp. 119–147, 1991.

[6] M. M. L.F. Escudero, A. Garín and G. Pérez, "On bfc-msmip strategies for scenario cluster partitioning, and twin node family branching selection and bounding for multistage stochastic mixed integer programming," *Computers & Operations Research*, vol. 37, pp. 738–753, 2010.

[7] ——, "An algorithmic framework for solving large scale multi-stage stochastic mixed 0-1 problems with nonsymmetric scenario trees," *To be submmited for publication*, 2011.

[8] E. J. M. Guignard and K. Spielberg, "Logical processing in integer programming," *Annals of Operations Research*, vol. 140, pp. 263–304, 2005.

[9] M. Guignard and K. Spielberg, "Logical reduction methods in zero-one programming. minimal preferred variables," *Operations Research*, vol. 29, pp. 49–74, 1981.

[10] M. Savelsbergh, "Preprocessing and probing techniques for mixed integer programming problems," *ORSA Jornal of Computing*, vol. 6, pp. 445–454, 1994.

[11] G. Cornuejols, "Revival of the gomory cuts in the 1990s," *Operations Research*, vol. 149, pp. 63–66, 2007.

[12] R. Gomory, *Recent Advances in Mathematical Programming*. R.L. Graves and P. Wolfe, Eds. McGraw-Hill, 1963, ch. An algorithm for integer solutions to linear programs, pp. 269–302.

[13] E. J. H. Crowder and M. Padberg, "Solving large-scale zero-one linear programming problems," *Annals of Operations Research*, vol. 31, pp. 803–834, 1983.

Scenario cluster lagrangean decomposition in stochastic mixed integer programming

L.F. Escudero [*] M.A. Garín [†] G. Pérez [‡] A. Unzueta [†]

[*] Dpto. Estadística e Investigación Operativa
Universidad Rey Juan Carlos, Móstoles (Madrid), Spain
laureano.escudero@urjc.es

[†] Dpto. de Economía Aplicada III
Universidad del País Vasco, Bilbao (Vizcaya), Spain
{mariaaraceli.garin, aitziber.unzueta}@ehu.es

[‡] Dpto. de Matemática Aplicada, Estadística e Investigación Operativa
Universidad del País Vasco, Leioa (Vizcaya), Spain
gloria.perez@ehu.es

ABSTRACT

In this paper we introduce a scenario cluster based Lagrangean Decomposition (LD) scheme for obtaining strong lower bounds to the optimal solution of two-stage stochastic mixed 0-1 problems. At each iteration of the Lagrangean based procedures, the traditional aim consists of obtaining the optimal solution value of the corresponding Lagrangean dual via solving scenario submodels once the nonanticipativity constraints have been dualized. Instead of considering a splitting variable representation over the set of scenarios, we propose to decompose the model into a set of scenario clusters. We compare the computational performance of several Lagrangean dual schemes, as the Subgradient Method, the Volume Algorithm and the Progressive Hedging Algorithm for different number of the scenario clusters and different dimensions of the original problem. Our computational experience shows how the bound value and its computational effort depend on the number of scenario clusters to consider. In any case, the computational experience reported in this extended abstract (as well as the extensive one reported in the full paper) shows that the scenario cluster LD scheme outperforms the traditional LD scheme for single scenarios both in lower bounds's quality and computing effort. All the procedures have been implemented in a C++ experimental code that uses the open source optimization engine *COIN-OR*, for solving the auxiliary *LP* and mixed 0-1 cluster submodels. We also give computational evidence of the model tightening effect that preprocessing techniques have in stochastic integer optimization as well, by using the probing and Gomory and clique cuts identification and appending schemes of the optimization engine of choice.

Keywords: Stochastic integer programming, Lagrangean decomposition, Subgradient, Volume, Progressive hedging algorithm, Scenario clusters

1. INTRODUCTION

In this work we consider a general two-stage stochastic mixed 0-1 problem. The uncertainty is modeled via a finite set of scenarios $\omega = 1, ..., |\Omega|$, each with an associated probability of occurrence w^{ω}, $\omega \in \Omega$. The traditional aim in this type of problems is to solve the so-called Deterministic Equivalent Model (DEM), which is a mixed 0-1 problem with a special structure, see e.g. [1] for a good survey on some mayor results in the area obtained during the last decade. A Branch-and-Bound algorithm for problems having mixed-integer variables in both stages is designed in [2], among others, by using Lagrangean relaxation for obtaining lower bounds to the optimal solution of the original problem. A Branch-and-Fix Coordination (BFC) methodology for solving such DEM in production planning under uncertainty is given in [3, 4], but the approach does not allow continuous first stage variables or 0-1 second stage variables. We propose in [5, 6] a BFC algorithmic framework for obtaining the optimal solution of the two-stage stochastic mixed 0-1 integer problem, where the uncertainty appears anywhere in the coefficients of the 0-1 and continuous variables in both stages. Recently, a general algorithm for two-stage problems is described in [7]. We study in [8] several solution methods for solving the dual problem corresponding to the Lagrangean Decomposition (LD) of two-stage stochastic mixed 0-1 models. At each iteration of these Lagrangean based procedures, the traditional aim consists of obtaining the optimal solution value of the corresponding parametric mixed 0-1 Lagrangean dual problem via solving scenario submodels once the nonanticipativity constraints (NAC) have been dualized, and the parameters (i.e., the Lagrangean multipliers) are updated by using different subgradient based methodologies.

Instead of considering a splitting variable representation over the set of scenarios, in this paper we propose to decompose the model into a set of scenario clusters. For different choices of the number of scenario clusters we computationally compare the solution given by the plain use of the optimization engine *COIN-OR*, see [9], against various schemes for solving the Lagrangean dual problems. After this comparison we observe that very frequently the new bounds give the optimal solution to the original problem. Moreover, the performance of the scenario cluster LD scheme outperforms the LD scheme based on single scenarios in both the bounds's quality and computing effort. These successful results may open the possibility for tightening the lower bounds of the solution at the candidate Twin Node Families in the exact BFC scheme for both two-stage and multistage types of problems.

2. TWO-STAGE STOCHASTIC MIXED 0-1 PROBLEM

Let us consider the *compact* representation of the DEM of a two-stage stochastic mixed integer problem (*MIP*),

$$
\begin{array}{lll}
(MIP): & z_{MIP} = \min c_1^T \delta + c_2^T x + \sum_{\omega\in\Omega} [w^\omega q_1^{\omega T} \gamma^\omega + w^\omega q_2^{\omega T} y^\omega] & \\
\text{s.t.} & b_1 \le A \begin{pmatrix} \delta \\ x \end{pmatrix} \le b_2 & \\
& h_1^\omega \le T^\omega \begin{pmatrix} \delta \\ x \end{pmatrix} + W^\omega \begin{pmatrix} \gamma^\omega \\ y^\omega \end{pmatrix} \le h_2^\omega, \omega \in \Omega & \\
& \delta, \gamma^\omega \in \{0,1\}, x, y^\omega \ge 0, \forall \omega \in \Omega, &
\end{array} \tag{1}
$$

where the uncertainty in the parameters is introduced by using a scenario analysis approach, such that a scenario consists of a realization of all random variables in both stages through a scenario tree. Notice that there are two types of decision variables at each stage, namely, the set of δ 0-1 and x continuous variables for the first stage, and the set of γ^ω 0-1 and y^ω continuous variables for the second stage. Notice also that for simplifying reasons, the objective function to optimize in the models dealt with in this paper is the expected value over the set of scenarios Ω.

Let us suppose that we have selected a set of scenario clusters for the second stage, whose number is say $\hat{p}$. In general, given a scenario tree, $\hat{p}$ can be chosen as any value between 1 and $|\Omega|$. Now, we can represent the *MIP* model (1) by a splitting variable representation, see [10, 11] among others, where the full model is included by the $\hat{p}$ cluster submodels and their related linking NAC. Additionally, we consider a compact representation for the Ω^p scenarios into each cluster submodel p, where $p \in \{1,...,\hat{p}\}$, and $|\Omega^p|$ defines the size of scenario cluster, p, i.e., the number of scenarios that belong to the corresponding cluster, for $p = 1,...,\hat{p}$. The scenario clusters are defined in terms of consecutive scenarios, $\Omega^1 = \{1,...,|\Omega^1|\}$, $\Omega^2 = \{|\Omega^1|+1,...,|\Omega^1|+|\Omega^2|\}$,..., $\Omega^{\hat{p}} = \{|\Omega^1|+...+|\Omega^{\hat{p}-1}|+1,...,|\Omega|\}$. The mixed 0-1 submodel to consider for each scenario cluster p can be expressed by the *compact* representation,

$$
\begin{array}{lll}
(MIP^p): & \mathbf{z}^p = \min \mathbf{w}^p c_1^T \delta^p + \mathbf{w}^p c_2^T \mathbf{x}^p + \sum_{\omega\in\Omega^p} w^\omega [q_1^{\omega T} \gamma^\omega + q_2^{\omega T} y^\omega] & \\
\text{s.t.} & b_1 \le A \begin{pmatrix} \delta^p \\ \mathbf{x}^p \end{pmatrix} \le b_2 & \\
& h_1^\omega \le T^\omega \begin{pmatrix} \delta^p \\ \mathbf{x}^p \end{pmatrix} + W^\omega \begin{pmatrix} \gamma^\omega \\ y^\omega \end{pmatrix} \le h_2^\omega, \omega \in \Omega^p & \\
& \mathbf{x}^p \ge 0, \delta^p \in \{0,1\}, \gamma^\omega \in \{0,1\}, y^\omega \ge 0, \forall \omega \in \Omega^p, &
\end{array} \tag{2}
$$

where $\mathbf{w}^p = \sum_{\omega\in\Omega^p} w^\omega$ denotes the likelihood for scenario cluster p, and δ^p and $\mathbf{x}^p$ are the variable vectors δ and x for scenario cluster p. Moreover, the $\hat{p}$ submodels (2) are linked by the NAC

$$
\delta^p - \delta^{p'} = 0 \tag{3}
$$

$$
\mathbf{x}^p - \mathbf{x}^{p'} = 0, \tag{4}
$$

for $p, p' = 1,\ldots,\hat{p} : p \ne p'$. So, the mixed 0-1 DEM (1) is equivalent to the *splitting variable* representation over the set of scenario clusters,

$$
\begin{array}{lll}
(MIP): & z_{MIP} = & \sum_{p=1}^{\hat{p}} \mathbf{z}^p \\
& s.t. & \\
& \delta^p - \delta^{p+1} \le 0, & \forall p = 1,...,\hat{p}-1, \\
& \delta^{\hat{p}} \le \delta^1 & \\
& \mathbf{x}^p - \mathbf{x}^{p+1} \le 0, & \forall p = 1,...,\hat{p}-1,. \\
& \mathbf{x}^{\hat{p}} \le \mathbf{x}^1. &
\end{array} \tag{5, 6}
$$

Observe that the NAC (3)-(4) have been represented as the set of inequalities (6), in order to avoid the use of non-signed vectors of Lagrangean multipliers in the dualization of such constraints, see below. Additionally, notice that for $\hat{p} = 1$, the model (5)-(6) coincides with the mixed 0-1 DEM in the compact representation (1), and for $\hat{p} = |\Omega|$ we obtain the splitting variable representation via scenarios.

3. SCENARIO CLUSTERING IN SCENARIO TREES

The scenario cluster Lagrangean Decomposition (LD) of the mixed 0-1 DEM, (*MIP*) model (5)-(6), for a given set of scenario clusters and a given nonnegative vector of Lagrangean multipliers $\mu = (\mu_\delta, \mu_x)$, is the μ-parametric mixed 0-1 minimization model (7) in (δ, x, γ, y) with objective function value $z_{LD}(\mu, \hat{p})$. Let us denote this model as $(MIP_{LD}^{\hat{p}}(\mu))$.

$$
\begin{array}{lll}
(MIP_{LD}^{\hat{p}}(\mu)): & z_{LD}(\mu,\hat{p}) = \min \sum_{p=1}^{\hat{p}} [\mathbf{w}^p c_1^T \delta^p + \mathbf{w}^p c_2^T \mathbf{x}^p + & \\
& + \sum_{\omega\in\Omega^p} w^\omega [q_1^{\omega T} \gamma^\omega + q_2^{\omega T} y^\omega]] + & \\
& + \sum_{p=1}^{\hat{p}-1} \mu_\delta^p (\delta^p - \delta^{p+1}) + \mu_\delta^{\hat{p}} (\delta^{\hat{p}} - \delta^1) + & \\
& + \sum_{p=1}^{\hat{p}-1} \mu_x^p (\mathbf{x}^p - \mathbf{x}^{p+1}) + \mu_x^{\hat{p}} (\mathbf{x}^{\hat{p}} - \mathbf{x}^1) &
\end{array} \tag{7}
$$

$$
\begin{array}{ll}
\text{s.t.} & b_1 \le A \begin{pmatrix} \delta^p \\ \mathbf{x}^p \end{pmatrix} \le b_2, p = 1,...,\hat{p} \\
& h_1^\omega \le T^\omega \begin{pmatrix} \delta^p \\ \mathbf{x}^p \end{pmatrix} + W^\omega \begin{pmatrix} \gamma^\omega \\ y^\omega \end{pmatrix} \le h_2^\omega, \omega \in \Omega^p, p = 1,...,\hat{p} \\
& \mathbf{x^p} \ge 0, \delta^p \in \{0,1\}, \forall p = 1,...,\hat{p} \\
& y^\omega \ge 0, \gamma^\omega \in \{0,1\}, \forall \omega \in \Omega^p, p = 1,...,\hat{p}
\end{array}
$$

It is well known that $(MIP_{LD}^{\hat{p}}(\mu))$ is a relaxation of (*MIP*), since (i) the feasible set of $(MIP_{LD}^{\hat{p}}(\mu))$ contains the feasible set of (*MIP*), and (ii) for any (δ, x, γ, y) feasible for (*MIP*) and any $\mu \ge 0$ and $1 < \hat{p} \le |\Omega|$, it results that $z_{LD}(\mu,\hat{p}) \le z_{MIP}$. Notice that if $\hat{p} = 1$, for any $\mu \ge 0$, $z_{LD}(\mu, 1) = z_{MIP}$ by definition of the compact representation. Then, it follows that the optimal value $z_{LD}(\mu,\hat{p})$, which depends on μ is a lower bound of the optimal value of (*MIP*), z_{MIP} for any choice of $\hat{p}$, with $1 < \hat{p} \le |\Omega|$.

Definition 1. *For any choice of $\hat{p}$, with $1 < \hat{p} \le |\Omega|$, the problem of finding the tightest Lagrangean lower bound on z_{MIP} is*

$$
(MIP_{LD}): \; z_{LD} = \max_{\mu\ge 0} z_{LD}(\mu,\hat{p}).
$$

It is called Lagrangean dual of (*MIP*) *relative to the (complicating) NAC (6), and $\hat{p}$ denotes the number of scenario clusters.*

It can be shown, see [16], that the Lagrangean decomposition gives equal or stronger bounds of the solution value of the original problem than the Lagrangean relaxation of the constraints related to any of the scenario clusters to be decomposed. See also [14].

Given a choice of the set of $\hat{p}$ scenario clusters, the μ-parametric $(MIP_{LD}^{\hat{p}}(\mu))$ model (7) must be solved, where the parametric vector $(\mu) = (\mu_\delta, \mu_x)$ is given. Moreover, the corresponding objective function in (7) can be rewritten as the sum of the objective function values of smaller submodels, one for each scenario cluster.

$$
\begin{array}{lll}
z_{LD}(\mu,\hat{p}) & = & \min \sum_{p=2}^{\hat{p}} [[\mathbf{w}^p c_1^T + (\mu_\delta^p - \mu_\delta^{p-1})]\delta^p + \\
& + & [\mathbf{w}^p c_2^T + (\mu_x^p - \mu_x^{p-1})]\mathbf{x}^p + \\
& + & \sum_{\omega\in\Omega^p} w^\omega [q_1^{\omega T} \gamma^\omega + q_2^{\omega T} y^\omega]] + \\
& + & [\mathbf{w}^1 c_1^T + (\mu_\delta^1 - \mu_\delta^{\hat{p}})]\delta^1 + \\
& + & [\mathbf{w}^1 c_2^T + (\mu_x^1 - \mu_x^{\hat{p}})]\mathbf{x}^1 + \\
& + & \sum_{\omega\in\Omega^1} w^\omega [q_1^{\omega T} \gamma^\omega + q_2^{\omega T} y^\omega]
\end{array} \tag{8}
$$

$$
\begin{array}{ll}
\text{s.t.} & b_1 \le A \begin{pmatrix} \delta^p \\ \mathbf{x}^p \end{pmatrix} \le b_2, \quad p = 1,...,\hat{p} \\
& h_1^\omega \le T^\omega \begin{pmatrix} \delta^p \\ \mathbf{x}^p \end{pmatrix} + W^\omega \begin{pmatrix} \gamma^\omega \\ y^\omega \end{pmatrix} \le h_2^\omega, \quad \omega \in \Omega^p, p = 1,...,\hat{p} \\
& \mathbf{x}^p \ge 0, \delta^p \in \{0,1\}, \quad \forall p = 1,...,\hat{p} \\
& y^\omega \ge 0, \gamma^\omega \in \{0,1\}, \quad \forall \omega \in \Omega^p, p = 1,...,\hat{p}
\end{array}
$$

That is, the $MIP_{LD}^{\hat{p}}(\mu)$ model can be decomposed in $\hat{p}$ smaller submodels, and its optimal solution value be calculated as the sum of the related $\mathbf{z}_{LD}^p(\mu^p)$ values, i.e., the optimal solution value of each pth scenario cluster model, and $p = 1,...,\hat{p}$.

4. COMPUTATIONAL EXPERIENCE

We have implemented the three procedures: Subgradient method [17], Volume algorithm [12] and Progresive Hedging Algorithm [11] in a C++ experimental code. The free optimization engine *COIN-OR* is used for solving the linear and mixed 0-1 auxiliary submodels and the whole model as well. The computational experiments were conducted in a Workstation Debian Linux (kernel v2.6.26 with 64 bits), 2 processors Xeon 5355 (Quad Core with 2x4 cores), 2.664 Ghz and 16 Gb of RAM. Table 1 gives the dimensions of the mixed 0-1 DEM, in compact representation. The headings are as follows: m, number of constraints; $n_{\delta+\gamma}$, number of 0-1 variables; n_{x+y}, number of continuous variables; *nel*, number of nonzero coefficients in the constraint matrix; *dens*, constraint matrix density (in %); and $|\Omega|$, number of scenarios. The testbed used for the reported experimentation is available from the authors under request.

Table 1: Testbed problem dimensions

Case	m	$n_{\delta+\gamma}$	n_{x+y}	*nel*	*dens*	$\|\Omega\|$
P1	136	132	132	2112	5.88	32
P2	148	138	138	3984	9.75	32
P3	324	483	327	6440	2.45	80
P4	520	516	516	8256	1.54	128
P5	520	516	516	8256	1.54	128
P6	516	771	519	10280	1.54	128
P7	532	522	522	14736	2.65	128
P8	1290	1290	1290	51400	1.54	128
P9	712	612	612	146496	16.81	128

Table 2: Stochastic Solution

Case	z_{MIP}	z_{LP}	*GAP*	T_{COIN}	T_{LP}	$\bar{z}_{LD}$
P1	-80.48	-81.14	0.81	0.71	0.01	-73.02
P2	-99.89	-100.42	0.52	1.12	0.02	-90.38
P3	-45.61	-47.48	3.95	256.30	0.03	-4.74
P4	-23.86	-27.19	12.23	10.55	0.05	-13.59
P5	-28.75	-31.71	9.33	16115.80	0.04	-3.17
P6	—	-52.77	—	—	0.03	-5.27
P7	-218.67	-277.95	21.33	6.33	0.07	-27.79
P8	—	-63.76	—	—	0.13	-6.37
P9	-1937.85	-2070.47	6.40	2.01	0.36	-5.27

—: Time limit exceeded (7 hours)

Table 2 shows some results of our computational experimentation. See an extensive computational experience in the full paper [19]. The headings of table 2 are as follows: z_{MIP} and z_{LP}, solution values of the original stochastic mixed 0-1 problem and its LP relaxation, respectively; *GAP*, optimality gap defined as $\frac{z_{MIP}-z_{LP}}{z_{LP}}$ (in %); T_{COIN} and T_{LP}, elapsed times (in seconds) to obtain the z_{MIP} and z_{LP} solution values, respectively, by plain use of *COIN-OR*; and $\bar{z}_{LD}$, upper bound of the optimal solution value of the original problem.

Table 3 shows some of our main computational results. We present the Lagrangen bounds that we obtain, with $\hat{p} = 4$ scenario clusters, see the results for other choices of the number of clusters in the full paper. The headings are as follows: z_{SUB}, z_{VOL}, and z_{PHA}, lower bounds of the optimal solution for the original problem obtained by the Subgradient Method (SUB), Volume Algorithm (VOL) and Progressive Hedging Algorithm (PHA), respectively; T_{SUB}, T_{VOL} and T_{PHA} elapsed times (in seconds) to compute the related Lagrangean bounds; and, finally, nit_{SUB}, nit_{VOL} and nit_{PHA}, number of iterations to compute the corresponding bounds.

The results in bold font are those where the Lagrangean bound coincides with the optimal solution value of the original stochastic integer problem. Notice that in general all the Lagrangean bounds obtained are very close to the optimal solution value, so, they are very good bounds, and are obtained after few iterations (zero, in many cases).

Table 3: Lagrangean bounds with $\hat{p} = 4$ scenario clusters

	z_{SUB}	T_{SUB}	nit_{SUB}
P1	**-80.48**	7.42	35
P2	**-99.89**	3.72	20
	z_{VOL}	T_{VOL}	nit_{VOL}
P1	**-80.48**	7.54	37
P2	-99.94	0.99	5
	z_{PHA}	T_{PHA}	nit_{PHA}
P1	**-80.48**	15.25	72
P2	**-99.89**	5.66	30
	z_{SUB}	T_{SUB}	nit_{SUB}
P3	**-45.61**	22.09	35
	z_{VOL}	T_{VOL}	nit_{VOL}
P3	**-45.61**	28.04	41
	z_{PHA}	T_{PHA}	nit_{PHA}
P3	-45.64	31.44	53
	z_{SUB}	T_{SUB}	nit_{SUB}
P4	**-23.86**	23.65	21
P5	-28.76	21.86	10
P6	-49.79	16.10	0
P7	**-218.67**	0.66	0
P8	-61.63	10593.90	29
P9	**-1937.85**	2.19	0
	z_{VOL}	T_{VOL}	nit_{VOL}
P4	**-23.86**	235.52	202
P5	**-28.75**	926.75	362
P6	-49.79	16.54	0
P7	**-218.67**	0.66	0
P8	-61.76	1212.43	3
P9	**-1937.85**	0.53	0
	z_{PHA}	T_{PHA}	nit_{PHA}
P4	**-23.86**	32.89	32
P5	-28.76	68.24	33
P6	-49.79	16.26	0
P7	**-218.67**	0.60	0
P8	-61.60	22071.50	59
P9	**-1937.85**	0.42	0

We propose to use preprocessing and probing techniques [15] and schemes for identifying and appending Gomory cuts and clique cuts [18] before solving the scenario cluster mixed 0-1 submodels. As it is well known, due to the tightening of these models, it is possible to reduce the LP feasible space without eliminating any feasible integer solution that potentially could become the optimal one. The difference in computing time by using preprocessing compared with the alternative that does not use it can be crucial in the whole procedure for obtaining good bounds for large scale instances with affordable computing effort. However, a very good performance of the Volume Algorithm has been reported in [13] for a highly combinatorial problem as it is the stochastic set packing problem.

5. CONCLUSIONS

In this paper we have presented a scenario cluster based Lagrangean Decomposition (LD) scheme for obtaining strong lower bounds to the optimal solution of two-stage stochastic mixed integer problems, where the uncertainty appears anywhere in the coefficients of the 0-1 and continuous variables in the objective function and

constraints in both stages. For obtaining the bounds we have used three popular subgradient based schemes, namely, the traditional Subgradient Method, the Volume Algorithm and the Progressive Hedging Algorithm. Based on the computational results that we have presented (and the extensive computational experience reported in the full paper), we can draw some conclusions: (1) Very frequently the new bounds give the optimal solution to the original problem; (2) The performance of the scenario cluster LD scheme outperforms the LD scheme based on single scenarios in both the bounds's quality and computing effort; and (3) it is difficult for the new Lagrangean multipliers updating schemes to outperform the traditional Subgradient Method in this type of problems.

6. ACKNOWLEDGEMENTS

This research has been partially supported by the projects ECO2008-00777 ECON from the Ministry of Education and Science, Grupo de Investigación IT-347-10 from the Basque Government, grant FPU ECO2006 from the Ministry of Education and Science, URJC-CM-2008-CET-3703 and RIESGOS CM from Comunidad de Madrid, and PLANIN MTM2009-14087-C04-01 from Ministry of Science and Innovation, Spain. The full paper is to be submitted for publication in a regular journal.

7. REFERENCES

[1] R. Schultz, "Stochastic programming with integer variables," *Mathematical Programming Ser. B*, vol. 97, 2003.

[2] C. Carøe and R. Schultz, "Dual decomposition in stochastic integer programming," *Operations Research Letters*, vol. 24, 1999.

[3] A. Alonso-Ayuso, L. Escudero, and M. Ortuño, "Branch-and-fix coordination algorithmic framework for solving some types of stochastic pure and mixed 0-1 programs," *European Journal of Operational Research*, vol. 151, 2003.

[4] A. Alonso-Ayuso, L. Escudero, M. Garín, M. Ortuño, and G. Pérez, "An approach for strategic supply chain planning based on stochastic 0–1 programming," *Journal of Global Optimization*, vol. 26, 2003.

[5] L. Escudero, M. Garín, M. Merino, and G. Pérez, "A general algorithm for solving two-stage stochastic mixed 0-1 first stage problems," *Computers and Operations Research*, vol. 36, 2009.

[6] ——, "An exact algorithm for solving large-scale two-stage stochastic mixed integer problems: some theoretical and experimental aspects," *European Journal of Operational Research*, vol. 204, 2010.

[7] H. Sherali and J. Smith, "Two-stage hierarchical multiple risk problems: Models and algorithms," *Mathematical Programming S. A*, vol. 120, 2009.

[8] L. Escudero, M. Garín, G. Pérez, and A. Unzueta, "Lagrangean decomposition for large-scale two-stage stochastic mixed 0-1 problems," *Working paper serie Biltoki DT.2010.07. http://econpapers.repec.org/paper/ehubiltok/201007.htm, UPV/EHU. Also submmited to TOP*, 2010.

[9] INFORMS, "Coin-or: Computational infrastructure for operations research," www.coin-or.org, 2010.

[10] J. Birge and F. Louveaux, *Introduction to Stochastic Programming*. Springer, 1997.

[11] R. Rockafellar and R.-B. Wets, "Scenario and policy aggregation in optimisation under uncertainty," *Mathematics of Operations Research*, vol. 16, 1991.

[12] F. Barahona and R. Anbil, "The volume algorithm: Producing primal solutions with a subgradient method," *Mathematical Programming*, vol. 87, 2000.

[13] L. Escudero, M. Landete, and A. Rodriguez-Chia, "Stochastic set packing problem," *European Journal of Operational Research, accepted for publication*, 2010.

[14] M. Guignard, "Lagrangean relaxation," *TOP*, vol. 11, 2003.

[15] M. Guignard and K. Spielberg, "Logical reduction methods in zero-one programming. minimal preferred variables," *Operations Research*, vol. 29, 2003.

[16] M. Guignard and S. Kim, "Lagrangean decomposition. a model yielding stronger lagrangean bounds," *Mathematical programming*, vol. 39, 1987.

[17] M. Held and R. M. Karp, "The traveling salesman problem and minimum spanning trees: part ii," *Mathematical programming*, vol. 1, 1971.

[18] G. Cornuejols, "Revival of the gomory cuts in the 1990s," *Operations Research*, vol. 149, 2007.

[19] L. Escudero, M. Garín, G. Pérez, and A. Unzueta, "Cluster based decomposition of lagrangean duals," *To be submmited for publication*.

Positive Edge: A Pricing Criterion for the Identification of Non-degenerate Simplex Pivots

Vincent Raymond [†] Francois Soumis [* †] Abdelmoutalib Metrane [*] Mehdi Towhidi [*]
Jacques Desrosiers [* ‡]

[*] GERAD
Montreal, Canada H3T 2A7
{francois.soumis, abdelmoutalib.metrane, mehdi.towhidi}@gerad.ca

[†] Ecole Polytechnique de Montreal
Montreal, Canada H3C 3A7
vincent.raymond@polymtl.ca

[‡] HEC Montreal
Montreal, Canada H3T 2A7
jacques.desrosiers@hec.ca

ABSTRACT

The *Positive Edge* is a new pricing rule for the Primal Simplex: it identifies, with a probability error less than or equal to 2^{-62} in double precision binary floating-point format, variables allowing for non-degenerate pivots. These are identified directly from a short calculation on the original coefficients of the constraint matrix. If such a variable has a negative reduced cost, it strictly improves the objective function value when entered into the basis. Preliminary computational experiments made with CPLEX and COIN-OR show its high potential.

Keywords: Linear programming, Simplex, Degeneracy

1. INTRODUCTION

Consider the following linear programming problem (LP) in standard form

$$\text{minimize } c^\top x \quad \text{subject to: } Ax = b,\ x \geq 0, \tag{1}$$

where $x, c \in \mathbb{R}^n$, $A \in \mathbb{R}^m \times \mathbb{R}^n$, and $b \in \mathbb{R}^m$. We are interested in problems for which the basic solutions are highly degenerate, that is, for which the number of non-zero variables is much less than m, the size of the basis. In that case, the Primal Simplex algorithm is likely to encounter degenerate pivots and possibly to cycle. To avoid cycling, several pivot rules and right-hand side perturbation methods have been proposed, e.g., [2, 10, 1, 6, 9]. However, these do not strongly improve the performance of the Primal Simplex algorithm. Another way is by using the steepest edge criterion [5] which computes the improvement of the cost function for possible entering variables. Hence, if one exists, it selects a variable with a non-degenerate pivot. However, this requires a significant amount of CPU time.

Pan [7] proposes the use of a *reduced problem* with a smaller number of constraints and variables. The method starts with an initial basic solution and identifies its p non-zero basic variables. Constraints are split in two: set P where the basic variables takes a positive value and set Z where the basic variables are zero. Variables are also split in two sets. *Compatible* variables are those for which all values are zero in the updated simplex tableau for constraint indices in Z, other variables are said to be *incompatible*. The $m-p$ constraints in Z are temporarily removed to leave a smaller constraint matrix with only p rows. To preserve feasibility, incompatible variables are also removed to form the reduced problem. Since the $p \times p$ basis of the reduced problem is non-degenerate, the next pivot is automatically non-degenerate. The resulting reduced problem is solved to optimality over the compatible variables and the reduced costs are computed by means of its dual variables. In Pan's method, dual variables of LP corresponding to the $m-p$ eliminated constraints are arbitrarily set to zero. Next, incompatible variables are considered. If such a variable is to become basic, some of the eliminated constraints must be reintroduced in the reduced problem. When compared to his own implementation of the primal simplex algorithm, [7] reports speed-up factors of 4 to 5.

In a column-generation framework (which can be seen as a Primal Simplex approach), authors of [3] propose a *Dynamic Constraint Aggregation* (DCA) method for the solution of the linear relaxation of set partitioning problems. Considering only the p non-zero basic variables, the DCA method identifies identical rows (composed of zeros and ones) of the corresponding columns. In the *constraint aggregation* phase, a single constraint per row-group remains in the reduced problem. Authors show that, once the reduced problem has been solved, the dual variable of a kept constraint is equal to the sum of the dual variables of the corresponding row-group. A full set of dual variables is recovered by distributing adequately the values of the dual variables of the reduced problem. For set partitioning problems, this is done by solving a shortest-path problem. These dual variables are used to price out the generated columns, allowing for updates of the constraint aggregation. On a set of large-scale bus driver scheduling problems, DCA reduces the solution time by a factor of more than 23 over the classical column-generation method.

The *Improved Primal Simplex* (IPS) method of [4] combines ideas from the reduced problem of Pan [7] and from DCA [3] to solve linear programming problems. Considering only the p non-zero basic variables, IPS identifies a set of p rows that are linearly independent and removes from the reduced problem all the other $m-p$ constraints. As in [7], the reduced problem is solved using the compatible variables only. Next, a *complementary problem* is constructed and solved to prove that the current solution of the reduced problem is optimal for LP, otherwise it selects a set of incompatible variables to be introduced into the current reduced problem. Authors of [4] show that when the solution of the reduced prob-

lem is not optimal for LP, the re-optimization after adding all the incompatible variables of the chosen set strictly decreases the objective function value. Indeed, they show that in that case, there exists a convex combination of the selected incompatible variables that is compatible with respect to the reduced problem, hence with a strictly positive step size.

The complementary problem contains all the incompatible variables and its coefficient matrix is created at the same time as the reduced problem. Both problems are built according to a modification of the original constraint matrix. Indeed, this modified matrix (which is the result of an updated simplex tableau) is obtained by multiplying A by the current inverse of the basis matrix. The complexity of computing this modified matrix for the identification of the compatible variables is $O(m^2n)$. The computational results of [8] show that, on medium-sized instances ($m \approx 5000$, $n \approx 25\,000$), IPS is faster than the primal simplex algorithm of CPLEX by factors ranging from 5 to 20. However, on large-scaled problems ($m \approx 100\,000$, $n \approx 450\,000$), constructing the reduced and the complementary problems is too costly compared to the Primal Simplex algorithm itself.

2. THE POSITIVE EDGE RULE

As in IPS, the *Positive Edge* rule gives priority to non-degenerate pivots. However, compatible variables that form the reduced problem are identified directly from the original constraint matrix A instead of from the modified matrix obtained by multiplying it by the inverse of the basis. Determining which variables are compatible is done in $O(mn)$, i.e., $O(m)$ for each variable, the same complexity as for the reduced cost computation of such a variable. Obviously, as in IPS, one might have to execute some degenerate pivots to reach optimality.

3. COMPUTATIONAL EXPERIMENTS

Preliminary computational experiments made with CPLEX show its high potential. We designed a simple algorithm using two external procedures: one identifies variables that allow for non-degenerate pivots while the other identifies variables with negative reduced cost. These are sent to the Primal Simplex algorithm of CPLEX. It has been tested on fourteen medium-sized aircraft fleet assignment instances (5000 constraints and 25 000 variables), two large-scale manpower planning problems (100 000 constraints and 450 000 variables), and nine PDS instances from the Mittelmann library. All these problems are highly degenerate. On the first group, our algorithm is 7.4 times faster than CPLEX on average and the number of pivots is almost reduced by a factor 2. On the second and third groups, it is 50% faster and the number of pivots is decreased by 2.4 and 3.6, respectively. It has also been tested on Fome12 and Fome13 from the Mittelmann library. For these two highly dense problems, our simple implementation failed.

The recent integration of the positive edge rule within the primal simplex code of COIN-OR prevents such cases by eliminating the external procedures and taking advantage of partial pricing strategies. Computational experiments show that the positive edge can help in solving difficult LPs in about half of the time required with the Devex rule.

4. REFERENCES

[1] Bland, R. G. 1977. New Finite Pivoting Rules for the Simplex Method. *Mathematics of Operations Research*, **2**(2), 103–107.

[2] Charnes, A. 1952. Optimality and Degeneracy in Linear Programming. *Econometrica*, **20**, 160–170.

[3] Elhallaoui, I., D. Villeneuve, F. Soumis, and G. Desaulniers. 2005. Dynamic Aggregation of Set Partitioning Constraints in Column Generation. *Operations Research* **53**(4), 632–645.

[4] Elhallaoui, I., A. Metrane, G. Desaulniers, and F. Soumis. 2007. An Improved Primal Simplex Algorithm for Degenerate Linear Programs. Forthcoming: *INFORMS Journal on Computing*.

[5] Forrest, J.J., and D. Goldfarb. 1992. Steepest-Edge Simplex Algorithms for Linear Programming. *Mathematical Programming* **57**(3), 341-374.

[6] Fukuda, K. 1982. *Oriented Matroid Programming*. Ph.D. Dissertation, University of Waterloo, Canada.

[7] Pan, P.-Q. 1998. A Basis Deficiency-Allowing Variation of the Simplex Method for Linear Programming. *Computers and Mathematics with Applications*, **36**(3), 33–53.

[8] Raymond, V., F. Soumis, and D. Orban. 2010. An New Version of the Improved Primal Simplex Algorithm for Degenerate Linear Programs. *Computers and Operations Research*, **37**(1), 91–98.

[9] Ryan, D. M. and Osborne, M. 1988. On the Solution to Highly Degenerate Linear Programmes. *Mathematical Programming*, **41**, 385–392.

[10] Wolfe, P. 1963. A Technique for Resolving Degeneracy in LP. *SIAM Journal*, **11**(2), 205–211.

On the transition from fluence map optimization to fluence map delivery in intensity modulated radiation therapy treatment planning

Humberto Rocha [*] Joana M. Dias [*] [†] Brígida C. Ferreira [‡] [§]
Maria do Carmo Lopes [§]

[*] INESC-Coimbra
Rua Antero de Quental 199, 3000-033 Coimbra, Portugal
hrocha@mat.uc.pt

[†] Faculdade de Economia, Universidade de Coimbra
Av. Dias da Silva 165, 3004–512 Coimbra, Portugal
joana@fe.uc.pt

[‡] I3N, Departamento de Física, Universidade de Aveiro
Campus Universitário de Santiago, 3810–193 Aveiro, Portugal
brigida@ua.pt

[§] Serviço de Física Médica, IPOC-FG, EPE
Av. Bissaya Barreto 98, 3000–075 Coimbra, Portugal
mclopes@ipocoimbra.min-saude.pt

ABSTRACT

The intensity modulated radiation therapy (IMRT) treatment planning problem is usually divided in three smaller problems that are solved sequentially: geometry problem, intensity problem, and realization problem. There are many models and algorithms to address each of the problems satisfactorily. However, the last two problems can not be seen separately, because strong links exist between them. In practice, the linkage between these problems is done, most of the time, by rounding, which can lead to a significant deterioration of the treatment plan quality. We propose a combinatorial optimization approach and use a binary genetic algorithm to enable an improved transition from optimized to delivery fluence maps in IMRT treatment planning. A clinical example of a head and neck cancer case is used to highlight the benefits of using a combinatorial optimization approach when linking the intensity problem and the realization problem.

Keywords: Radiotherapy, IMRT, Fluence Map Optimization, Combinatorial Optimization

1. INTRODUCTION

The goal of radiation therapy is to deliver a dose of radiation to the cancerous region to sterilize the tumor minimizing the damages on the surrounding healthy organs and tissues. In the inverse planning of radiation therapy, for a prescribed treatment plan, a correspondent set of parameters (beams and fluences) is algorithmically computed in order to fulfil the prescribed doses and restrictions. Inverse treatment planning allows the modeling of highly complex treatment planning problems and optimization has a fundamental role in the success of this procedure. An important type of inverse treatment planning is IMRT where the radiation beam is modulated by a multileaf collimator (MLC) that enables the transformation of the beam into a grid of smaller beamlets of independent intensities (see Figure 1). Despite the illustration of Figure 1, beamlets do not exist physically. Their existence is generated by the movement of the leaves of the MLC that block part of the beam during portions of the delivery time. The MLC has movable leaves on both sides that can be positioned at any beamlet grid boundary. In the "step and shoot mode", considered here, the leaves are set to open a desired aperture during each segment of the delivery and radiation is on for a specific fluence time or intensity. This procedure generates a discrete set (the set of chosen beam angles) of intensity maps like in Figure 1.

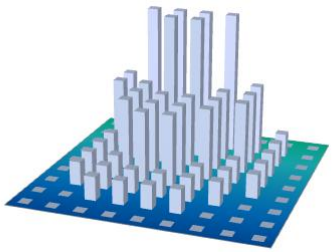


Figure 1: *Illustration of a beamlet intensity map.*

A common way to solve the inverse planning in IMRT optimization problems is to use a beamlet-based approach. This approach leads to a large-scale programming problem with thousands of variables and hundreds of thousands of constraints. Due to the complexity of the whole optimization problem, many times the treatment planning is divided into three smaller problems which can be solved separately: geometry problem, intensity problem, and realization problem. The geometry problem consists of finding the minimum number of beams and corresponding directions that satisfy the treatment goals using optimization algorithms (see, e.g., [1]). After deciding which beam angles should be used, a patient will be treated using an optimal plan obtained by solving the intensity problem - the problem of determining the optimal beamlet weights for the fixed beam angles. Many mathematical optimization models and algorithms have been proposed for the intensity problem, including linear models (e.g., [2]), mixed integer linear models (e.g., [3]), nonlinear models (e.g., [4]), and multiobjective models (e.g., [5]). After an acceptable set of intensity

Structure	Mean Dose	Max Dose	Prescribed Dose
Spinal cord	–	45 Gy	–
Brainstem	–	54 Gy	–
Left parotid	26 Gy	–	–
Right parotid	26 Gy	–	–
PTV left	–	–	59.4 Gy
PTV right	–	–	50.4 Gy
Body	–	70 Gy	–

Table 1: *Prescription dose for the target volumes and tolerance doses for the organs at risk.*

maps is produced, one must find a suitable way for delivery (realization problem). Typically, beamlet intensities are discretized over a range of values (0 to 7, e.g.) and one of the many existing techniques (see, e.g., [6]) is used to construct the apertures and intensities that approximately match the intensity maps previously determined.

Most of the research published relates to each of the above problems separately. However, there is the need to link well the three problems. While the linkage between the geometry problem and the intensity problem is straightforward, the linkage between the intensity problem and the realization problem is all but simple and may lead to significant deterioration of plan quality.

The outcome of the intensity problem is a set of optimal fluence maps (one for each fixed beam) that can be represented by real matrices whose entries correspond to each beamlet intensity. These matrices, solutions of the intensity problem, cannot be directly implemented, because of hardware constraints. The matrices have to be transformed to accommodate hardware settings, with a resulting degradation of the plan quality. The process of converting an optimal fluence map into a set of MLC segments is called segmentation. Segmentation needs to receive as input integer matrices, that are obtained by the discretization of each beamlet intensity over a range of values. This discretization, typically done by simple rounding of the optimized beamlets, is one of the main causes for deterioration of plan quality. This subject is poorly documented in literature (see [7], e.g.) and the general idea transmitted is that deterioration is mainly caused by segmentation issues. The lack of criteria to increase or reduce a beamlet intensity should be avoided since all optimization effort is jeopardized by doing so. Using a clinical example of a head and neck cancer case, numerical evidence of the resulting deterioration of plan quality is presented next.

2. ILLUSTRATION OF PLAN QUALITY DETERIORATION USING A HEAD & NECK CLINICAL EXAMPLE

A clinical example of a head and neck case is used to verify the deterioration caused by the rounding of the optimal fluence maps. In general, the head and neck region is a complex area to treat with radiotherapy due to the large number of sensitive organs in this region (e.g. eyes, mandible, larynx, oral cavity, etc.). For simplicity, in this study, the OARs used for treatment optimization were limited to the spinal cord, the brainstem and the parotid glands. The tumor to be treated plus some safety margins is called planning target volume (PTV). For the head and neck case in study it was separated in two parts: PTV left and PTV right (see Figure 2). The prescribed doses for all the structures considered in the optimization are presented in Table 1.

In order to facilitate convenient access, visualization and analysis of patient treatment planning data, the computational tools developed within Matlab [8] and CERR [9] (computational environment for radiotherapy research) were used as the main software platform

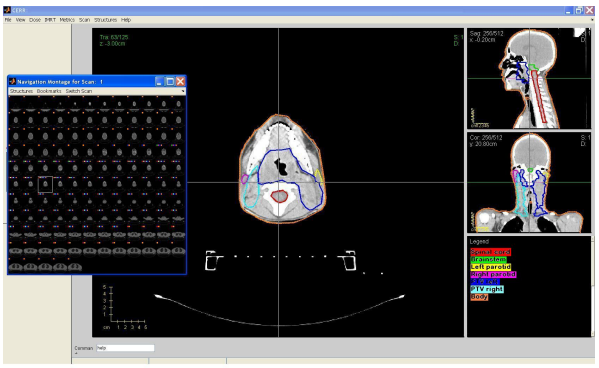


Figure 2: *Structures considered in the IMRT optimization visualized in CERR.*

Level	level intensity	beamlet intensity range
0	0.0000	[0.0000 ; 1.2857)
1	2.5714	[1.2857 ; 3.8571)
2	5.1429	[3.8571 ; 6.4286)
3	7.7143	[6.4286 ; 9.0000)
4	10.285	[9.0000 ; 11.571)
5	12.857	[11.571 ; 14.142)
6	15.428	[14.142 ; 16.714)
7	18.000	[16.714 ; 18.000]

Table 2: *Beamlet distribution to correspondent intensity level for 7 levels.*

to embody our optimization research and provide the necessary dosimetry data to perform optimization in IMRT.

A linear model was used to perform IMRT optimization on this case [7]. Our tests were performed on a 2.66Ghz Intel Core Duo PC with 3 GB RAM. We used CERR 3.2.2 version and MATLAB 7.4.0 (R2007a). The dose was computed using CERR's pencil beam algorithm (QIB) with seven equispaced beams in a coplanar arrangement, with angles 0^o, 51^o, 103^o, 154^o, 206^o, 257^o and 309^o, and with 0^o collimator angle. To address the linear problem we used one of the most effective commercial tools to solve large scale linear programs – Cplex[10]. We used a barrier algorithm (*baropt* solver of Cplex 10.0) to tackle our linear problem.

In order to acknowledge the degree of plan quality deterioration, results obtained for the optimal fluence maps were compared with the fluence maps obtained after rounding optimal intensities using 7 levels and 5 levels. In Tables 2 and 3 we have the beamlet intensity range for each intensity level. By decreasing the number of levels, the segmentation problem will be simplified resulting in more efficient delivery. However, by decreasing the number of levels the beamlet intensity range will increase, potentiating a more expressive deterioration of results. In the best case scenario for both levels there are no differences between the optimal intensities and the rounded intensities. However, for the worst case scenario, for 7 levels the difference between the optimal and the rounded intensity for each beamlet is 1.2857 and for 5 levels that difference is 1.8.

The quality of the results can be perceived considering a variety of metrics and can change from patient to patient. Typically, results are judged by their cumulative dose-volume histogram (DVH). The DVH displays the fraction of a structure's volume that receives at least a given dose. An ideal DVH for the tumor would present 100% volume for all dose values ranging from zero to the prescribed dose value and then drop immediately to zero, indicating that the whole target volume is treated exactly as prescribed. Ideally, the curves for the organs at risk would instead drop immediately to zero, meaning that no volume receives radiation.

Level	level intensity	beamlet intensity range
0	0.0000	[0.0000 ; 1.8000)
1	3.6000	[1.8000 ; 5.4000)
2	7.2000	[5.4000 ; 9.0000)
3	10.800	[9.0000 ; 12.600)
4	14.400	[12.600 ; 16.200)
5	18.000	[16.200 ; 18.000]

Table 3: *Beamlet distribution to correspondent intensity level for 5 levels.*

In Figure 3, DVH curves for tumor volumes and parotids are presented for optimal fluences obtained by the linear model and for the rounded optimal intensities when using 5 and 7 levels. DVH curves for OARs other than parotids only suffer residual changes with the rounding procedure. By simple inspection of Figure 3 we can observe the deterioration of the results from the transition of the optimal fluence maps to the rounded ones. That deterioration affect mostly the PTVs and is aggravated when fewer levels are considered, i.e., when faster delivery is aimed.

Another metric usually used considers prescribed dose that 95% of the volume of the PTV receives (D_{95}). Typically, 95% of the prescribed dose is required. D_{95} is represented in Figure 3 with an asterisk and we can observe that the rounded fluences fail to meet that quality criteria. Note that no segmentation was done and the observed deterioration is exclusively caused by the rounding of the optimal fluence maps.

3. COMBINATORIAL OPTIMIZATION APPROACH

After obtaining an optimal fluence map, and defining the number of levels of intensity to consider, we need to decide to which level of intensity each beamlet should be assigned to. The typical approach is to decide based on smaller distance and assign the level intensity closer to the optimal fluence (rounding). However, that decision criteria can put two beamlets with very close optimal intensities in distinct levels of intensity. Moreover, in such a complex large-scale optimization process, with such interdependence between beamlet intensity values, increasing or reducing the intensity of a beamlet should not be based on distance to closest intensity level. An alternative decision criteria is to decide between the two boundary levels of the optimal beamlet intensity, based on a dose-volume response, rather than on a distance criteria.

The combinatorial optimization problem of deciding, based on a dose-volume criteria, to which neighbor intensity level a beamlet intensity should be assigned to, can be stated as a binary optimization problem. Let x^{opt} denote the vector of the optimal beamlet intensities obtained in the end of the intensity problem. Let x^{round} denote the vector of the usual rounded intensities and let x^{trunc} denote the vector of the truncated intensities, i.e., the vector of the intensities corresponding to the smaller intensity value of the two neighbor level intensities. The difference of intensity levels between x^{round} and x^{trunc} is a binary vector, where each 1 represents a choice of an upper level of intensity, and each 0 represent a choice of an under intensity level. The combinatorial optimization problem can be stated as

$$\begin{aligned} \min\ & f(x) \\ & x \ \textit{binary}, \end{aligned}$$

where $f(x)$ is a penalty function of the distances between the DVH curves for x^{opt} and DVH curves for $x^{trunc} + x \times (\text{levels range})$.

This formulation originates a large combinatorial optimization problem. For the head and neck problem introduced in the previous section, the number of beamlets is 1613, which means we have $2^{1613} = 3.6423\text{E}485$ possibilities to consider. The magnitude of those numbers implies that both an exhaustive approach and an exact approach (branch and bound) are inviable.

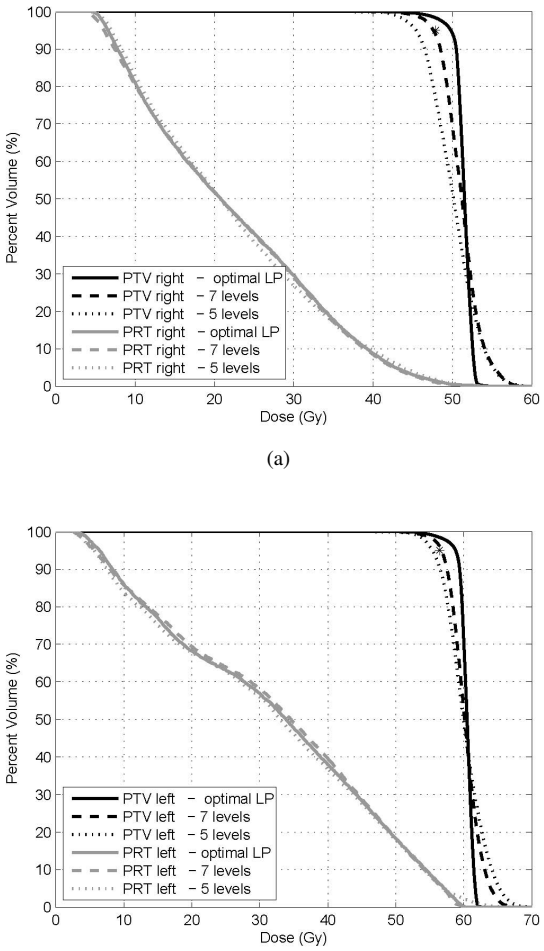
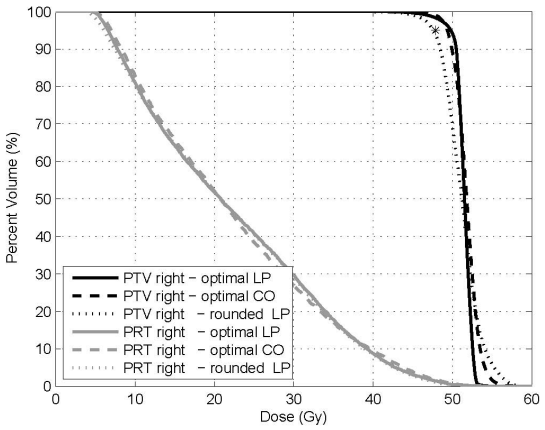


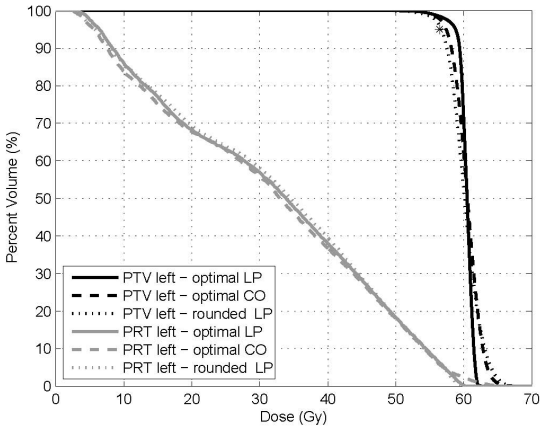
Figure 3: *DVH for ideal optimal fluence using LP vs 7 and 5 level for right PTV and PRT –* (a) *and DVH for rounded fluence using LP vs 7 and 5 level for left PTV and PRT–* (b).

There exists a number of heuristics to address successfully this problem. Here, we used a tailored version of binary genetic algorithms (using the Genetic Algorithm Optimization Toolbox of MATLAB).

In Figure 4, DVH curves for PTV's and parotids are presented for optimal fluences obtained by the linear model, for the rounded optimal fluences using 7 levels of intensity, and the fluences obtained by the resolution of the combinatorial optimization problem using the binary genetic algorithm. Again, DVH curves for OARs other than parotids only suffer residual changes. Even for parotids, DVH differences are not significative. However, looking at the DVH curves for PTV's we can see the benefit of the combinatorial optimization approach on the improvement of the rounded solution. That improvement is particularly notorious in Figure 4(a), for DVH curve of PTV right, since the DVH curve of the rounded LP fluences failed to meet the criteria of having 95% of the volume of the PTV receiving 95% of the prescribed dose. Not only the DVH curves of the optimal CO fluences for PTV right meet that criteria (DVH curve is over the asterisk) but they are almost as good as the DVH curves for the optimal LP fluences. The benefit of the combinatorial optimization approach on the improvement of the rounded solution is amplified when using less intensity levels.



(a)



(b)

Figure 4: *Cumulative dose volume histogram comparing the optimal beamlets obtained by the linear model (optimal LP), the rounded optimal beamlets using 7 levels (rounded LP), and the beamlets solution of the combinatorial optimization problem (optimal CO), for PTV right* – (a) *and PTV left* – (b).

4. CONCLUSION

A common way to solve the inverse planning in IMRT optimization problems is to use a beamlet-based approach. This approach leads to a large-scale programming problem, with thousands of variables and hundreds of thousands of constraints, and as a consequence, typically, the treatment planning is divided into three smaller problems which can be solved separately: geometry problem, intensity problem, and realization problem. That division has the consequence of causing a plan quality deterioration arising from the transition between the intensity problem and realization problem. Typically, after the optimal beamlet intensities are determined, they are discretized over a range of values using a distance criteria (rounding). However, that decision criteria is not appropriate and can lead to severe plan quality deterioration. Here, we propose an alternative decision criteria based on a dose-volume response. That criteria has physical meaning and originates a combinatorial optimization problem of deciding, based on a dose-volume criteria, to which intensity level a beamlet intensity should be assigned. A binary genetic algorithm was used to solve the combinatorial optimization problem. A head and neck clinical example was used to test the ability of the proposed formulation and resolution method to obtain improved plans compared to the usual rounding procedure. The results obtained did improve the rounded solution, with a clear increase of the plan quality. Although these results were obtained for a particular clinical example, and using a linear model to solve the intensity problem, we believe that the transition using this combinatorial approach can always improve the usual transition between the intensity problem and the realization problem, regardless the model used to solve the intensity problem and the clinical case at hand. Future work includes the development of tailored binary combinatorial algorithms, suited to tackle the problem at hand more efficiently, both in terms of final solution and in terms of computational time.

5. ACKNOWLEDGEMENTS

Support for this work was partly provided by the European Social Fund and MCTES under QREN and POPH programs.

6. REFERENCES

[1] H. Rocha, J. M. Dias, B. C. Ferreira, M. C. Lopes, "Direct search applied to beam angle optimization in radiotherapy design," Inescc Research Report 06/2010, ISSN: 1645–2631. Available at `http://www.inescc.pt/documentos/6_2010.PDF`.

[2] H. E. Romeijn, R. K. Ahuja, J. F. Dempsey, A. Kumar, J. Li, "A novel linear programming approach to fluence map optimization for intensity modulated radiation therapy treatment planing," *Phys. Med. Biol.*, vol. 48, pp. 3521–3542, 2003.

[3] E. K. Lee, T. Fox, I. Crocker, "Integer programing applied to intensity-modulated radiation therapy treatment planning," *Ann. Oper. Res.*, vol. 119, pp. 165–181, 2003.

[4] S. Spirou, C. -S. Chui, "A gradient inverse planning algoritm with dose-volume constraints," *Med. Phys.*, vol. 25, pp. 321–333, 1998.

[5] H. E. Romeijn, J. F. Dempsey, J. Li, "A unifying framework for multi-criteria fluence map optimization models," *Phys. Med. Biol.*, vol. 49, pp. 1991–2013, 2004.

[6] H. E. Romeijn, R. K. Ahuja, J. F. Dempsey, A. Kumar, "A column generation approach to radiation therapy treatment planning using aperture modulation," *SIAM J. Optim.*, vol. 15, pp. 838–862, 2005.

[7] H. Rocha, J. M. Dias, B. C. Ferreira, M. C. Lopes, "Towards efficient transition from optimized to delivery fluence maps in inverse planning of radiotherapy desing," Inescc Research Report 07/2010, ISSN: 1645–2631. Available at `http://www.inescc.pt/documentos/7_2010.PDF`.

[8] MATLAB, http://www.mathworks.com.

[9] J. O. Deasy, A. I. Blanco, V. H. Clark, "CERR: A Computational Environment for Radiotherapy Research," *Med. Phys.*, vol. 30, pp. 979–985, 2003.

[10] CPLEX, ILOG CPLEX, `http://www.ilog.com/products/cplex`.

Hybrid large neighborhood search for the dial-a-ride problem

Sophie N. Parragh [*] Verena Schmid [†]

[*] INESC Porto / IBM CAS Portugal
Rua Dr. Roberto Frias, 378, 4200-465 Porto (Portugal)
sophie.parragh@inescporto.pt

[†] Department of Business Administration, University of Vienna
Bruenner Strasse 72, 1210 Vienna (Austria)
verena.schmid@univie.ac.at

ABSTRACT

Demographic change towards an ever aging population entails an increasing demand for specialized transportation systems to compliment the traditional public means of transportation. Typically, users place transportation requests specifying a pickup and a drop off location and a fleet of minibuses or taxis is used to serve these requests. Those systems are usually referred to as demand responsive transportation systems. The underlying optimization problem can be modeled in terms of a dial-a-ride problem. In the dial-a-ride problem considered in this article, total routing costs are minimized while respecting time window, maximum user ride time, maximum route duration, and vehicle capacity restrictions. We propose a hybrid large neighborhood search algorithm and compare different hybridization strategies on a set of benchmark instances from the literature.

Keywords: Dial-a-ride, Large neighborhood search, Hybrid

1. INTRODUCTION

Demand responsive transportation services are requested, e.g. in remote rural areas, where no general public transportation systems exist, as a complementary service to available public transportation systems for the elderly or disabled, or in the area of patient transportation to and from hospitals or other medical facilities. All these services involve the transportation of persons who place transportation requests, specifying an origin and a destination location. The underlying optimization problem is usually modeled in terms of a dial-a-ride problem (DARP). The field of dial-a-ride problems has received considerable attention in the literature. However, due to the application oriented character of this problem, the objectives considered as well as the constraints imposed vary considerably. Rather recent surveys covering dial-a-ride problems and demand responsive transportation are due to Cordeau and Laporte [1] and Parragh et al. [2].

In the DARP considered in this article, the objective corresponds to the minimization of the total routing costs. A homogeneous fleet of vehicles of size m has to serve a given set of transportation requests n. These are all known in advance of the planning. In the following we will refer to the origin or pickup location of a request i by i, and to its destination or drop off location by $n+i$. Users specify time windows for either the origin or the destination. In a addition, maximum user ride times, route duration limits, and vehicle capacity constraints have to be considered in the planning. This version of the DARP has been considered by Cordeau and Laporte [3], who propose a tabu search heuristic and a set of 20 benchmark instances, and by Parragh et al. [4], who develop a competitive variable neighborhood search heuristic. A formal definition of the problem can be found in [5], where a branch-a-cut algorithm is proposed that solves instances with up to 36 requests.

In recent years, the field of hybrid metaheuristics, and matheuristics in particular, has received more and more attention [6, 7]. In the field of vehicle routing, metaheuristic and column generation hybrids have shown to be especially successful: Prescott-Gagnon et al. [8] propose a branch-and-price based large neighborhood search algorithm for the vehicle routing problem with time windows; heuristic destroy operators are complimented by a branch-and-price based repair algorithm. Muter et al. [9], on the other hand, propose a hybrid tabu search heuristic, where the column pool is filled with feasible routes identified by the tabu search. The search is then guided by the current best lower and upper bound; the current best lower bound is obtained from solving the linear relaxation of a set covering type formulation on the current column pool; the current best upper bound is computed by imposing integrality on the decision variables. The resulting method is also tested on benchmark instances for the vehicle routing problem with time windows.

Given its success for several vehicle routing problems [10], and the pickup and delivery problem with time windows in particular [11], we investigate and compare different hybridization strategies of large neighborhood search (LNS) and column generation (CG) in the context of the DARP.

2. SOLUTION METHOD

In the following we first describe LNS and CG. Thereafter, we introduce different hybridization schemes.

2.1. Large neighborhood search

LNS has been introduced by Shaw [12]. Its principle is relatively simple: in each iteration the incumbent solution is partially destroyed and then it is repaired again; that is, first a given number of elements are removed and these elements are then reinserted. Every time these operations lead to an improved solution, the new solution replaces the incumbent solution, otherwise it is discarded.

Ropke and Pisinger [11] propose to use a number of different destroy and repair operators; in this article, we use the following (they are all based on [11]): random removal, worst removal, related removal, greedy insertion, and k-regret insertion.

Before a removal operator is applied to the incumbent solution, the number of requests to be removed q has to be determined. In our case, in each iteration, q is chosen randomly between $0.1n$ and $0.5n$. Then, one of the destroy operators is randomly selected. The random removal operator randomly removes q re-

quests from the incumbent solution. The worst removal operator randomly removes requests while biasing the selection towards requests whose removal would improve the objective function value the most. Finally, the related removal operator removes related requests. Two requests i and j are said to be related if $(|B_i - B_j| + |B_{n+i} - B_{n+j}| + t_{ij} + t_{n+i,n+j})$ is small; t_{ij} denotes the distance between location i and j; and B_i the beginning of service at i.

In a next step, a solution that has been partially destroyed is repaired again. We randomly choose a repair operator among greedy insertion, 2-regret insertion, 3-regret insertion, 4-regret insertion and m-regret insertion; see [11] for further details.

In order to further diversify the search we allow solutions that deteriorate the incumbent solution by at most 3% to be accepted with a probability of 1%. In order to facilitate switching between LNS and exact components, in a first step, we decided to refrain from using more sophisticated acceptance schemes.

Furthermore, following the findings of [11], in each iteration, we randomly choose if the selected repair operator is used in its deterministic or in its randomized version. If the randomized version is selected, every time the evaluation function is called, it randomly chooses a noise factor in $[0.5, 1.5]$ and multiplies the original insertion costs by it.

Finally, like in [4], every time a new solution is generated and it is at most 5% worse than the current best solution, the new solution undergoes local search based improvement; we refer to [4] for details on this procedure.

2.2. Column generation

In order to use column generation based components, we formulate the DARP in terms of a set covering problem. Let Ω denote the set of feasible routes and let P denote the set of requests. The parameter m denotes the number of available vehicles. For each route $\omega \in \Omega$, let c_ω be the cost of the route and let the constant $b_{i\omega}$ represent the number of times vertex $i \in P$ is traversed by ω. Binary variable y_ω takes value 1 if and only if route ω is used in the solution. The problem can thus be formulated as the following set-covering problem (SCP):

$$\min \sum_{\omega \in \Omega} c_\omega y_\omega \tag{1}$$

subject to

$$\sum_{\omega \in \Omega} b_{i\omega} y_\omega \geq 1 \qquad \forall i \in P, \tag{2}$$

$$\sum_{\omega \in \Omega} y_\omega \leq m, \tag{3}$$

$$y_\omega \in \{0,1\} \qquad \forall \omega \in \Omega. \tag{4}$$

Replacing (4) by,

$$y_\omega \geq 0 \qquad \forall \omega \in \Omega, \tag{5}$$

we obtain the linear relaxation of SCP denoted as LSCP.

Due to the large size of Ω, LSCP cannot be solved directly. Instead, a restricted version of LSCP, denoted as RLSCP, considering only a small subset of columns $\Omega' \subset \Omega$, is solved. Usually, the set Ω' is generated using column generation. In column generation, in each iteration, the column or route that is associated with the smallest negative reduced cost value is searched. The according problem is usually referred to as the subproblem whereas RLSCP is denoted as the master problem. In our case, the reduced cost of a given column is computed as follows:

$$\bar{c}_\omega = c_\omega - \sum_{i \in P} b_{i\omega} \pi_i - \sigma, \tag{6}$$

where π_i denotes the dual variable associated with constraint (2) for index i, and σ the dual variable associated with constraint (3). The solution of the master and the subproblem are iterated until no more negative reduced cost columns can be found. In this case, the optimal solution of LSCP has been found. The column generation concept can also be exploited in a heuristic way. In the following, we describe how we intend to use it.

2.3. Hybridization schemes

We investigate three hybridization schemes. In all schemes, all feasible routes identified by LNS are added to the common column pool Ω'. We apply the following column pool management. In the case where a column already exists, the new column replaces the old one if the new column is associated with lower routing costs; otherwise, the old column is kept and the new column is discarded.

In the first hybridization scheme (denoted as $h1$), in addition to the above described destroy and repair operators, a destroy and a repair operator taking into account dual information are introduced. Before executing either of the two operators, the RLSCP is solved on the current column pool. The destroy operator works in a similar way as the worst removal operator; but instead of the difference in cost, selection is biased towards requests with a high π_i value. The repair operator also uses this idea. It sequentially inserts all currently not routed requests ordered with respect to their π_i values, at the the best possible position.

In the second hybridization scheme (denoted as $h2$) we follow ideas of [13]. Every 1000 iterations we interrupt the LNS and we solve RSCP, that is the restricted integer set covering problem, on the current column pool. Since we solve a set covering problem and not a set partitioning problem, requests might appear on more than only one route. In this case, duplicated requests are sequentially removed in a greedy way and LNS resumes the search from this solution.

The third hybridization scheme (denoted as $h3$) is always combined with $h2$. Here we propose to use additional heuristic column generators that take into account dual information to populate the column pool. These column generators are called in the same frequency as RSCP. Since they use dual information in order to find routes of negative reduced cost, RLSCP has to be solved as well. We use several different column generators. The simplest one works in a similar way as the new repair operator and generates a new route from scratch. A second more sophisticated generator uses ideas from variable neighborhood search [14]. It considers one route at a time. The size of a neighborhood is defined as the percentage share of requests that is removed, inserted from, or swapped with requests currently not on this route. It starts from each route part of the current optimal solution of RLSCP; in addition, it considers the new route that was generated from scratch, an empty route, and a randomly generated route. The third column generator uses ideas from LNS and works in a similar way as the LNS based column generator introduced in [15]. It is only called if the first two column generators are not able to find columns of negative reduced cost.

In Algorithm 1, we outline the LNS and the different hybridization schemes.

3. PRELIMINARY RESULTS

The algorithm was implemented in C++ and for the solution of the set covering problems CPLEX 12.1 together with Concert Technology 2.9 was used. All tests were carried out on a Xeon CPU at

Algorithm 1: LNS hybridization schemes

```
generate a feasible starting solution s
s_best := s
initialize column pool Ω'
repeat
    randomly choose a destroy H_d and a repair heuristic H_r
    [h1:also choose from heuristics based on dual information]
    apply first H_d and then H_r to s yielding s'
    add columns to Ω'
    if s' is better than s_best then
        s_best := s'
        s := s'
    else if s' meets the acceptance criteria then
        set s := s'
    end if
    [h3: every 1000 iterations solve RLSCP on Ω'
    generate columns by heuristic column generators
    add columns to Ω']
    [h2: every 1000 iterations solve RSCP on Ω' yielding s*
    s_best := s*
    s := s*]
until some stopping criterion is met
return s_best
```

2.67 GHz with 24GB of RAM (shared with 7 other CPUs). In the following we describe the test data set and then the results obtained. Note that the results reported in this article, although promising, are of preliminary nature. Additional parameter tuning tests and improvements of the implementation are still necessary.

3.1. Test instances

Cordeau and Laporte [3] proposed a data set of 20 randomly generated instances. They contain 24 to 144 requests. In each instance, the first $n/2$ requests have a time window on the destination, while the remaining $n/2$ instances have a time window on the origin. For each vertex a service time $d_i = 10$ was set and the number of persons transported per request was set to 1. Routing costs and travel times from a vertex i to a vertex j are equal to the Euclidean distance between these two vertices. The instances are split into two parts. In the first 10 instances, narrow time windows are considered. For the second 10 instances, wider time windows are given. The route duration limit was set to 480, the vehicle capacity to 6, and the maximum ride time to 90, in all instances.

3.2. Comparison of the different hybridization schemes

We test all of the different hybridization strategies and benchmark them against pure LNS as described above. In all experiments, LNS is run for 25000 iterations. In setting $LNS+h1$, in addition to the heuristic destroy and repair operators, the dual information based ones are used. In setting $LNS+h2$, every 1000 iterations RSCP is solved. In setting $LNS+h1+h2$, both of the above hybridization schemes are used and in setting $LNS+h1+h2+h3$, all three hybridization schemes are employed.

In Table 1, the different results on a per instance level are given. The best values per column and instance are marked in bold. The results displayed are average values over five random runs per instance. Comparing LNS and $LNS+h1$, it is not clear if the dual information based operators, proposed in $h1$, are contributing to the search in a positive way; purely heuristic LNS seems to be the better option. Hybridization scheme $h2$, on the other hand, definitely has a positive impact on the overall performance; it obtains most of the per instance average best results. On a total average level, it ties with $LNS+h1+h2$. This seems to indicate that in combination with $h2$, $h1$ does not have a negative impact on the overall performance of the method. It even has a slightly positive impact in the case of the largest instances of the first half of the data set. Last but not least, we also tested a combination of all three hybridization schemes. In the table it is denoted as $LNS+h1+h2+h3$. The total average value associated with this method is comparable to the total average values of $LNS+h2$ and $LNS+h1+h2$. On a per instance level, it obtains the best per instance average results for the largest instances in the second part of the data set.

In comparison to the variable neighborhood search proposed in [4], both solution quality as well as run times are comparable. The deviations of the latter three settings from the average results of the variable neighborhood search are less then 0.5% on average. So far, we were able to improve one best known solution.

4. CONCLUSIONS AND OUTLOOK

As noted above, the presented results are of very preliminary character. They do, however, indicate that the different hybridization schemes have a positive impact on the overall performance of the solution method. At the moment it seems as if hybridization scheme $h2$ increases the performance the most; but also the other two schemes show positive potential.

Further tests with different parameter settings are still needed in order to fully understand the interplay between the heuristic and the different column generation based components. In addition, an adaptive layer as in [11] and a simulated annealing [16] based acceptance scheme should be incorporated into the LNS, to further improve its performance as a stand alone method. Finally, using ideas of [9], we also plan to investigate different guiding mechanisms that allow the search to switch between the different components as needed.

5. ACKNOWLEDGMENTS

We wish to thank the Austrian Research Promotion Agency (FFG, grant #826151, program IV2Splus) for sponsoring this work.

6. REFERENCES

[1] J.-F. Cordeau and G. Laporte, "The dial-a-ride problem: Models and algorithms," *Ann Oper Res*, vol. 153, pp. 29–46, 2007.

[2] S. N. Parragh, K. F. Doerner, and R. F. Hartl, "Demand responsive transportation," *Wiley Encyclopedia of Operations Research and the Management Sciences*, to appear.

[3] J.-F. Cordeau and G. Laporte, "A tabu search heuristic for the static multi-vehicle dial-a-ride problem." *Transport Res B-Meth*, vol. 37, pp. 579–594, 2003.

[4] S. N. Parragh, K. F. Doerner, and R. F. Hartl, "Variable neighborhood search for the dial-a-ride problem," *Comput Oper Res*, vol. 37, pp. 1129–1138, 2010.

[5] J.-F. Cordeau, "A branch-and-cut algorithm for the dial-a-ride problem." *Oper Res*, vol. 54, pp. 573–586, 2006.

[6] G. R. Raidl, J. Puchinger, and C. Blum, "Metaheuristic hybrids," in *Handbook of Metaheuristics, 2nd edition*, M. Gendreau and J. Y. Potvin, Eds. Springer, 2010, pp. 469–496.

[7] C. Blum, M. J. Blesa Aguilera, A. Roli, and M. Sampels, Eds., *Hybrid Metaheuristics*, ser. Studies in Computational Intelligence, vol. 114. Berlin: Springer, 2008.

	m	n	LNS	$LNS+h1$	$LNS+h2$	$LNS+h1+h2$	$LNS+h1+h2+h3$
R1a	3	24	**190.02**	**190.02**	**190.02**	**190.02**	**190.02**
R2a	5	48	303.59	304.64	303.34	303.92	**302.50**
R3a	7	72	544.88	542.84	**537.89**	538.96	540.32
R4a	9	96	584.02	590.71	**573.78**	582.48	585.35
R5a	11	120	651.58	650.67	**639.74**	643.74	641.36
R6a	13	144	814.64	823.15	813.30	**801.02**	804.66
R7a	4	36	294.38	295.56	**294.02**	294.46	294.37
R8a	6	72	510.32	505.35	**496.73**	499.72	497.36
R9a	8	108	695.68	695.34	679.58	**677.98**	682.97
R10a	10	144	896.93	905.36	883.48	**881.50**	885.09
R1b	3	24	166.58	166.19	165.79	166.19	**165.52**
R2b	5	48	302.51	303.58	299.16	**296.71**	299.61
R3b	7	72	498.85	501.18	**492.78**	493.41	495.70
R4b	9	96	553.73	553.83	**539.31**	543.97	541.97
R5b	11	120	598.52	603.59	591.59	591.97	**589.77**
R6b	13	144	764.67	768.47	752.40	750.61	**749.65**
R7b	4	36	249.06	249.58	**248.72**	248.72	248.72
R8b	6	72	474.68	475.40	473.36	**469.93**	469.99
R9b	8	108	627.53	623.69	**612.20**	616.62	614.67
R10b	10	144	833.88	833.56	821.44	816.14	**813.42**
Avg			527.80	529.13	520.43	520.40	520.65

Table 1: Comparison of the different hybridization schemes

[8] E. Prescott-Gagnon, G. Desaulniers, and L.-M. Rousseau, "A branch-and-price-based large neighborhood search algorithm for the vehicle routing problem with time windows," *Networks*, vol. 54, no. 4, pp. 190–204, 2009.

[9] I. Muter, S. I. Birbil, and G. Sahin, "Combination of metaheuristic and exact algorithms for solving set covering-type optimization problems," *INFORMS J Comput*, 2010, published online before print March 23, 2010, DOI: 10.1287/ijoc.1090.0376.

[10] D. Pisinger and S. Ropke, "Large neighborhood search," in *Handbook of Metaheuristics, 2nd edition*, M. Gendreau and J.-Y. Potvin, Eds. Springer, 2010, pp. 399–419.

[11] S. Ropke and D. Pisinger, "An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows." *Transport Sci*, vol. 40, pp. 455–472, 2006.

[12] P. Shaw, "Using constraint programming and local search methods to solve vehicle routing problems." in *Proceedings CP-98 (Fourth International Conference on Principles and Practice of Constraint Programming)*, 1998.

[13] S. Pirkwieser and G. Raidl, "Boosting a variable neighborhood search for the periodic vehicle routing problem with time windows by ILP techniques," in *Preprints of the 8th Metaheuristic International Conference (MIC 2009), Hamburg, Germany*, 2009.

[14] N. Mladenovic and P. Hansen, "Variable neighborhood search." *Comput Oper Res*, vol. 24, pp. 1097–1100, 1997.

[15] S. Ropke and J.-F. Cordeau, "Branch-and-cut-and-price for the pickup and delivery problem with time windows," *Transport Sci*, vol. 43, pp. 267–286, 2009.

[16] S. Kirkpatrick, C. D. Gelatt Jr., and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, pp. 671–680, 1983.

An integer programming approach for elective surgery scheduling in a Lisbon hospital

Inês Marques * † Maria Eugénia Captivo * ‡ Margarida Vaz Pato * §

* Centro de Investigação Operacional, Faculdade de Ciências, Universidade de Lisboa
Bloco C6, Piso 4, sala 6.4.16, Campo Grande, 1749-016 Lisboa, Portugal

† Universidade Lusófona de Humanidades e Tecnologias
FCTS/FEG, Campo Grande, 376, 1749-024 Lisboa, Portugal
ines.marques@fc.ul.pt

‡ Universidade de Lisboa, Faculdade de Ciências
DEIO, Edifício C6 - Piso 4, Campo Grande, 1749-016 Lisboa, Portugal
mecaptivo@fc.ul.pt

§ Instituto Superior de Economia e Gestão, Universidade Técnica de Lisboa
Dept. Matemática, ISEG, Rua do Quelhas, 6, 1200-781 Lisboa, Portugal
mpato@iseg.utl.pt

ABSTRACT

Elective surgery planning is an important problem for any hospital. In particular, in Portugal, this problem reaches a level of great importance as it has direct relation with an efficient use of the operating theater, which also results on reducing waiting lists for surgery. Thus, a better surgical suite planning has economic and social impact. Both outcomes appear as guidelines of the Portuguese National Health Plan for 2004-2010. The authors present an integer linear programming model approach developed to address the elective surgery planning problem of a hospital in Lisbon, as well as results obtained with real data from the hospital. The results are analyzed in view of the impact on productivity indicators of the surgical suite and, as a consequence, on the hospital's waiting list for surgery.

Keywords: Health Care, Operating rooms, Elective case scheduling, Integer Programming

1. INTRODUCTION

The health sector has been progressively affected by increasingly restrictive budgets that not only call for an urgent need to promote a resource rationalization practice among hospitals but, above all, the demand for greater efficiency in the use of resources and in the performance of each service. The surgical suite is widely regarded as hospital's central engine as it has a direct impact in many other hospital departments, such as surgical wards and recovery units [1, 2]. As such, it is deemed a priority to improve the efficiency of this component. On the other hand, improvement of the surgical suite's efficiency may lead to increased productivity, in terms of the number of surgeries undertaken, thus contributing to a reduction in surgery waiting lists. Costs involved in keeping a patient on the waiting list for surgery are high, at both prevention and maintenance levels, even more so as considering the user's quality of live. In addition, according to Portugal's General Direction of Health [3], reducing surgery waiting list is one of the priorities of the National Health Service (SNS). Cutting down waiting lists for surgery is beneficial in many respects, at human and scientific, as well as economic levels.

This work focuses on a general, central and university hospital in Lisbon, incorporated within the Portuguese National Health Service. It has no maternity or outpatient emergency service and performs about 5 000 surgeries per year. The hospital has five surgical specialties. Its surgical suite has six operating theaters, one of which is reserved for ambulatory surgeries. Although all rooms in the surgical suite are equipped with the same basic equipment, the practice of this hospital is to daily assign the rooms for conventional surgeries to surgical specialties. Between two surgeries performed in the same room, cleaning and disinfecting protocols, performed by auxiliary staff and taking about 30 minutes must take place. Each operating room has a fixed and permanent nursing team assigned throughout the surgical suite's regular time. Each patient is assigned to a surgeon at waiting list booking time and, therefore, when planning, patient and surgeons are already assigned. Currently, the surgical suite's regular work schedule is between 8 am and 8 pm, from Monday to Friday. Surgery planning is performed on a weekly base and is finalized on Friday for the following week. The problem considers daily and weekly operating time limits for each surgeon, different priorities related to the surgeries and surgeons' unavailability.

In the literature, there are other integer programming approaches to surgeries' scheduling [4, 5, 6, 7, 8] and some heuristic approaches to the same problem [9, 10]. The specificities of the different cases under study are the most relevant factor contributing to the diversity of each work in this area.

2. MATHEMATICAL MODEL

The problem described in the previous section consists of scheduling elective surgeries for a day, a room and a starting time period, with a weekly planning horizon. Since surgeries are non-preemptive jobs, starting time variables were considered in formulating the problem [11]. Thus, the decision variables used in the model are: $x_{srtd} = 1$, if surgery s starts at the beginning of period t on day d in room r. Additional variables were also considered to register, on a daily basis, the surgical specialty assigned to each operating room: $y_{jrd} = 1$, if a surgery of specialty j starts in room r on day d.

In response to the urgent need of improving efficiency in resources utilization of the operating theater, the model objective function maximizes surgical suite occupation. The model constraints reflect the structure of the problem as presented in the previous section. There are constraints forcing the higher priority surgeries to be scheduled on Monday. Other constraints oblige surgeries of the following level of priority to be scheduled during the planning week, and the remaining surgeries may be scheduled or not during the planning week. There are constraints assuring that different surgeries do not overlap in the same room. These constraints also impose empty periods for room cleaning at the end of each surgery. An additional set of constraints provides the possibility to consider surgeons' or patients' unavailability periods. Constraints preventing assignment of more than one surgery specialty to each room and day are also included. Therefore, it is not permitted to exchange surgery specialty in the room during the day. It is also ensured that surgeons do not overlap between rooms in the same time period and day. In the real situation of the hospital involved, surgeons may exchange operating rooms. On the one hand, this exchange is feasible as the rooms are physically side by side. On the other hand, permission to exchange operating rooms by surgeons allows them to work in another operating room during hygiene periods in the previous room (about 30 minutes idle). Daily and weekly operating time limits for each surgeon are also considered.

If the hospital objective was to reduce the waiting list for surgery, the same set of constraints could be used. Only the objective function should be changed to the maximization of the number of surgeries planned.

The model objective function and constraints are all linear, thus resulting in a binary integer programming model.

3. SOLVING APPROACH

The problem is highly complex and attains a large dimension in hospital real instances [12]. Hence, the elective surgeries' scheduling problem was decomposed into two hierarchical phases according to the nature of surgeries: conventional surgeries are planned in the first phase and ambulatory surgeries are planned in the second phase. The first planning phase generates a high dimension problem, while the second one is of rather reduced dimension. The output of the conventional planning phase is included as input for the ambulatory surgery planning to ensure feasibility of the whole week's planning due to common resources (surgeons) between the two planning phases.

In each planning phase, an integer linear programming solver is used with limited time. If solver times out without optimality, the best feasible integer solution obtained is improved using a simple improvement heuristic.

4. COMPUTATIONAL RESULTS

The solution approach was tested with real data from the hospital, for seven planning weeks of 2007. The binary integer programming models were solved using CPLEX 11.0 with CONCERT 2.5 [13, 14]. The improvement heuristic was coded in C++ language. All tests were performed in a Core2 Duo, 2.53 GHz computer with 4GB of RAM. Time limit to run the model with Cplex was set to 30 000 seconds.

The proposed approach originated a valid surgical plan for all tested weeks, producing a potential surgical suite occupation rate in regular time superior to 75%, well above the corresponding rate currently obtained in hospital planning (under 40%). Furthermore, this approach also achieves to improve the waiting list reduction rate of the hospital surgical plans, which shows that the hospital surgical plans are clearly dominated by the corresponding proposed surgical plans with respect to these two conflicting criteria.

A similar approach can be used if we consider the objective to be the waiting list reduction. In this case, we also improve the waiting list reduction rate (above 11%) of the hospital surgical plans (under 6%), with a potential surgical suite occupation rate in regular time superior to 64%.

The previous results clearly show that the two criteria considered are conflituous. In the first case, longer surgeries are planned (with less cleaning periods) contrary to what happens in the second case where shorter surgeries are chosen.

Detailed results will be presented and discussed at the talk.

5. FINAL REMARKS

The approach employed allowed the authors to obtain an operating plan for each one of the seven weeks tested. The operating plans obtained are feasible and meet the necessary requirements imposed by the hospital in question. This approach enables the hospital surgical suite to be more efficiently used, thus achieving the purpose of the study undertaken and responding to the hospital management's interest. Moreover, the methodologies developed have also an impact on reducing the waiting list for surgery.

6. ACKNOWLEDGEMENTS

The authors would like to thank Dr. Manuel Delgado for his enthusiastic interest in this work, Dra. Margarida Baltazar for her patient provision of all necessary data and nurse Fátima Menezes for the friendly and patient description of the entire process.

This research is partially supported by Portuguese Foundation for Science and Technology (FCT) under project POCTI/ISFL-1/152.

7. REFERENCES

[1] Health Care Financial Management Association, "Achieving operating room efficiency through process integration," Health Care Financial Management Association, Tech. Rep., 2005.

[2] E. Litvak and M. C. Long, "Cost and quality under managed care: irreconcilable differences?" *The American Journal of Managed Care*, vol. 6, no. 3, pp. 305–312, 2000.

[3] General Direction of Health, Ed., *Plano Nacional de Saúde 2004-2010: mais saúde para todos*. Lisbon: General Direction of Health, 2004, (in Portuguese).

[4] B. Cardoen, E. Demeulemeester, and J. Beliën, "Optimizing a multiple objective surgical case sequencing problem," *International Journal of Production Economics*, vol. 119, no. 2, pp. 354–366, 2009.

[5] ——, "Sequencing surgical cases in a day-care environment: an exact branch-and-price approach," *Computers & Operations Research*, vol. 36, no. 9, pp. 2660–2669, 2009.

[6] R. Velásquez and M. T. Melo, "A set packing approach for scheduling elective surgical procedures," in *Operations Research Proceedings 2005*. Springer Berlin Heidelberg, 2006, pp. 425–430.

[7] H. Fei, C. Chu, and N. Meskens, "Solving a tactical operating room planning problem by a column-generation-based heuristic procedure with four criteria," *Annals of Operations Research*, vol. 166, no. 1, pp. 91–108, 2009.

[8] E. Marcon, S. Kharraja, and G. Simonnet, "The operating theatre planning by the follow-up of the risk of no realization," *International Journal of Production Economics*, vol. 85, no. 1, pp. 83–90, 2003.

[9] E. Hans, G. Wullink, M. van Houdenhoven, and G. Kazemier, "Robust surgery loading," *European Journal of Operational Research*, vol. 185, no. 3, pp. 1038–1050, 2008.

[10] M. van der Lans, E. W. Hans, J. L. Hurink, G. Wullink, M. van Houdenhoven, and G. Kazemier, "Anticipating urgent surgery in operating room departments," University of Twente, Tech. Rep. WP-158, 2006.

[11] J. P. Sousa and L. A. Wolsey, "A time indexed formulation of non-preemptive single machine scheduling problems," *Mathematical Programming*, vol. 54, pp. 353–367, 1992.

[12] I. Marques, "Planeamento de cirurgias electivas - abordagens em programação inteira," Ph.D. dissertation, Faculdade de CiÍncias, Universidade de Lisboa, 2010.

[13] *ILOG CPLEX 11.0 User's Manual*, ILOG, Incline Village, Nevada, 2007.

[14] *ILOG CONCERT 2.0 User's Manual*, ILOG, Incline Village, Nevada, 2004.

Tackling Freshness in Supply Chain Planning of Perishable Products

Pedro Amorim * Hans-Otto Günther † Bernardo Almada-Lobo *

* DEIG, Faculty of Engineering, University of Porto
Rua Dr. Roberto Frias, s/n, 4600-001 Porto, Portugal
{amorim.pedro, almada.lobo}@fe.up.pt

† Department of Production Management, Technical University of Berlin
Strasse des 17. Juni 135, 10623
hans-otto.guenther@tu-berlin.de

ABSTRACT

Integrated production and distribution planning has received a lot of attention throughout the years and its economic advantages over a decoupled approach is well documented. However, for highly perishable products this integrated approach has to include, further than the economic aspects, the intangible value of customers' willingness to pay, which is related to product freshness. Hence, in this work we explore, through a multi-objective framework, the potential advantages of integrating these two intertwined planning problems at an operational level for this kind of products. We formulate integrated and decoupled models for the case where perishable goods have a fixed and a loose shelf-life in order to test our hypothesis. An illustrative example is used to interpret the models and the results show that the economic benefits derived from using an integrated approach are much dependent on the freshness level of products delivered that the planner is aiming at as well as on the type and degree of perishability the product is subject to.

Keywords: Suppy chain planning, Multi-objective, Perishability

1. INTRODUCTION

Rapidly deteriorating perishable goods, such as fruits, vegetables, yoghurt and fresh milk, have to take into account the perishability phenomenon even for the operational level of production and distribution planning, which has a timespan ranging from one week to one month. Usually these products start deteriorating from the moment they are produced on. Therefore, without proper care, inventories may rapidly get spoiled before their final use making the stakeholders incur on avoidable costs. The customers of these products are aware of the intense perishability they are subject to, and they attribute an intangible value to the relative freshness of the goods [1]. To evaluate freshness customers rely on visual cues which may differ among the broad class of perishable products. Nahmias [2] dichotomized deteriorating goods in two categories according to their shelf-life: (1) fixed lifetime: items' lifetime is pre-specified and therefore the impact of the deteriorating factors is taken into account when fixing it. In fact, the utility of these items may decrease during its lifetime, and when passing its lifetime, the item will perish completely and become of no value, e.g., milk, inventory in a blood bank, and yoghurt, etc. (2) random lifetime: there is no specified lifetime for these items. The lifetime for these items is assumed as a random variable, and its probability distribution may take on various forms. Examples of items that keep deteriorating with some probability distribution are electronic components, chemicals, and vegetables, etc.

When the shelf-life is fixed the most common visual cue that customers rely on is the best-before-date (BBD). The BBD can be defined as the end of the period, under any stated storage conditions, during which the product will remain fully marketable and retain any specific qualities for which tacit or express claims have been made. In this case, customers will adapt their willingness to pay for a product based on how far away the BBD is. On the other hand, when the expiry date of a product is not printed and then the shelf-life is loose, customers have to rely on their senses or external sources of information to estimate the remaining shelf-life of the good. For example, if a banana has black spots or if flowers look wilted, then customers know that these products will be spoiled rather soon.

In the case of loose shelf-life, especially in the fresh food industry, manufacturers can make use of predictive microbiology to estimate the shelf-life of these kind of products based on external controllable factors, such as humidity and temperature [3]. To make concepts clearer, shelf-life is defined as the time period for the product to become of no value for the customer due to the lack of the tacit initial characteristics that the product is supposed to have. Thus, in our case, this period starts on the day the product is produced. The determination of shelf-life as a function of variable environmental conditions has been the focus of many research activities in this field and a considerable number of reliable models exist, such as the Arrhenius model, the Davey model and the square-root model. These models take into account the knowledge about microbial growth in decaying food goods under different temperature and humidity conditions.

Regarding production and distribution planning, many authors have shown the economic advantages of using an integrated decision model over a decoupled approach [4, 5]. These advantages are believed to be leveraged when the product suffers a rapid deterioration process and hence pushes towards a more connected view of these intertwined problems. For perishable goods the final products inventory that is usually used to buffer and decouple these two planning decisions have to be questioned since customers distinguish between different degrees of freshness and there is an actual risk of spoilage. In this work, we want to study the potential advantages of using an integrated approach for operational production and distribution planning of perishable goods compared with a decoupled one. These advantages will be analysed through the economic and product freshness perspective. We focus on highly perishable consumer goods industries, with a special emphasis on food processing, which have to cope with complex challenges, such as the integration of lot sizing and scheduling, the definition of setup families considering major and minor setup times and costs, and multiple non-identical production lines [6]. We are also interested in understanding if these potential advantages differ among the two distinct perishable goods classes that we have mentioned before: with fixed shelf-life and with loose shelf-life. For both cases, since we are interested in rapidly deteriorating goods,

we consider a customer who prefers products with a higher freshness level. To tackle explicitly this customer satisfaction issue we embedded our integrated operational production and distribution planning problem in a multi-objective framework distinguishing two very different and conflicting objectives of the planner. The first objective is concerned with minimizing the total costs over the supply chain covering transportation, production, setup and spoilage costs. The second one aims at maximizing the freshness of the products delivered to distribution centres and, therefore, maximize customers' willingness to pay [7].

2. PROBLEM STATEMENT

The production and distribution planning problem considered in this paper consists of a number of plants having dedicated lines which produce multiple perishable items with a limited capacity to be delivered to distribution centres. It is relevant to understand the importance of the design choice of having such a complex supply chain instead of just considering one plant and multiple distribution centres. As said before, we focus on perishable consumer goods industries which are known for demanding increasing flexibility in the supply chain planning processes. Thus, to consider a network of production plants which can add increased flexibility and reliability to hedge against the complex dynamics of such industries is crucial. Therefore, although we are tackling an operational level of decision making for these two planning tasks we assume a central organizational unit which makes decisions that are followed directly at a local level. The length of the planning horizon for such planning problem ranges from one week to one month.

All product variants belonging to the same family form a block. Therefore a product can only be assigned to one block. Blocks are to be scheduled on parallel production lines over a finite planning horizon consisting of macro-periods with a given length. The scheduling takes into account that the setup time and cost between blocks is dependent on the sequence of production (major setup). The sequence of products in a block is set *a priori* due to natural constraints in this kind of industries. Hence, when changing the production between two products of the same block only a minor setup is needed that is not dependent on the sequence, but only on the product to be produced.

In order to consider the initial stock that might be used to fulfil current demand it is important to have an overview of the inventory built up in each macro-period due to perishability concerns. The length of the horizon that needs to be considered is related to the product with the longest shelf-life. One shall consider an integer multiple X of past planning horizons that is enough to cover the longest shelf-life, i.e. $X = \lceil \frac{max\tilde{u}_k}{T} \rceil$, where $\tilde{u}_k$ is a conservative value for shelf-life of product k.

A macro-period is divided into a fixed number of non-overlapping micro-periods with variable length. Since the production lines can be scheduled independently, this is done for each line separately. It is important to notice that each line is assigned to a plant. The length of a micro-period is a decision variable, expressed by the production of several products of one block in the respective micro-period on a line and by the time to set up the block in case it is necessary. A sequence of consecutive micro-periods, where the same block is produced on the same line, defines the size of a lot of a block through the quantity of products produced during these micro-periods. Therefore, a lot may aggregate several products from a given block and may continue over several micro and macro-periods. Moreover, a lot is independent of the discrete time structure of the macro-periods. The number of micro-periods of each day defines the upper bound on the number of blocks to be produced daily on each line.

There is no inventory held at production plants. Thus, at the end of each day the production output is delivered to distribution centres (DCs), which have an unlimited storage capacity. The delivery function is assured by a third-party logistics (3PL), and we assume that it charges a flat rate per pallet transported between a plant and a DC. Moreover, it is assumed that the 3PL is able to cope with whatever distribution planning was decided beforehand and, hence, there is no capacity restriction for transportation. The distances between production plants and distribution centres are small enough so that the product is delivered on the same day it is produced. Therefore, the decrease of freshness during the transportation process is considered to be negligible. The small distance assumption is quite realistic in supply chains of highly perishable goods where the distribution centres are not very far away from the production plants. For our purposes these assumptions shall not pose a problem since we are still considering directly the most important cost drivers for transportation services: distance, quantity and service level. The demand for an item in a macro-period at a distribution centre is assumed to be dynamic and deterministic.

The problem is to plan production and distribution so as to minimize total cost and maximize mean remaining shelf-life of products at the distribution centres over a planning horizon.

3. RESULTS

To understand the trade-off present in the two developed models (fixed and loose shelf-life) regarding total costs and product freshness as well as the differences between using an integrated over a decoupled approach for production and distribution planning an illustrative example was developed.

In this instance there are four products to be scheduled and produced on two production lines that are located in two different production plants. Each of these products belongs to a different block and therefore there is always sequence dependent setup time and cost to consider when changing from one product to another. Moreover, although the first line is able to produce every product, the second one is not able to produce all of the products. The production lines are considered similar and, therefore, variable production costs are neglected. The number of micro-periods per macro-period was set at the constant value of four allowing the production of all products in a macro-period. The capacity of each line is the same in all macro-periods and every production plant. The planning horizon is ten days (macro-periods) and the shelf-life of products varies considerably among them, from highly perishable ones (one day) to others which can last throughout the entire planning horizon. Demand has to be satisfied in two different DCs and products can be transported between any pair *production plant – DC*. Initial stock was set to zero in both DCs. In case shelf-life is not fixed, there are three different temperature levels possible to be chosen at each DC influencing its duration. Finally, a sensitivity analysis regarding the perishability impact was conducted and different scenarios where shelf-lives and decay rates are varied were analysed.

3.1. Fixed Shelf-Life (Case 1)

In this section results for the case where the shelf-life is fixed are presented. In Figure 1 the Base scenario solutions of the Pareto-optimal fronts for both the integrated and decoupled approach are presented.

It is rather clear from the comparison of the Pareto fronts that the integrated approach strongly dominates the decoupled one. Both curves have a similar behaviour, which means that for the lower values of freshness just a small increase in costs fosters significantly the remaining shelf-life of delivered products. Nevertheless,

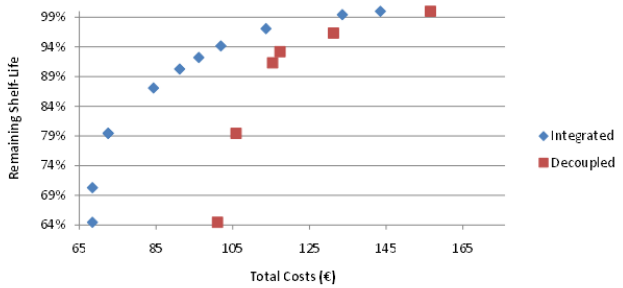


Figure 1: *Pareto-optimal fronts of the illustrative example when using an integrated and a decoupled approach (Case 1).*

when we are approaching a strict Just-in-Time (JIT) accomplishment of the demand, touching very high freshness standards, the costs start to increase in a more important way. Furthermore, it is interesting to notice that the savings in costs when using an integrated approach over a decoupled one tend to fade when we aim at an increased freshness. This may be explained by the fact that to achieve very high freshness standards almost no inventory is allowed since we are working under in a JIT policy, this will constrain so much the solution space that the integrated and coupled solutions are rather the same. Finally, in Figure 2 we perform a sensitivity analysis regarding the perishability settings. The percentage saving of using an integrated approach over a decoupled one is plotted for the three scenarios. In order to calculate the saving both Pareto fronts (integrated and decouple approach) were estimated through a second-order polynomial regression which has a good fit to the experimental data with all R^2 above 90%.

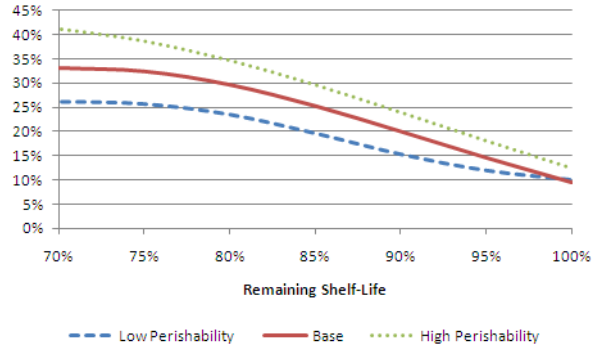


Figure 2: *Total percentage saving when using a integrated approach over a decoupled one for three scenarios (Case 1).*

The potential savings of using an integrated approach over a decoupled one are rather considerable for the fixed-shelf-life case and, independently of the scenario, the behaviour over the remaining shelf-life is quite similar. For the scenario with highly perishable products the savings can ascend up to 42% when aiming at 70% of remaining shelf-life.

When comparing the three scenarios it is observable that the advantages of using an integrated approach are leveraged by the degree of perishability the goods are subject to. In fact, when we are planning using a decoupled approach and the products are subject to intense perishability, the myopic mistakes incurred in production planning will be hardly corrected by the distribution process because the buffer between those activities is reduced by the small amount of time that goods can stay stored. Therefore, the advantages of using an integrated approach are boosted considerably for this scenario. On the other hand, when dealing with products with low perishability the buffer enables the possibility of correcting the potential production mistakes and the integrated approach has less comparative advantage.

3.2. Loose Shelf-Life (Case 2)

In this section we focus on the case where the shelf-life is loose. In Figure 3 the results of the Pareto fronts for both integrated and decoupled approach are presented. These solutions concern the Base scenario.

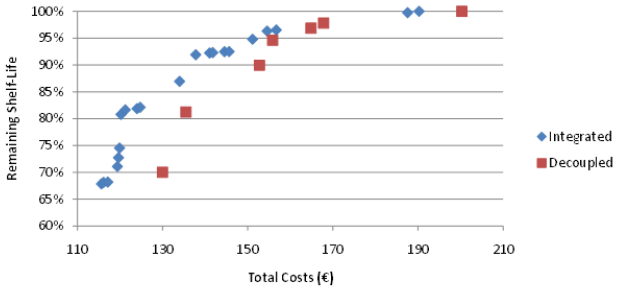


Figure 3: *Pareto-optimal fronts of the illustrative example when using an integrated and a decoupled approach (Case 2).*

As it happened with the results of Case 1, the Pareto front related to the integrated approach strongly dominates the one corresponding to the decoupled approach. It is interesting to note that both Pareto fronts are non-convex. The other reasoning made before for Case 1 regarding the behaviour of the fronts also applies to this case.

As done before for Case 1, in Figure 4 the results of the sensitivity analysis to understand the effect of different perishability settings is presented. The percentage saving of using an integrated approach over a decoupled one is plotted for the three scenarios.

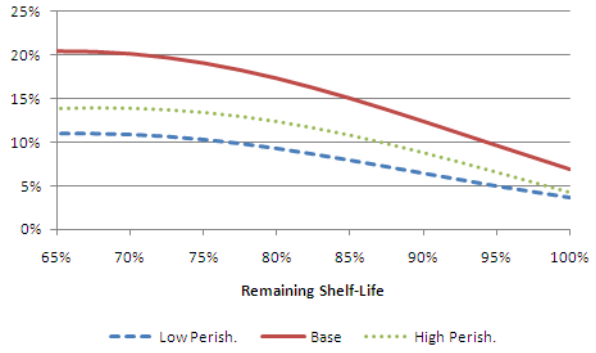


Figure 4: *Total percentage saving when using a integrated approach over a decoupled one for three scenarios (Case 2).*

Unlike Case 1 the savings are not as bold and the maximum saving ascends to 20% for an average remaining shelf-life of about 65% in the Base scenario, which is still rather remarkable. Nevertheless, the behaviour of both saving curves (from Case 1 and Case 2) is very similar. The explanation for the difference in the amount of savings between the fixed and loose shelf-life case may lie in the fact that for the loose shelf-life case the distribution process has much more freedom to influence both costs and specially product freshness. Hence, for the decoupled approach even after the production process has fixed the production quantities, the distribution process is still able to compensate potential mistakes through the decisions on temperature of storage.

Looking to the differences between the three scenarios it is interesting to notice that in this case the reasoning is not as straightfor-

ward as in Case 1. Here, the two extreme scenarios have a similar behaviour for different reasons. The scenario with products subject to low perishability has a rather humble saving when using an integrated approach for the same reasons as in Case 1. Hence, since the time buffer between production and distribution is rather large the advantages of using an integrated approach are hindered. In the scenario having products with a high perishability the explanation for the relative low saving is related to the possibility of correcting freshness problems coming from a myopic production planning in the decoupled approach through controlling the temperature of storage in the distribution planning. When products are highly perishable a small decrease in the storage temperature will entail a strong percentage augmentation of shelf-life. Hence, fi in a product with 7 days of shelf-life we are able to augment it to 8 days through storing it at cooler temperature, then the percentage increasing of shelf-life is not very significant. But, if the product is highly perishable, then an absolute increase of one day will reflect a strong percentage increase. Therefore, the scenario with products subject to intermediate perishability (Base scenario) is the one which gains more from an integrated approach.

4. CONCLUSIONS

In this paper, we have discussed the importance of integrating the analysis for a production and distribution planning problem dealing with perishable products. The logistic setting of our operational problem is multi-product, multi-plant, multi-DC and multi-period. We have developed models for two types of perishable products: with fixed shelf-life and with loose shelf-life, always taking into account that customers attribute decreasing value to products while they are aging until they completely perish. The novel formulations allow a comprehensive and realistic understanding of these intertwined planning problems. Furthermore, the loose-shelf-life model was able to incorporate the possibility of dealing with the underlying uncertainty of a random spoilage process with the help of predictive microbiology. To understand the impact of the integrated approach in both the economic and the freshness perspective a multi-objective framework was used. Since the formulations for the loose shelf-life case were not possible to solve with standard solvers, even for a small example, a simple heuristic was developed for these cases.

Computational results for an illustrative example show that the Pareto front of the integrated approach strongly dominates the Pareto front of the decoupled one for both classes of perishable products. The economic savings that this coupled analysis entail is smoothed as we aim to deliver fresher products. Nevertheless, in the fixed shelf-life case for a 70% mean remaining shelf-life of delivered products we may reach savings around 42%. The explanation regarding the fact that the gap between the integrated and the decoupled approach tends to smooth for very high freshness standards, may be due to the reason that in the latter case no inventory is allowed since we are working completely under a JIT policy, turning the problem at hand so constrained that the integrated and coupled solutions are rather the same. The multi-objective framework proved to be essential to draw these multi-perspective conclusions.

5. REFERENCES

[1] M. Tsiros and C. Heilman, "The effect of expiration dates and perceived risk on purchasing behavior in grocery store perishable categories," *Journal of Marketing*, vol. 69, pp. 114–129, 2005.

[2] S. Nahmias, "Perishable inventory theory: a review," *Operational Research*, vol. 30, no. 4, pp. 680–708, 1982.

[3] B. Fu and T. Labuza, "Shelf-life prediction: theory and application," *Food Control*, vol. 4, no. 3, pp. 125–133, 1993. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/0956713593902983

[4] C. Martin, D. Dent, and J. Eckhart, "Integrated Production, Distribution, and Inventory Planning at Libbey-Owens-Ford," *Interfaces*, vol. 23, no. 3, pp. 68–78, 1993.

[5] D. Thomas and P. Griffin, "Coordinated supply chain management," *European Journal of Operational Research*, vol. 94, no. 1, pp. 1–15, Oct. 1996. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/0377221796000987

[6] B. Bilgen and H.-O. Günther, "Integrated production and distribution planning in the fast moving consumer goods industry: a block planning application," *OR Spectrum*, vol. 32, no. 4, pp. 927–955, Jun. 2009. [Online]. Available: http://www.springerlink.com/index/10.1007/s00291-009-0177-4

[7] P. Amorim, C. H. Antunes, and B. Almada-Lobo, "Multi-objective lot-sizing and scheduling dealing with perishability issues," *Industrial & Engineering Chemistry Research*, vol. 50, no. 6, pp. 3371–3381, 2011.

Approaching a robust bi-objective supply chain design problem by a metaheuristic procedure

Cardona-Valdés Yajaira * Álvarez Ada * Pacheco Joaquín †

* Universidad Autónoma de Nuevo León
Nuevo León, México
yajaira@yalma.fime.uanl.mx, adalvarez@mail.uanl.mx

† Universidad de Burgos
Burgos, España
jpacheco@ubu.es

ABSTRACT

We consider the design of a two-echelon production distribution network with multiple manufacturing plants, customers and a set of candidate distribution centers. On this study we incorporate uncertainty on the demand of the customers which is represented through scenarios.

As well, there are several transportation options available for each pair of facilities between echelons. Each option represents a type of service with associated cost and time parameters leading an inverse correspondence between them. This tradeoff is handled through a bi–objective optimization model, where the involved objectives should be minimized. One criterion minimizes the expected cost of facility location, transportation, and the penalty for unmet demand. The other criterion looks for the minimum time to transport the product along any path from the plants to the customers.

An estimated Pareto robust front is found using several tabu searches. Preliminary experiments show the computational effect.

Keywords: Robust optimization, Multiobjective optimization, Supply chain, Metaheuristic, Tabu search

1. INTRODUCTION

In this study, we address the design of a supply chain of a two–echelon distribution system. The supply chain planning decisions can be classified as those concerning inventory, transportation and facility location [1]. This work is devoted to facility location and the selection of transportation modes, where both define the distribution network in the supply chain.

Network design decisions determine the supply chain configuration and have a significant impact in logistic costs and responsiveness [2]. For instance, facility location has a long term impact in the supply chain because of the high cost to open a facility or to move it. While cost related to opening a new facility and inventory pooling costs induce to reduce the number of facilities, responsiveness causes a contrary effect. A high number of facilities may reduce the lead time to deliver a product to the final customer. In certain products lead time can be viewed as an added value so that the firm that makes them available first can obtain short and long term competitive advantages in the market. As it can be seen, facility location decisions play a critical role in the strategic design of supply chain networks. For more details, a recent review on this topic can be found at [3]. The authors highlight the facility location models incorporated inside the supply chain management (SCM) framework, in particular, the integration of location decisions with other decisions relevant to the design of a supply chain network (typical decisions as capacity, inventory, procurement, production, routing, and the choice of transportation modes). As well as strengthen the missing literature involving uncertainty on the SCM.

In this study we consider a set of potential locations for new distribution centers. Each candidate site has a fixed cost for opening a facility with a limited capacity. In the supply network the number and location of plants and customers are known.

There are several transportation options available for each pair of facilities between echelons. The alternatives are generated by the supplier from different companies, the availability of different types of service at each company (e.g. express and regular), and the use of different modes of transportation (e.g. truck, rail, airplane, ship or inter–modal). Commonly, these differences involve an inverse correspondence between time and cost, i.e. a faster service will be more expensive.

In order to balance the economic concerns with prompt demand satisfaction, our approach is to minimize the total cost and the maximum time needed for shipping the product across the whole supply chain, simultaneously.

This bi–objective problem was first introduced by [4] as the "Capacitated Fixed Cost Facility Location Problem with Transportation Choices". In their contribution, the authors consider that all the design parameters are deterministic.

However, in practice, supply chains are characterized by numerous sources of technical and commercial uncertainties. Critical parameters as customer demands, prices and future facility capacities are quite uncertain. The fact that meeting customer demand is what mainly drives most supply chain initiatives motivated us to study the problem considering that the demand is a random variable whose value is not known at the time of designing the network.

Literature reveals that there are several studies which deal with uncertainty in supply chain management at different levels. In tactical level supply chain planning we can mention, for example, some papers related to the distribution of raw materials and products [5, 6]. At the strategic level, there is a great deal of research in the facility location component of supply chain network design under uncertainty. A good review can be found in the studies of [7, 3].

The optimization focus in traditional SCM problems on maximizing profit or minimizing costs as a single objective [8, 9, 10]. Nevertheless, other criteria to meet customer demand on time such as customer response time, or fill rate should be taken into account because they are related to the most basic functions of the SCM:

to meet customer requirements.

In the last years customer response time related considerations have been revisited in the distribution network design [11, 12]. Controlling lead time is becoming a competitive advantage for many firms because of the transformation of the manufacturing–distribution chains throughout the world. This parameter has effects on costs and also can be affected by the supply chain configuration.

Papers involving an integrated design of supply chain networks under uncertainty and considering several objectives are significantly smaller in number [13, 14, 15].

As commented in [3] in relation to the type of the objective function measuring supply chain performance, 75% attempts a cost minimization function, 16% attempts a profit maximization and only 9% refers to models with multiple and conflicting objectives.

In this study we assume that the response time is influenced by the selection of the transportation channel between facilities. The existence of third party logistic companies allows that different transportation services are available in the market, so in this paper we consider several alternatives to transport the product between facilities, where each option represents a type of service with associated cost and time parameters. The implicit assumption is that a faster transportation mode is also a more expensive one, creating a tradeoff between cost and time that affects the distribution network configuration.

The selection of a transportation channel has commonly been limited to the transportation mode. In an international context, different transportation modes are usually a consequence of the natural options of transportation around the world: by air, by sea or by land. On this matter, the literature related to supply chain management allowing several transportation modes to be chosen is too few, only four papers feature this characteristic on the review given by [3]. In this study, the term "transportation channel" is more generic and includes not only choices for the transportation mode but also for different types of services from one or several transportation companies. Although the principles may be the same, this distinction is important to describe a more general case.

Therefore, on one side, the objective of this study is to select the appropriate sites to open distribution centers and determine the flow between facilities to minimize the total expected cost involving facility location, transportation and a penalty for unmet demand. The last term describes model infeasibility and represents unmet demand under some scenario. An application in agile manufacturing is shown in [16].

On the other side, it is desired to minimize the transportation time from the plant to the customers. This part of the problem determines which services will be selected in order to reduce the transportation time in each echelon of the supply chain. Hence, the tradeoff between cost and time creates a bi–objective problem.

Even though there are studies that identify the cost–time tradeoff as an important element in supply chain design, they do not relate this balance to the availability of transportation choices between facilities [17, 18]. In [19], authors use an aggregated function for time and cost. Although different transportation modes are included in their model (rail and truck), the problem is to select between a direct and an inter–modal shipping strategy. They do not have transportation choices between each pair of locations.

The cost–time trade off, in conjunction with the uncertainty in demands, means that we are handling a novel multi–objective optimization problem under uncertainty. The contribution of this study is to propose a procedure to find a set of non-dominated robust solutions this problem

2. PROBLEM DESCRIPTION

The "Stochastic Capacitated Fixed Cost Facility Location Problem with Transportation Choices" (SCFCLP–TC) is based on a two–echelon system for the distribution of one product in a single time period. In the first echelon the manufacturing plants send product to distribution centers (DC). The second echelon corresponds to the flow of product from the distribution centers to the customers. The number of customers is known. The number of plants, their locations and manufacturing capacities are also known. There is a set of potential locations to open distribution centers. The number of open DC is not defined a priori. Each candidate site has a fixed cost for installing a DC, where the DC will have a limited operational capacity. There are several transportation options available for each pair of facilities between echelons. Each option represents a type of service with associated cost and time parameters. Each customer has an associated product demand, which must be supplied from a single DC. The exact demand realization is not known in advance. Thus, the demand will be considered as a random variable modeled through scenarios.

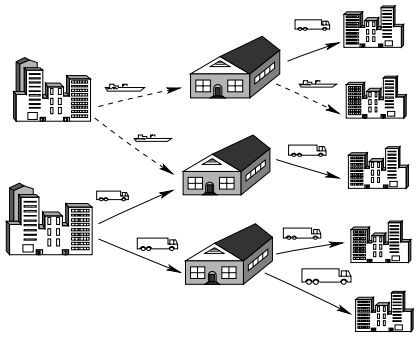


Figure 1: Supply Chain Configuration

An improved stochastic programming called robust programming was presented by [20]. An optimal solution to a robust optimization model is defined as *solution robust* if it remains "close" to optimal for all scenarios of the input data, and *model robust* if it remains "almost" feasible for all data scenarios. In this way, minimizing the expected combined cost of transportation and facility location will lead to a solution robust, while minimizing the expected cost for unmet demand will contribute to a model robust. That is, we are penalizing the unmet demand, so in the final solution the amount of unmet demand will be as small as possible.

The robust optimization model, by design, yield solutions that are less sensitive to the model data, given that the model measures the tradeoff between solution and model robustness.

The decision related to the transportation options has an impact on the transportation time from the plant to the customer. The tradeoff between cost and time must be considered in the formulation of a mathematical model that minimizes both criteria simultaneously. Hence, the problem should be addressed through a bi–objective optimization model. Following this approach, one criterion minimizes the combined expected cost of transportation, facility location, and the penalty for unmet demand. The other criterion looks for the minimum time to transport the product along any path from the plants to the customers.

3. METHODOLOGY PROPOSED

Metaheuristics have many desirable features to be an excellent method to solve very complex SCM problems: in general they are simple, easy to implement, robust and have been proven highly effective to solve hard problems [21].

The procedure for determining an estimated Pareto front is based in MOAMP [22] which consists on the following three phases:

1. Look for efficient solutions close to the ends of the Pareto frontier, that is, solutions that approximate to the best solutions of the single-objective problems that result considering each objective separately
2. Look for additional points inside the efficiency curve, that is, find the efficient points that represent a good compromise between the distinct objectives considered
3. Intensify the search around the efficient points found in previous phases

The first phase of MOAMP starts from an arbitrary initial solution and optimizes, at first, the objective f_1. Starting from the last point visited at the end of this search (usually a local optimum for f_1) the search is conducted again to find the best solution to the problem with the single objective f_2. In the case of two objectives, one more search is carried out for the objective f_1, starting from the best point found in the last search.

In our implementation, we first build a near optimal solution for objective f_1 and this point is considered as the initial point for the optimization process for f_2. Then, we build a near-optimal solution for f_2 and from this point we start the optimization for f_1.

In the second phase we launch several tabu searches using a global criterion method. In this step, the aim is to minimize a function that measures the distance to the ideal point following the notion of compromise programming, on the understanding that is logical for the decision maker to prefer a solution that is closer to the ideal point over the one that is farther away. The metric employed is the L_∞ because it has been shown to lead to balanced efficient solutions, as showed in [22]. In general, a point that minimizes an L_q $(1 \leq q \leq \infty)$ distance to the ideal point is an efficient point. The set of all points obtained in this way is called the compromise set. These have the characteristic of providing a good balance among the values of the p objective functions.

A graphical representation of the two phases of MOAMP is shown in figure 2.

Finally, the third phase consists on an intensification process of the initial Pareto front obtained during the first two phases. Here, each point on the efficient set is improved via a local search. After each point is improved the set of efficient points will be the approximation to the Pareto front provided.

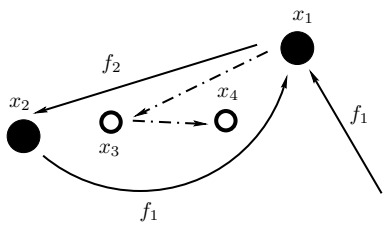


Figure 2: A general framework of the MOAMP procedure for a biobjective problem

4. DISCUSSION AND EXPERIMENTAL PRELIMINARY RESULTS

As mentioned before, the first part of MOAMP starts from an "arbitrary point". In our case, we first build a near optimal solution for objective f_1 and this point is considered as the initial point for the optimization process for f_2. The solution for objective f_1 is made via a GRASP procedure. For constructing a feasible solution, we proceed in a backward form. Starting from the second level of the supply chain, we solve a generalized assignment problem and apply the procedure developed by [23]. Once the customer-distribution center assignment has been determined, the first level of the supply chain gets the structure of a transportation problem, so we proceed solving that problem to complete a feasible solution. The objective function that guides the construction of this initial point is a robust function that minimizes the total expected cost for facility location, transportation and the penalty for demand unmet. Then, this feasible solution is improved using a local search procedure, exchanging customers–distribution centers assignments until it is not possible to get a better solution.

After that, an initial near-optimal solution for f_2 is constructed using also a GRASP procedure. This procedure is similar to the one designed for constructing a solution for objective f_1, however, the greedy function that guides the search is properly re-defined to take now into account the time required to transport the products along the supply chain.

For the intensification phase the same local search used in the GRASP procedure is applied. All visited points during the searches conducted on the three phases are checked for inclusion in the set of non-dominated solutions, which is the output of the algorithm.

In order to validate the proposed algorithm, several computational experiments have been designed. The first experiment intends to measure the performance of the method regarding to the size of the problem. For that purpose preliminary test were conducted on instances with 3 and 5 plants, 3 and 5 distribution centers, 4 and 8 customers, 2 or 3 transportation channels, and 2 or 3 scenarios. The second experiment attempts to measure the contribution of each phase of the proposed method toward the quality of the final approximation of the efficient set.

5. CONCLUSIONS

In this paper, we have studied a supply chain design problem that involves uncertainty on the customers' demands modeled by scenarios. Two conflicting objectives are considered: as well as the total cost, the maximum time needed for shipping the product across the chain total transportation time, has to be minimized.

We have formulated it by a biobjective model that minimizes the cost for opening distribution centers, the expected value of the transportation cost and the expected value for unmet demand. Simultaneously, the model minimizes the sum of the maximum lead time for the plants to the customers through each distribution center.

As the model penalizes the unmet demand in the final solution the amount of unmet demand will be as small as possible.

Taking into account the computational complexity of the addressed problem we have designed a solution approach based on metaheuristics. Preliminary results show the proposed method performs well, but the computational time grows as the numbers of scenarios increase. Therefore, ongoing work is conducted in order to improve this.

6. ACKNOWLEDGMENT

This work was partially supported by CONACYT (México) grant 61903.

7. REFERENCES

[1] R. H. Ballou, *Business Logistics Management*. USA: Upper Saddle River, 1999.

[2] S. Chopra and P. Meindl, *Supply Chain Management: Strategy, Planning and Operation.* USA: Prentice Hall, Upper Saddle River, 2004.

[3] M. T. Melo, S. Nickel, and F. Saldanha-da Gama, "Facility location and supply chain management– a review," *European Journal of Operational Research*, vol. 196, no. 1, pp. 401–412, 2009.

[4] E. Olivares, J. González-Velarde, and R. Ríos-Mercado, "A metaheuristic algorithm for a bi-objective supply chain design problem," in *RED-M*, 2007, pp. 5–8.

[5] H. Van Landeghem and H. Vanmaele, "Robust planning: A new paradigm for demand chain planning," *Journal of Operation Management*, vol. 20, no. 6, pp. 769–783, 2002.

[6] C.-S. Yu and H.-L. Li, "A robust optimization model for stochastic logistic problems," *International Journal of Production Economics*, vol. 64, no. 1-3, pp. 385–397, 2000.

[7] L. V. Snyder, "Facility location under uncertainty: a review," *IIE Transactions*, vol. 38, no. 7, pp. 537–554, 2006.

[8] P. Tsiakis, N. Shah, and C. Pantelides, "Design of multi–echelon supply chain networks under demand uncertainty," *Industrial and engineering chemistry research*, vol. 40, no. 16, pp. 3585–3604, 2001.

[9] T. Santoso, S. Ahmed, M. Goetschalckx, and A. Shapiro, "A stochastic programming approach for supply chain network design under uncertainty," *European Journal of Operational Research*, vol. 167, no. 1, pp. 96–115, 2005.

[10] S. Elhedhli and F. Gzara, "Integrated design of supply chain networks with three echelons, multiple commodities and technology selection," *IIE Transactions*, vol. 40, no. 1, pp. 31–44, 2008.

[11] I. Erol and W. G. Ferrell Jr., "A methodology to support decision making across the supply chain of an industrial distributor," *International Journal of Production Economics*, vol. 89, no. 1, pp. 119–129, 2004.

[12] A. De Toni and S. Tonchia, "Performance measurement systems: Models, characteristics and measures," *International Journal of Operations & Production Management*, vol. 21, no. 1-2, pp. 46–70, 2001.

[13] E. Sabri and B. Beamon, "A multi–objective approach to simultaneous strategic and operational planning in supply chain design," *The International Journal of Management Science*, vol. 28, no. 5, pp. 581–598, 2000.

[14] C. Chen, B. Wang, and W. Lee, "Multi–objective optimization for a multi-enterprise supply chain network," *Industrial and Engineering Chemistry Research*, vol. 42, no. 6–7, pp. 1879–1889, 2008.

[15] G. Guillén, F. D. Mele, M. J. Bagajewicz, A. Espuña, and L. Puigjaner, "Multiobjective supply chain design under uncertainty," *Chemical Engineering Science*, vol. 60, no. 6, pp. 1535–1553, 2005.

[16] F. Pan and R. Nagi, "Robust supply chain design under uncertain demand in agile manufacturing," *Computers and Operation Research*, vol. 37, pp. 668–683, 2010.

[17] G. Zhou, H. Min, and M. Gen, "A genetic algorithm approach to the bi-criteria allocation of customers to warehouses," *International Journal of Production Economics*, vol. 86, no. 1, pp. 35–45, 2003.

[18] T. Truong and F. Azadivar, "Optimal design methodologies for configuration of supply chains," *International Journal of Production Research*, vol. 43, no. 11, pp. 2217–2236, 2005.

[19] E. Eskigun, R. Uzsoy, P. Preckel, G. Beaujon, S. Krishnan, and J. Tew, "Outbound supply chain network design with mode selection, lead times and capacitated vehicle distribution centers," *European Journal of Operational Research*, vol. 165, no. 1, pp. 182–206, 2005.

[20] J. M. Mulvey, R. J. Vanderbei, and S. A. Zenios, "Robust optimization of large–scale systems," *Operations Research*, vol. 43, no. 2, pp. 264–281, 1995.

[21] H. Ramalhinho-Lourenço, "Supply chain management: An opportunity for metaheuristics," Department of Economics and Business, Universitat Pompeu Fabra, Economics Working Papers 538, 2001.

[22] E. Caballero, J. Molina, and R. V., MOAMP*: Programación Multiobjetivo mediante un procedimiento de búsqueda tabú*. Universidad de Oviedo, España: Actas del II Congreso Español de Metaheurísticas y Algoritmos Evolutivos y Bioinspirados: MAEB, 2003, pp. 153–159.

[23] S. Martello and P. Toth, *Knapsack problems: algorithms and computer implementations*. John Wiley & Sons, 1990.

A Tabu Search Approach for the Hybrid Flow Shop

Nicolau Santos * João Pedro Pedroso *

* INESC Porto and Faculdade de Ciências, Universidade do Porto
Rua do Campo Alegre, 4169-007 Porto, Portugal
nsantos@inescporto.pt, jpp@fc.up.pt

ABSTRACT

In this work we present a metaheuristic based on tabu search, designed with the objective of minimizing makespan in a hybrid flow shop problem. In order to assess the performance of the proposed method we performed tests using both well known benchmarks and randomly generated instances; preliminary results indicate that the approach is valid.

Keywords: Scheduling, Metaheuristics, Flow Shop, Combinatorial Optimization

1. INTRODUCTION

A Hybrid Flow Shop (HFS) consists of series of production stages, each of which has one or more machines operating in parallel; at least one stage has multiple machines, and at least one job has more than one stage. HFS problems appear as a natural extension of the traditional Flow Shop Problem. With the increasing complexity of modern production systems, the introduction of parallel machines, as well as additional constraints, are nowadays common. The HFS problem was initially stated in [1]; surveys of problems arising in this area and methods for solving them was provided in [2], [3], and more recently in [4].

In the HFS, each job is processed by one machine in each stage, and it must go through one or more stages. It can be defined as follows: there is a set of n jobs to be processed in m stages; all jobs have the same production direction, from stage 1 to stage m, and the production times t_{ik} of job i at stage k, are known. In this paper, we make the following further assumptions:

- setup and transportation times between stages are negligible;
- there are buffers with infinite capacity between stages;
- each machine can only process a job at a given time;
- a job can only be processed by a single machine at each stage;
- job preemption is not allowed.

While many objective functions are considered in the literature, we will focus on makespan minimization.

Figure 1 shows an example of an HFS with two machines at the first and second stages, and three machines at the third stage.

The numerous practical applications of HFS have attracted many researchers, and many approaches have been developed, from simple dispatching heuristics to exact methods. Regarding metaheuristics, there are two common exploration strategies: the first is to find the best job/machine association at each stage, as in [5], the second—the one we will use in this study—is to consider permutation schedules as in [6]. The main idea is to generate a permutation that defines the job order at the first stage; in the subsequent

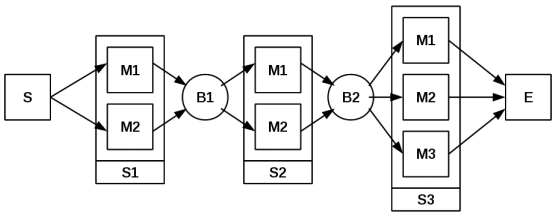


Figure 1: *Hybrid Flow Shop example.*

stages, jobs join a queue and are loaded to the machines in FIFO order, being assigned to the first available machine. Although this approach may fail to find the optimal association of jobs to stage machines, it is one of the most widely used in practice [7], as it prevents stock accumulation between stages, and naturally keeps the work in process at low levels, two important requirements in modern production systems.

Figure 2 presents an example of a seven job Gantt chart for the HFS presented in Figure 1. Notice that there is an exchange in the production order of jobs six and seven from the first to the second stage; this is due to different orders of arrival to the queues between those stages.

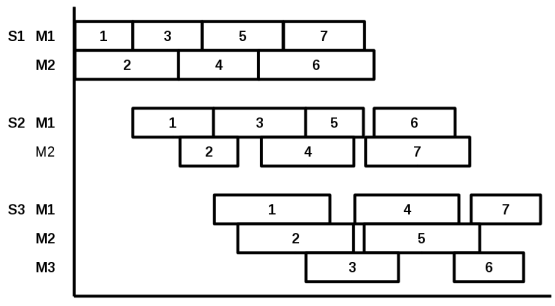


Figure 2: *Gantt chart example.*

2. TABU SEARCH

Tabu Search was proposed by Glover [8] as a method to guide heuristics through the solution space. Its main characteristic is the avoidance of being trapped in local optima, through the use of a tabu list: a recent memory record that prevents the repetition of moves as long as they are kept in the list. This avoids cycling in many cases (depending on the length of that list), leading the algorithm to explore promising regions.

Our implementation is based on the insertion neighborhood, and is relatively problem independent; it can easily be adapted to other objectives, if required.

```
procedure tabuSearch(π_init)
    π* = π_init # initialize best found solution
    π = π_init # initialize incumbent solution
    forall π_k in π: tabu[π_k] = 0 # initialize tabu list
    iter = 0 # iteration counter
    while termination criteria not satisfied do
        iter = iter + 1
        select L non-tabu jobs of π
        let r be a random integer, r_min ≤ r ≤ r_max
        let R be a set of r randomly chosen jobs of L
        evaluate N(R, π)
        let π' be the best neighbor found and π'_b the job selected for insertion
        update solution and tabu list
        if obj(π') < obj(π*) then
            π* = π'
            forall π_k in π': tabu[π_k] = 0
            tabu[π_b] = iter + 1
        else
            let t be a random integer, 1 ≤ t ≤ t_max
            tabu[π_b] = iter + t
        end if
    end while
    return π*
end procedure
```

Figure 3: *Tabu Search pseudocode*

2.1. Moves and Neighborhood

Tabu search exploration is based on moving iteratively to a solution in the neighborhood. In our algorithm we use insertion moves: given a permutation π and a pair of positions (i, j), $i \neq j$, the permutation π' obtained by removing job at position i and inserting it at position j is:

$\pi' = \pi_1, \ldots, \pi_{i-1}, \pi_{i+1}, \ldots, \pi_j, \pi_i, \pi_{j+1}, \ldots, \pi_n$ if $i < j$;

$\pi' = \pi_1, \ldots, \pi_{j-1}, \pi_i, \pi_j, \ldots, \pi_{i-1}, \pi_{i+1}, \ldots, \pi_n$ if $j < i$.

Having a set of U jobs we define $\mathcal{N}(U, \pi)$ as the neighborhood that contains all the possible insertion moves of the jobs in U.

2.2. Tabu list and search strategy

In our implementation the tabu list consists of an array: for each job i we assign a value $tabu[i]$, at iteration $iter$ we say that job i is *tabu* if $tabu[i] > iter$. The job π_b chosen to perform the move becomes tabu for t iterations, where $1 \leq t \leq t_{max}$. Hence, we update $tabu[\pi_b] = iter + t$, except when the best known solution is improved; in that case, we set $tabu[k] = 0, \forall k \neq \pi_b$ and $tabu[\pi_b] = iter + 1$, in order to prevent immediate reversion. Evaluating the neighborhood generated by trying insertion among any consecutive pair of jobs is a demanding computational task, so we propose a neighborhood restriction: instead of evaluating the complete neighborhood we evaluate a set of r randomly chosen jobs, with $r_{min} \leq r \leq r_{max}$.

To illustrate the behaviour of the algorithm, let us consider the example from Figure 2. Suppose that at a given iteration we have as incumbent solution (1,2,3,4,5,6,7). The operations to perform during a tabu search iteration are the following:

1. find the list of legal (non tabu moves) L; suppose we obtain $L = (1,2,3,4)$;
2. draw r to find the number of jobs to evaluate; suppose we obtain $r = 2$;
3. randomly choose r jobs from L; suppose we choose jobs 1 and 4 so $R = (1,4)$;
4. evaluate $\mathcal{N}(R, \pi)$, i.e., the permutations

(2,1,3,4,5,6,7)
(2,3,1,4,5,6,7)
(2,3,4,1,5,6,7)
(2,3,4,5,1,6,7)
(2,3,4,5,6,1,7)
(2,3,4,5,6,7,1)
(4,1,2,3,5,6,7)
(1,4,2,3,5,6,7)
(1,2,4,3,5,6,7)
(1,2,3,5,4,6,7)
(1,2,3,5,6,4,7)
(1,2,3,5,6,7,4).

Then, we choose the permutation that yields the best objective as the incumbent solution. Suppose the first permutation is chosen; in this case, set $\pi_b = 1$. The final step is to update the tabu list, as stated previously.

An extensive computational experiment is currenlty being conducted. The results of our method are being compared with the lower bound of [9] and known heuristics; though the results are preliminary, quality of the proposed method is very promising, with respect to other results found in the literature. We also refer that establishing a direct comparison with results from other authors is very hard, as each reports results based on its own set of randomly generated instances. In table 1 we present some results with instances from [10], initialy proposed for the flow shop problem and available in the Internet. Each instance as n jobs to be processed on m stages, p is the number of parallel machines introduced at each stage. *LB* is the value of the lower bound of [9], *B* is the best makespan found by our method and *AD* is the average of the relative percentage error *D,* were *D* is calculated by Equation 1.

$$D = \frac{heu_{sol} - LB}{LB}.100 \quad (1)$$

The results are derived from five runs on each instance with a running time of *n.m.45 ms* of CPU time in a computer AMD Athlon 64 X2 Dual Core 3800+ with 2Gb of RAM running OS Mandriva 2010 Free. For the problems with 5 and 10 stages we can observe small values of *AD*, though they are slightly larger for instances with 20 stages. To our knowledge, this is the first time results for this problem with 20 stages are presented.

			p=2			*p=4*		
inst	*n*	*m*	*LB*	*B*	*AD*	*LB*	*B*	*AD*
ta001	*20*	*5*	*688*	*721*	*5.41*	*428*	*459*	*7.57*
ta011		*10*	*885*	*1009*	*14.44*	*657*	*737*	*12.48*
ta021		*20*	*1332*	*1578*	*19.11*	*1237*	*1261*	*2.10*
ta031	*50*	*5*	*1395*	*1405*	*0.77*	*756*	*756*	*0.44*
ta041		*10*	*1572*	*1713*	*9.66*	*931*	*1052*	*13.64*
ta051		*20*	*2077*	*2430*	*17.59*	*1430*	*1658*	*16.58*
ta061	*100*	*5*	*2766*	*2803*	*1.47*	*1443*	*1468*	*1.93*
ta071		*10*	*2961*	*3058*	*3.71*	*1606*	*1737*	*8.61*
ta081		*20*	*3202*	*3720*	*16.79*	*1948*	*2270*	*17.26*
ta091	*200*	*10*	*5533*	*5603*	*1.61*	*2898*	*2966*	*3.22*
ta101		20	*5756*	*6290*	*10.05*	*3167*	*3569*	*13.81*
ta111	*500*	*20*	*13199*	*14246*	*8.25*	*6848*	*7666*	*12.30*

Table 1: *Average results for 5 independent runs on Taillard's benchmark*

3. ACKNOWLEDGEMENTS

The presented research was developed at INESC Porto under the European Commission, Framework Programme 7 project *FIT4U: Framework of Integrated Technologies for User Centred Products*.

4. REFERENCES

[1] T. Arthanari and K. Ramamurthy, "An extension of two machines sequencing problem," *Opsearch*, vol. 8, pp. 10–22, 1971.

[2] R. Linn and W. Zhang, "Hybrid flow shop scheduling: A survey," *Computers & Industrial Engineering*, vol. 37, no. 1-2, pp. 57 – 61, 1999, proceedings of the 24th international conference on computers and industrial engineering.

[3] H. Wang, "Flexible flow shop scheduling: optimum, heuristics and artificial intelligence solutions," *Expert Systems*, vol. 22, no. 2, pp. 78–85, 2005.

[4] R. Ruiz and J. Vázquez-Rodríguez, "The hybrid flow shop scheduling problem," *European Journal of Operational Research*, vol. 205, no. 1, pp. 1–18, 2010.

[5] E. Nowicki and C. Smutnicki, "The flow shop with parallel machines: a tabu search approach," *European Journal of Operational Research*, vol. 106, no. 2-3, pp. 226–253, 1998.

[6] D. Santos, J. Hunsucker, and D. Deal, "FLOWMULT: Permutation sequences for flow shops with multiple processors," *Journal of Information and Optimization Sciences*, vol. 16, pp. 351–366, 1995.

[7] M. Pinedo, *Scheduling: theory, algorithms, and systems*. Springer Verlag, 2008.

[8] F. Glover, "Future paths for integer programming and links to artificial intelligence," *Computers & Operations Research*, vol. 13, no. 5, pp. 533–549, 1986.

[9] D. Santos, J. Hunsucker, and D. Deal, "Global lower bounds for flow shops with multiple processors," *European Journal of Operational Research*, vol. 80, no. 1, pp. 112–120, 1995.

[10] E. Taillard, "Benchmarks for basic scheduling problems," *European Journal of Operational Research*, vol. 64, no. 2, pp. 278 – 285, 1993, project Management anf Scheduling. [Online]. Available: http://www.sciencedirect.com/science/article/B6VCT-48MYGV0-4W/2/9dd8e0f50213f3302f7ebf1a80dca3b7

Sequencing approaches in Synchronous Manufacturing

Jan Riezebos *

* University of Groningen, The Netherlands
P.O.box 800, 9700AV Groningen, The Netherlands
j.riezebos@rug.nl

ABSTRACT

We consider a sequencing problem in a synchronized manufacturing environment. Order release is an essential part of this system. As orders may differ in the amount and distribution of their capacity requirements over subsequent production stages, total capacity load may vary over time. We encountered this problem in a labor-intensive cellular environment. In practice, heuristics are used to solve this problem, but their effectiveness is questioned. This paper examines heuristics that are based on insights from assembly system design and work load control. The heuristics are evaluated in a rolling schedule environment.

Keywords: Synchronous manufacturing, Bottleneck, Employee scheduling

1. INTRODUCTION

The basic idea of synchronous manufacturing is to create a flow of work through the manufacturing system, either continuous or intermittent, in order to achieve short and constant throughput times and a predictable loading of the resources in the system. Recently, fixed cycle-time synchronization approaches have been developed that no longer implicitly assume an inflexible capacity. They consider total capacity to be limited, but capacity to be flexible between the stages of the production system. The reason for relaxing this assumption is that workers are nowadays increasingly multi-skilled and cross-trained. This flexibility makes it possible to handle capacity fluctuations between stages in a production system. However, total capacity in terms of the number of workers available does not increase through such measures. Therefore, these synchronization approaches aim at an order release decision that effectively uses the available capacity of the multi-skilled workers while still realizing a high output for the whole production system. Examples of papers in this area are [1], [2], [3], [4], and [5]. The complexity of the resulting synchronization problems has been analyzed by [6]. They showed that most leveling problems in such systems are NP-complete, even if they only consist of two stages. Therefore, in practice heuristic solutions are being used to solve the order release decision, as the number of stages often is much larger than two.

This paper discusses various single pass heuristics for the order release decision in such a synchronization approach with a fixed cycle time in a multi-product multi-stage situation. Single pass heuristics determine a sequence without back tracking or pair wise interchanging parts of a solution. Such heuristics are often used in practice. The question is whether these heuristics can be improved by incorporating insights from related fields, such as workload control and assembly line balancing. This paper will examine the performance of several heuristics in a rolling schedule environment. Testing in a rolling schedule environment provides better insights in the performance of these heuristics in the long term, as it prohibits the negative impact of postponing problems to the end of the cycle.

2. PROBLEM DEFINITION

In a synchronous manufacturing mode, stages represent a subset of operations that are to be performed in a cell within a fixed period of time. At the end of each period, all jobs that are in progress are transferred to their next stage. At such a transfer moment, employees may have to switch to other tasks within the cell.

The cell starts each period with a new order that is selected from a list of orders that should be completed during the cycle (i.e., at the end of the week). The batch size, process plans, and work content per stage may differ per order. Therefore, capacity requirements may vary strongly, both per order and per stage.

An important problem faced in such synchronous manufacturing systems concerns the capacity balance over time ([7]). A cellular system makes it less appropriate to vary the total number of employees over time. Employees should feel themselves responsible for the whole task of the cell. A relatively constant number of employees over various periods is therefore preferred.

Figure 1 presents a realistic example with 10 orders that have to be released during one week. We have 10 orders (A,...,J) and five stages j=1,...,5. An order that starts in period t=1 in stage j=1 arrives in period $t = 2$ in stage j=2, and leaves the system at the end of period t=5, if stage j=5 has been finished. Orders are represented using different shades. The amount of capacity required in a stage differs per order and is presented in the cells of the table. A row shows the fluctuating capacity requirements of that stage. Orders may also require a different total number of employees (i.e. the sum of the cells with identical shade). However, the sum of the cells in the same column is more important for the capacity management of the firm. This shows the total number of employees needed in a single period. If this number exceeds the available capacity, the firm has to hire additional employees or change the sequence of orders. The problem is to determine one or more sequences for the orders A,...,J such that the available capacity in each period t is not exceeded as long as possible. For the first four periods, earlier decisions on the sequence in a previous cycle affect the loading of the available capacity. The cells at the bottom-left side express the capacity requirements of these already started orders that still have to complete one or more stages. The same effect appears at the end of the cycle, as the last four orders in the sequence affect not only capacity in this cycle, but also capacity requirements in the next cycle. Here the end-of-horizon effect or truncated-horizon effect appears (see e.g. [8]). The loading should result in a sequence for which the available capacity per period is not exceeded. The sequence presented in Figure 1 results in four periods that encounter a capacity shortage if available capacity equals 20 per period: t=2,6,9,10. Can a better order sequence be found?

day	Mon		Tue		Wed		Thu		Fri		Mon2		Tue2	
bucket (4 hours)	am	pm	am	pm	am	pm	am	pm	am	pm	am	pm	am	pm
period *t*	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Stage \ Order	A	B	C	D	E	F	G	H	I	J				
j=1	3	5	3	2	2	3	2	4	3	6	3.3	3.3	3.3	3.3
j=2	4	5	3	2	6	2	2	2	4	3	2	3.2	3.2	3.2
j=3	6	4	4	4	2	6	5	2	5	4	5	2	4.1	4.1
j=4	3	4	5	6	4	8	3	4	8	4	4	5	2	4.6
j=5	2	3	2	4	2	4	5	2	4	5	4	4	3	2
Total	18	21	17	18	16	23	17	14	24	22	21	22	21	28
(Exp.) Shortage	0	1	0	0	0	3	0	0	4	2	0	0	0	0

Figure 1: Sequencing 10 products, 5 stages, period length 4 hours, cycle length one week

3. MATHEMATICAL PROBLEM FORMULATION

The problem can be formulated as:
Given:

- a set of *i=1,...,n* orders that have to start during periods *t=1,...,n*;
- their capacity requirements C_{ij} during *j=1,...,m* stages of completion;
- the capacity requirements CR_{tj} for orders that have already started but are not yet completed;
- average required capacity in stage j (*j=1,...,m-1*)

$$ARC_j = \frac{\sum_{t=1}^{j-1} CR_{tj} + \sum_{i=1}^{n} C_{ij}}{n+j-1},$$

- capacity requirements to complete stage j in period t Y_{tj} *(t=1,...,n+m-1; j=1,...,m)*;
- available capacity per period AC_t,
- capacity shortage (expected) in period t $CS_t = \max(0, \sum_{j=1}^{m} Y_{tj} - AC_t)$ (t=1,...,n+m-1)
- weight factor for capacity shortage in period t w_t ($w_t = 1$ if $t = 1,...,n$; $w_t \leq 1$ if $t > n$ (*t=1,...,n+m-1*))

determine the sequence of the n orders $X_{it}(i = 1,...,n; t = 1,...,n)$ such that $\sum_{t=1}^{n+m-1} w_t CS_t$ is minimized, where

$X_{it} = 1$ if order i starts in period t, else 0 (*i=1,...,n; t=1,...,n*)

The problem can mathematically be formulated as:

$$Minimize \sum_{t=1}^{n+m-1} w_t \cdot CS_t \quad (1)$$

such that

$$\sum_{i=1}^{n} X_{it} = 1 \forall t = 1,...,n \quad (2)$$

$$\sum_{i=1}^{n} X_{it} = 1 \forall i = 1,...,n \quad (3)$$

$$Y_{tj} = CR_{tj} \forall j = 2,...,m \forall t = 1,...,j-1 \quad (4)$$

$$Y_{tj} = ARC_j \forall j = 1,...,t-n \forall t = n+1,...,n+m-1 \quad (5)$$

$$Y_{t+j-1,j} = \sum_{i=1}^{n} X_{it} \cdot C_{ij} \forall t = 1,...,n \forall j = 1,...,m \quad (6)$$

$$\sum_{j=1}^{m} Y_{tj} - CS_t <= AC_t \forall t = 1,...,n+m-1 \quad (7)$$

$$X_{it} \in [0,1]; CS_t \geq 0; Y_{tj} \geq 0; \quad (8)$$

Constraints (2) and (3) guarantee that a feasible assignment is obtained. Constraint (4) shows that we use this model in a rolling schedule environment, as we take the effects of the release decisions in the former cycle into account in the current decision. Constraint (5) calculates the consequences for future periods that will be affected by the current decision. We use the average load in a stage if the actual load is not being determined in the current decision round. Constraint (6) determines the consequences of the release decision for the stage load in subsequent periods. Constraint (7) calculates the capacity shortage per period. The objective function (1) minimizes a weighted sum of the capacity shortages over time. Through the use of positive weights <1 for periods behind the sequence horizon, the end of horizon effect is avoided.

4. ORDER RELEASE HEURISTICS

As similar sequencing problems in [6]are NP complete in the strong sense for situations with the number of stages m greater or equal to three, we have investigated heuristics to determine a sequence of orders for release. The first heuristic FillCap was shown to be the most effective single pass heuristic in [9]. It is compared with a new heuristic FillCapBottleneck. Both heuristics are presented in the appendix. FillCap chooses the work order that can maximally fill the capacity of period t at the beginning of that period from the set of assignable orders. The set of assignable orders consists of orders to be released and have a capacity requirement in stage j not exceeding the available capacity. In situations when there are more than one order available, it will choose randomly. The problem of FillCap is that it only focuses on filling the capacity in the current period. It does not take into account the effect of sequencing decisions in the next periods. Therefore, FillCapBottleneck has been developed so that work orders can be properly selected to minimize the difference between available capacity and expected work load at the bottleneck stage. In this case, the set of assignable work orders in period t contains all work orders that still have to be released in this cycle and have an expected capacity requirement in the bottleneck stage nearest to the available capacity of period t. The expected capacity at a bottleneck stage is calculated based upon the average load of preceding stages and the already assigned work load of orders for the succeeding stages.

5. RESULTS

We evaluate the approaches on the total capacity shortage per cycle. This is a kind of tardiness measure, as capacity shortage is zero if there is overcapacity in every bucket. For each bucket $t = 1,...,n$ we calculate the capacity shortage CS_t and add it to the total capacity shortage in that cycle. We experimented with problems of different size, i.e. cycles with 10 jobs, 20 jobs, and 40 jobs. The number of stages equals 5. See figure 2 for the results.

	Optimal Solution	Heuristics	
		FillCap	FillCap Bottleneck
10-Product (1)	1	4	5
10-Product (2)	2	5	4
10-Product (3)	1	6	4
10-Product (4)	2	7	6
10-Product (5)	1	5	3
20-Product (1)	1	9	4
20-Product (2)	1	8	8
20-Product (3)	1	6	9
20-Product (4)	1	9	8
20-Product (5)	1	8	4
40-Product (1)	1	12	14
40-Product (2)	1	13	10
40-Product (3)	1	9	9
40-Product (4)	1	13	10
40-Product (5)	1	16	14

Figure 2: Optimal solution compared with heuristics

The results show that in 73% of the cases, FillCapBottleneck outperforms FillCap, while in 18% of the cases FillCap was better. However, the gap with the optimal solution remains very high, making it attractive to extend the search or apply the optimal solution method if time allows. Calculation time of the optimal solution rapidly increases with the number of products (0.4 sec for 10-products, 1.6 hours for 40-products). See Figure 3 for an overview.

	Optimal solution	Heuristics
10-Product	0.4 sec.	< 1 sec.
20-Product	20 min., 12 sec.	< 1 sec.
40-Product	1.6 hour	< 1 sec.

Figure 3: Execution time optimal solution and heuristics

6. CONCLUSIONS

The concept of synchronous manufacturing aims at achieving short and reliable throughput times through the introduction of fixed transfer moments between several stages of production. This causes a loading of the resources that can be predicted in advance, which makes it easier for planners to do their job.

We developed optimal and heuristic solution approaches for this multi-product multi-stage problem. The FillCap heuristic focuses on maximum utilization of the first stage. It selects an order that consumes as much capacity in the first stage as possible, while FillCapBottleneck focuses on loading the bottleneck stage.

Future research should verify if other heuristics, such as genetic algorithms, can be applied as well. The current study has applied deterministic optimization in a rolling schedule horizon. We think that the use of a rolling schedule has important benefits when testing the performance of different solution approaches.

7. REFERENCES

[1] C.-Y. Lee and G. L. Vairaktarakis, "Workforce planning in mixed model transfer lines," *Operations Research*, vol. 45, pp. 553–567, 1997.

[2] G. L. Vairaktarakis, X. Cai, and C.-Y. Lee, "Workforce planning in synchronous production systems," *European Journal of Operational Research*, vol. 136, pp. 551–572, 2002.

[3] M. Gronalt and R. Hartl, "Workforce planning and allocation for mid-volume truck manufacturing: a case study," *International Journal of Production Research*, vol. 41, pp. 449–463, 2003.

[4] J. Bukchin and M. Masin, "Multi-objective design of team oriented assembly systems," *European Journal of Operational Research*, vol. 156, pp. 326–352, 2004.

[5] E. Cevikcan, M. B. Durmusoglu, and M. E. Unal, "A team-oriented design methodology for mixed model assembly systems," *Comput. Ind. Eng.*, vol. 56, pp. 576–599, 2009.

[6] G. L. Vairaktarakis and X. Cai, "Complexity of workforce scheduling in transfer lines," *J. of Global Optimization*, vol. 27, pp. 273–291, 2003.

[7] V. I. Cesaní and H. J. Steudel, "A study of labor assignment flexibility in cellular manufacturing systems," *Comput. Ind. Eng.*, vol. 48, pp. 571–591, 2005.

[8] H. Stadtler, "Improved rolling schedules for the dynamic single-level lot-sizing problem," *Manage. Sci.*, vol. 46, pp. 318–326, 2000.

[9] J. Riezebos, "Order sequencing and capacity balancing in synchronous manufacturing," *International Journal of Production Research*, vol. 49, pp. 531–552, 2011.

8. APPENDIX

Step 0 *Unassigned* $:=\{1,\ldots,n\}$

$Y_{tj} := CR_{tj} \quad \forall j = 2\ldots m \quad \forall t = 1\ldots m-1$

$t := 0$

FillCap: $b := 1$

FillCapBottleneck: $b := \arg\max_{j=1\ldots m}(ARC_j)$ *Bottleneck stage*

Step 1 $t := t+1$

$AvailableAtBottleneck_t := AC - \sum_{j=1}^{b-1} ARC_j - \sum_{j=b+1}^{m} Y_{tj}$

Step 2 $Assignable_t := \arg\min_{i \in Unassigned} \left(\left|C_{ib} - AvailableAtBottlebeck_t\right|\right)$

IF $Assignable_t$ *contains more than one element: GoTo Step 3*

ELSE $i = Assignable_t$*; GoTo Step 4*

Step 3 $AvailableAtStage1_t := AC - \sum_{j=2}^{m} Y_{tj}$

$LimitedSetAssignable_t := \arg\min_{i \in assignable_t} \left(\left|C_{i1} - AvailableAtStage1_t\right|\right)$

IF $LimitedSetAssignable_t$ *contains more than one element:*

Select i randomly from $LimitedSetAssignable_t$*;*

GoTo Step 4

Step 4 $X_{it} := 1$ *(i allocated to sequence number t)*

$Y_{t+j-1,j} := C_{ij} \quad \forall j = 1\ldots m$

Unassigned := Assigned – {i}

IF Unassigned $\neq \emptyset$*: GoTo Step1; ELSE Stop*

Figure 4: FillCap and FillCapBottleneck heuristics

Affine recourse for the robust network design problem: between static and dynamic routing

Michael Poss * Christian Raack * †

* Department of Computer Science, Faculté des Sciences
Université Libre de Bruxelles, Brussels, Belgium
mposs@ulb.ac.be

† Zuse Institute Berlin (ZIB)
Takustr. 7, D-14195 Berlin, Germany
raack@zib.de

ABSTRACT

Affinely-Adjustable Robust Counterparts are used to provide tractable alternatives to (two-stage) robust programs with arbitrary recourse. We apply them to robust network design with polyhedral demand uncertainty, introducing the affine routing principle. We compare the affine routing to the well-studied static and dynamic routing schemes for robust network design. It is shown that affine routing can be seen as a generalization of the widely used static routing still being tractable and providing cheaper solutions. We investigate properties on the demand polytope under which affine routings reduce to static routings and also develop conditions on the uncertainty set leading to dynamic routings being affine. We show however that affine routings suffer from the drawback that (even strongly) dominated demand vectors are not necessarily supported by affine solutions. The proofs and computational results are not presented due to the space restriction.

Keywords: Robust optimization, Network design, Recourse, Affine Adjustable Robust Counterparts, Demand polytope

1. INTRODUCTION

In the classical deterministic network design problem, a set of point-to point commodities with known demand values is given, and capacities have to be installed on the network links at minimum cost such that the resulting capacitated network is able to accommodate all demands simultaneously by a multi-commodity flow. In practice however, exact demand values are usually not known at the time the design decisions must be made. Robust optimization overcomes this problem by explicitly taking into account the uncertainty of the data introducing so-called uncertainty sets. A solution is said to be feasible if it is feasible for all realizations of the data in a predetermined uncertainty set $\mathscr{D}$ [1]. Introducing even more flexibility, two-stage robust optimization allows to adjust a subset of the problem variables only after observing the actual realization of the data [2]. In fact, it is natural to apply this two-stage approach to network design since very often first stage capacity design decisions are made in the long term while the actual routing is adjusted based on observed user demands. This second stage adjusting procedure is called *recourse* which in the context of network design relates to what is known as traffic engineering. Unrestricted second stage recourse in robust network design is called *dynamic routing*, see [3]. Given a fixed design, the commodity routing can be changed arbitrarily for every realization of the demands. In [3] it is shown that allowing for dynamic routing makes robust network design intractable. Already deciding whether or not a fixed capacity design allows for a dynamic routing of demands in a given polytope is $\mathscr{NP}$-complete (on directed graphs).

This paper is motivated by the scarcity of works using affine routing. Following [2], we introduce affine routing as a generalization of static routing allowing for more routing flexibility but still yielding polynomially solvable robust counterparts, (in opposition to the schemes from [4] and [5]). In this context affine routing provides a tractable alternative in between static and dynamic routing. Affine routing has been used implicitly already in [6] for a robust network design problem with a particular uncertainty set. The contributions of this paper consist of a theoretical and empirical study of network design under the affine routing principle for general polyhedral demand uncertainty sets $\mathscr{D}$. Section 2 introduces the mathematical models and defines formally static, affine and dynamic routings. In Section 3 we present our main results. Proofs are ommited due to space restrictions. We also conducted numerical comparisons of static, affine and dynamic routings, which are not presented due to space restrictions.

2. ROBUST NETWORK DESIGN WITH RECOURSE

We are given a directed graph $G = (V,A)$ and a set of commodities K. A commodity $k \in K$ has source $s(k) \in V$, destination $t(k) \in V$, and demand value $d^k \geq 0$. A *flow* for k is a vector $f^k \in \mathbb{R}^A_+$ satisfying:

$$\sum_{a \in \delta^+(v)} f^k_a - \sum_{a \in \delta^-(v)} f^k_a = d^k \psi_{vk} \quad \text{for all } v \in V, \tag{1}$$

where $\delta^+(v)$ and $\delta^-(v)$ denote the set of outgoing arcs and incoming arcs at node v, respectively. For node $v \in V$ and commodity $k \in K$ we set $\psi_{vk} := 1$ if $v = s(k)$, $\psi_{vk} := -1$ if $v = t(k)$, and $\psi_{vk} := 0$ else. Flows are non-negative. A *multi-commodity flow* is a collection of flows, one for each commodity in K. A *circulation* (or cycle-flow) is a vector $g \in \mathbb{R}^A$ satisfying

$$\sum_{a \in \delta^+(v)} g_a - \sum_{a \in \delta^-(v)} g_a = 0 \quad \text{for all } v \in V. \tag{2}$$

A circulation is not necessarily non-negative. A value $g_a < 0$ can be seen as a flow from the head of arc a to its tail. We call a circulation g *non-negative* if $g \geq 0$ and *positive* if additionally $g \neq 0$. Notice that any two flows $\hat{f}^k, f^k$ for k only differ by a circulation, that is, there always exists a circulation g such that $\hat{f}^k = f^k + g$.

In many practical situations, the demand vector $d \in \mathbb{R}^K_+$ is uncertain. In the sequel we assume that $d \in \mathscr{D} \subset \mathbb{R}^K$ with $\mathscr{D}$ being a polytope. Any $d \in \mathscr{D}$ is said to be a realization of the demand. A *routing* is a function $f : \mathscr{D} \to \mathbb{R}^{A \times K}_+$ that assigns a multi-commodity flow to every realization of the demand. We say that

f serves $\mathscr{D}$ and call f a dynamic routing if there is no further restriction on the routing. A capacity allocation $x \in \mathbb{R}^A_+$ is said to *support* the set $\mathscr{D}$ if there exists a routing f serving $\mathscr{D}$ such that for every $d \in \mathscr{D}$ the corresponding multi-commodity flow $f(d)$ is not exceeding the arc-capacities given by x. Robust network design now aims at providing the cost minimal capacity allocation supporting $\mathscr{D}$. In this respect, robust network design is a two-stage robust program with recourse, following the more general framework described by [2]. The capacity design has to be fixed in the first stage and observing a demand realization $d \in \mathscr{D}$, we are allowed to adjust the routing $f(d)$ in the second stage. The problem can be written as the following (semi-infinite) linear program:

$$(RND) \quad \min \sum_{a\in A} \kappa_a x_a$$

$$\sum_{a\in\delta^+(v)} f^k_a(d) - \sum_{a\in\delta^-(v)} f^k_a(d) = d^k \psi_{vk}, \quad v \in V, k \in K, d \in \mathscr{D} \tag{3}$$

$$\sum_{k\in K} f^k_a(d) \le x_a, \qquad a \in A, d \in \mathscr{D} \tag{4}$$

$$f^k_a(d) \ge 0, \qquad a \in A, k \in K, d \in \mathscr{D} \tag{5}$$

$$x_a \ge 0, \qquad a \in A,$$

where $\kappa_a \in \mathbb{R}_+$ is the cost for installing one unit of capacity on arc $a \in A$. As already mentioned, deciding whether or not a given capacity vector x supports $\mathscr{D}$ is $\mathscr{NP}$-complete for general polytopes $\mathscr{D}$ [3]. It follows that (unless $\mathscr{P} = \mathscr{NP}$) it is impossible to derive a compact formulation for (*RND*) with dynamic routing. Using a branch-and-cut approach based on Benders decomposition, Mattia [7] shows how the solve the $\mathscr{NP}$-hard separation problem for robust metric inequalities using bilevel and mixed integer programs.

Most authors ([8, 9, 10], among others) use a simpler version of (*RND*) introducing a restriction on the second stage recourse known as static routing (also called oblivious routing). Each component $f^k : \mathscr{D} \to \mathbb{R}^A_+$ is forced to be a linear function of d^k:

$$f^k_a(d) := y^k_a d^k \qquad a \in A, k \in K, d \in \mathscr{D}. \tag{6}$$

Notice that by (6) the flow for k is not changing if we perturb the demand for $h \neq k$. By combining (6) and (3) it follows that the multipliers $y \in \mathbb{R}^{A\times K}_+$ define a multi-commodity (percentage) flow. For every $k \in K$, the vector $y^k \in \mathbb{R}^A_+$ satisfies (1) setting $d^k = 1$. The flow y is called a *routing template* since it decides, for every commodity, which paths are used to route the demand and what is the percental splitting among these paths. The routing template has to be used by all demand scenarios $d \in \mathscr{D}$ under the static routing scheme.

Ben-Tal *et al.* [2] introduce Affine Adjustable Robust Counterparts restricting the recourse to be an affine function of the uncertainties. Applying this framework to (*RND*) means restricting f^k to be an affine function of *all* components of d giving

$$f^k_a(d) := f^{0k}_a + \sum_{h\in K} y^{kh}_a d^h \ge 0, \qquad a \in A, k \in K, d \in \mathscr{D}, \tag{7}$$

where $f^{0k}_a, y^{kh}_a \in \mathbb{R}$ for all $a \in A, k, h \in K$, also see [6]. In what follows, a routing f serving $\mathscr{D}$ and satisfying (7) for some vectors f^0 and y is called affine. We see immediately that static routing can be obtained from (7) by imposing $f^{0k}_a = 0$ and $y^{kh}_a = 0$ for each $a \in A$ and all $k, h \in K$ with $k \neq h$. In this context affine routing generalizes static routing allowing for more flexibility in reacting to demand fluctuations, but it is not as flexible as dynamic routing. Formally it holds

$$opt_{dyn} \le opt_{aff} \le opt_{stat},$$

where opt_{dyn}, opt_{aff}, and opt_{stat} denote the cost values of the optimal solution to (*RND*) where f is allowed to be dynamic, affine, or static, respectively. Note that there is a proven (tight) worst-case optimality gap of $O(log|V|)$ between the dynamic and static routing principle, see [11]. In this paper we do not establish optimality gaps between the three routing principles. We rather focus on studying properties of the demand scenarios $\mathscr{D}$ that either yield $opt_{stat} = opt_{aff}$ or $opt_{aff} = opt_{dyn}$.

Given a demand polytope $\mathscr{D}$, a static routing f is completely described by the vector $y \in \mathbb{R}^{A\times K}_+$. Similarly, an affine routing is completely described by fixing the vectors $f^0 \in \mathbb{R}^{A\times K}$ and $y \in \mathbb{R}^{A\times K\times K}$. Extending the previous definitions, any routing template $y \in \mathbb{R}^{A\times K}_+$ is said to serve $\mathscr{D}$ if it yields a (static) routing f serving $\mathscr{D}$. Similarly, any pair of vectors $f^0 \in \mathbb{R}^{A\times K}$ and $y \in \mathbb{R}^{A\times K\times K}$ that satisfies (3) and (7) are said to *serve* $\mathscr{D}$. Given a capacity allocation $x \in \mathbb{R}^A_+$, the pair (x,y) with y serving $\mathscr{D}$, or the triplet (x, f^0, y) with (f^0, y) serving $\mathscr{D}$ are said to *support* $\mathscr{D}$ if the corresponding routings satisfy (4).

The model (*RND*) contains an infinite number of inequalities. However, when $\mathscr{D}$ is convex, we can replace $\mathscr{D}$ by the set of its extreme points, which is finite whenever $\mathscr{D}$ is a polytope.

Lemma 1. *Let* $\mathscr{D} \subset \mathbb{R}^K$ *be a bounded set and* x *be a capacity allocation* $x \in \mathbb{R}^A$.

(a) x *supports* $\mathscr{D}$ *if and only if* x *supports* $\mathrm{conv}(\mathscr{D})$.

(b) (x,y) *supports* $\mathscr{D}$ *if and only if* (x,y) *supports* $\mathrm{conv}(\mathscr{D})$.

(c) (x, f^0, y) *supports* $\mathscr{D}$ *if and only if* (x, f^0, y) *supports* $\mathrm{conv}(\mathscr{D})$.

Hence (*RND*) can be discretized by restricting the model to the extreme demand scenarios that correspond to vertices of $\mathscr{D}$ (for all three routing schemes).

3. AFFINE ROUTINGS

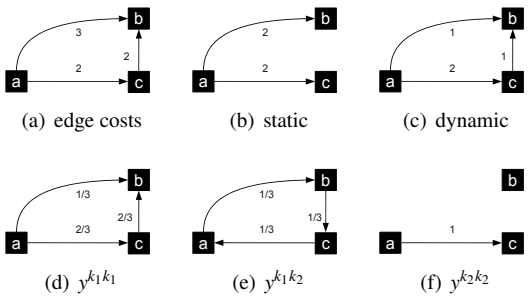
In this section, we study properties and consequences of the affine routing principle. Using (7) and substituting the flow variables in the balance constraints (3) it can be seen that affine routing has a nice interpretation as paths and cycles:

Lemma 2. *Let* $\mathscr{D}$ *be a demand polytope and let* $(f^0, y) \in \mathbb{R}^{A\times K} \times \mathbb{R}^{A\times K\times K}$ *be an affine routing serving* $\mathscr{D}$. *If* $\mathscr{D}$ *is full-dimensional, then* $y^{kk} \in \mathbb{R}^A$ *is a routing template for* $k \in K$ *and* $f^{0k} \in \mathbb{R}^A, y^{kh} \in \mathbb{R}^A$ *are circulations for every* $k, h \in K$ *with* $k \neq h$.

Just like in the static case, the flow for commodity k changes linearly with d^k on the paths described by the template y^{kk}_a. However, the flow for commodity k may change also if the demand for $h \neq k$ changes which is described by circulations y^{kh}. In addition there is a constant circulation shift described by variables f^{0k}.

As already mentioned, a dynamic routing for commodity k could also be described by one (representative) routing and circulations depending on the demand fluctuations. In the dynamic case however, the circulations can be chosen arbitrarily while in the affine case the actual flow changes according to (7). We illustrate this concept in Example 1 which shows that affine routing can be as good as dynamic routing in terms of the cost for capacity allocation and that f^0 and y^{kh} may not describe circulations when $\mathscr{D}$ is not full-dimensional

Example 1. Consider the network design problem for the graph depicted in Figure 1(a) with two commodities $k_1 : a \to b$ and $k_2 : a \to c$. The uncertainty set $\mathscr{D}$ is defined by the extreme points $d_1 = (2,1), d_2 = (1,2)$ and $d_3 = (1,1)$, and the capacity unitary costs are the edge labels of Figure 1(a). Edge labels from Figure 1(b) and 1(c) represent optimal capacity allocations with static



(a) edge costs (b) static (c) dynamic

(d) $y^{k_1k_1}$ (e) $y^{k_1k_2}$ (f) $y^{k_2k_2}$

Figure 1: Static, dynamic, and affine recourse.

and dynamic routing, respectively. They have costs of 10 and 9, respectively. Then, Figure 1(d)-1(f) describes coefficients y^{kh} for an affine routing feasible for the capacity allocation 1(c). If we remove $d_3 = (1,1)$ from the set of extreme points, the dimension of the uncertainty set reduces to 1. The affine routing prescribed by $y^{k_2k_2}_{ac} = 1$, $f^{0k_1}_{ab} = 3$ and $y^{k_1k_2}_{ab} = -1$ serves all demands in the convex hull of $d_1 = (2,1)$ and $d_2 = (1,2)$ but f^{0k_1} and $y^{k_1k_2}$ do not describe a circulation.

Compact reformulations In the following, we assume that $\mathscr{D}$ is full-dimensional. If the number of vertices of $\mathscr{D}$ is polynomial in the number of nodes, arcs, and commodities then model (RND) can be written in a compact way for all three routing schemes, that is, it can be written with a polynomial number of variables and constraints. However, even if the number of vertices is exponential there are compact reformulations for (RND) with static or affine routing as long as $\mathscr{D}$ is compact. Reformulating by dualizing constraints is a standard technique in robust optimization resulting in so-called *robust counterparts*, see for instance [12]. Applying this technique to (RND) with affine routing yields the following result.

Proposition 3. *Consider (RND) with affine routing for a full-dimensional uncertainty polytope $\mathscr{D}$. If $\mathscr{D}$ has polynomial many vertices or can be described by a polynomial number of inequalities then (RND) can be solved in polynomial time.*

Proposition 3 implies that given a capacity allocation x, the existence of an affine routing can be answered in polynomial time as long as $\mathscr{D}$ can be described in a compact way, which is also true in the static case but is in contrast to the $\mathscr{NP}$-complete results for dynamic routing [3].

Domination of demands For static and dynamic routings, not all extreme points of $\mathscr{D}$ have to be considered in a discretization of $\mathscr{D}$. For instance, if $0 \in \mathscr{D}$, it is an extreme point of $\mathscr{D}$ that any capacity allocation using static (resp. dynamic) routing supports. This intuitive idea has been formalized by Oriolo [13] introducing the concept of domination. Given two demands vectors d_1 and d_2, we say that d_1 *dominates* d_2 if any capacity allocation $x \in \mathbb{R}^A_+$ supporting d_1 also supports d_2 (dynamic routing). Moreover, d_1 *totally dominates* d_2 if any pair (x,y) supporting d_1 also supports d_2 (static routing). Thus, removing dominated (extreme) points from $\mathscr{D}$ is not changing the problem in the static and the in dynamic case.

For general affine routings, however, there is no notion of domination of demands:

Proposition 4. *Let $d_1, d_2 \in \mathbb{R}^K_+$, $d_1 \neq d_2$. There exists (x, f^0, y) that supports d_1 but does not support d_2.*

Relation to static routing Notice that if a flow f^k for k contains a positive circulation, that is, there exists a positive circulation g such that $f^k - g$ is a flow for k then f^k can be reduced to $f^k - g$ without changing the flow balance at $s(k)$ and $t(k)$. Moreover, the percental splitting among the used paths is unchanged. In this spirit we call any routing f *cycle-free* if for all $d \in \mathscr{D}$ and all commodities $k \in K$ the commodity flows do not contain positive circulations. Of course every optimal capacity allocation has a *cycle-free* (static, affine, or dynamic) routing.

Notice that if a flow f^k for k contains a positive circulation, that is, there exists a positive circulation g such that $f^k - g$ is a flow for k then f^k can be reduced to $f^k - g$ without changing the flow balance at $s(k)$ and $t(k)$. Moreover, the percental splitting among the used paths is unchanged. In this spirit we call any routing f *cycle-free* if for all $d \in \mathscr{D}$ and all commodities $k \in K$ the commodity flows do not contain positive circulations. Of course every optimal capacity allocation has a *cycle-free* (static, affine, or dynamic) routing.

Let e^k be the k-th unit vector in $\mathbb{R}^K_+$ and $\mathscr{D}^k_0$ be the set obtained from $\mathscr{D}$ by removing $d \in \mathscr{D}$ with $d^k > 0$, that is, $\mathscr{D}^k_0 := \{d \in \mathscr{D} : d^k = 0\}$. We can prove the following:

Proposition 5. *Let $\mathscr{D}$ be a demand polytope. If $0 \in \mathscr{D}$ and for each $k \in K$ there is $\varepsilon_k > 0$ such that $\varepsilon_k e^k \in \mathscr{D}$, then all cycle-free affine routings serving $\mathscr{D}$ are static.*

Proposition 6. *Let $\mathscr{D}$ be a demand polytope and let G be acyclic. If $\dim(\mathscr{D}^k_0) = |K| - 1$ for all $k \in K$, then all cycle-free affine routings serving $\mathscr{D}$ are static.*

Theorem 7. *Let $\mathscr{D}$ be a demand polytope. If all cycle-free affine routings serving $\mathscr{D}$ are static then $\dim(\mathscr{D}^k_0) = |K| - 1$ for all $k \in K$.*

Combining Proposition 6 with Theorem 7, we have complectly described polytopes for which cycle-free affine routings and static routings are equivalent, assuming that G is acyclic. However, Proposition 6 is wrong for general graphs because $f^k(d)$ for $d \in \mathscr{D}^k_0$ is not necessarily equal to 0, it can also be a positive circulation. Then, one can check that, when G has the required structure, a positive circulation can be decomposed into circulations that are not positive, thus yielding a cycle-free affine routing and a counterexample to Proposition 6.

Relation to dynamic routing Theorem 5 identifies demand polytopes for which affine routing is no better than static routing. However, we saw in Example 1 that affine routing may also perform as well as dynamic routing does, yielding strictly cheaper capacity allocations. For general robust optimization problems, [14] show that affine recourse is equivalent to dynamic recourse when $\mathscr{D}$ is a simplex. Here we show that in the context of robust network design this condition is also necessary.

Theorem 8. *Given a demand polytope $\mathscr{D}$, all dynamic routings serving $\mathscr{D}$ are affine routings if and only if $\mathscr{D}$ is a simplex.*

Example 2 shows that when $\mathscr{D}$ is not a simplex and does not contain the origin, capacity allocation costs required by static, affine, and dynamic routings can be strictly different.

Example 2. Consider the network design problem from Example 1 with the uncertainty set $\mathscr{D}$ defined by the extreme points $d_1 = (3,0)$, $d_2 = (0,3)$, $d_3 = (2,2)$ and $d_4 = (0.5,0.5)$. The optimal capacity allocation costs with static, affine, and dynamic routings are, respectively, $13 + \frac{1}{2}$, $13 + \frac{1}{3}$, and 13. Notice that moving d_4 along the segment $(0,0) - (1,1)$ leaves static and dynamic optimal capacity allocations unchanged while the affine solution cost moves between 13 and $13 + \frac{1}{2}$. In particular, if d_4 is set to $(0,0)$, the affine and static costs are the same, which we knew already from Theorem 5. If d_4 is in $\mathrm{conv}\{d_1, d_2, d_3, (1,1)\}$, the affine and dynamic cots are the same.

4. REFERENCES

[1] A. Ben-Tal and A. Nemirovski, "Robust solutions of linear programming problems contaminated with uncertain data," *Mathematical Programming*, vol. 88, pp. 411–424, 2000.

[2] A. Ben-Tal, A. Goryashko, E. Guslitzer, and A. Nemirovski, "Adjustable robust solutions of uncertain linear programs," *Mathematical Programming*, vol. 99, no. 2, pp. 351–376, 2004.

[3] C. Chekuri, F. B. Shepherd, G. Oriolo, and M. G. Scutellà, "Hardness of robust network design," *Networks*, vol. 50, no. 1, pp. 50–54, 2007.

[4] W. Ben-Ameur, "Between fully dynamic routing and robust stable routing," in *6th International Workshop on Design and Reliable Communication Networks, 2007. DRCN 2007*, 2007.

[5] M. G. Scutellà, "On improving optimal oblivious routing," *Operations Research Letters*, vol. 37, no. 3, pp. 197–200, 2009.

[6] A. Ouorou and J.-P. Vial, "A model for robust capacity planning for telecommunications networks under demand uncertainty," in *6th International Workshop on Design and Reliable Communication Networks, 2007. DRCN 2007*, 2007, pp. 1–4.

[7] S. Mattia, "The robust network loading problem with dynamic routing," La Sapienza, University of Rome, Tech. Rep. Vol 2, n. 3, 2010. [Online]. Available: http://ojs.uniroma1.it/index.php/DIS_TechnicalReports

[8] W. Ben-Ameur and H. Kerivin, "Routing of uncertain demands," *Optimization and Engineering*, vol. 3, pp. 283–313, 2005.

[9] A. Altin, E. Amaldi, P. Belotti, and M. Ç. Pinar, "Provisioning virtual private networks under traffic uncertainty," *Networks*, vol. 49, no. 1, pp. 100–115, 2007.

[10] A. M. C. A. Koster, M. Kutschka, and C. Raack, "Towards robust network design using integer linear programming techniques," in *Proceedings of the NGI 2010*. Paris, France: Next Generation Internet, Jun 2010.

[11] N. Goyal, N. Olver, and F. B. Shepherd, "Dynamic vs. oblivious routing in network design," in *Proceedings of the ESA 2009*, 2009, pp. 277–288.

[12] D. Bertsimas and M. Sim, "The Price of Robustness," *Operations Research*, vol. 52, no. 1, pp. 35–53, Jan 2004.

[13] G. Oriolo, "Domination between traffic matrices," *Mathematics of Operations Research*, vol. 33, no. 1, pp. 91–96, 2008.

[14] D. Bertsimas and V. Goyal, "On the power and limitations of affine policies in two-stage adaptive optimization," Columbia University, USA, Tech. Rep., 2009. [Online]. Available: http://www.columbia.edu/~vg2277/

Solving a Hub Location Problem by the Hyperbolic Smoothing Approach

Adilson Elias Xavier * Claudio Martagão Gesteira * Henrique Pacca Loureiro Luna †

* Federal University of Rio de Janeiro - Brazil
Rio de Janeiro, Brazil
{adilson, gesteira}@cos.ufrj.br

† Federal University of Alagoas - Brazil
Maceio, Brazil
pacca@ic.ufal.br

ABSTRACT

Hub-and-spoke (HS) network designs arise in transportation and telecommunications systems, where one must flow commodities among spatially separate points and where scale economies can be attained through the shared use of high capacity links. As an alternative for the discrete approach of selecting as hubs a subset of the existing nodes, this paper explores the possibility of a continuous location for the hubs. Therefore, the problem is to find the least expensive HS network, continuously locating hubs and assigning traffic to them, given the demands between each origin-destination pair and the respective transportation costs. The problem leads to a $min-sum-min$ formulation that is strongly non-differentiable. The proposed method overcomes this difficulty with a smoothing strategy that uses a special differentiable function. The approach is a particular application of the hyperbolic smoothing technique, which has been proven to be able to solve quite efficiently large instances of clustering problems. The final solution is obtained by solving a sequence of differentiable unconstrained optimization subproblems which gradually approach the original problem. The most important feature of the methodology is the low dimension of the subproblems, dependent only on the number of hubs. The efficiency of the method is shown through a set of computational experiments with large continuous hub-and-spoke problems.

Keywords: Hub Location, Min-Sum-Min Problems, Global Optimization, Non-differentiable Programming, Hyperbolic Smoothing

1. INTRODUCTION

The hierarchical organization of telecommunication and transportation systems can be found in several real world applications, such as the location of switching centers or postal offices, and plays a major role in operations research and management science models. Cost minimization is the objective of most of these models and optimized levels of customer concentrations enables the economies of scale of aggregating the flows in the related networks. The main differences among the models concern the hierarchical level of network design, typically backbone versus local access network, and how the relevant aspects of connectivity, capacity, reliability, demand patterns, routing, pricing, performance and quality of service are considered for such networks[1, 2, 3]. Depending on the context or application, hub nodes are called switches, warehouses, facilities, concentrators or access points. Likewise, backbones may be referred to as hub-level networks and local access networks may be called tributary networks or many other names. Normally, backbone links carry larger volumes of traffic than tributary links. Traffic originating at a specific customer location can pass through a local access network to get to one or more hub nodes, depending on whether single or multiple assignmenst are considered to link the backbone to the remote locations. After passing through the backbone network, the traffic again uses a local access network to travel from a hub to its final destination at another location.

2. THE CONTINUOUS HUB-AND-SPOKE PROBLEM SPECIFICATION

The continuous hub-and-spoke problem consists of locating a set of q centers or hubs in order to minimize a particular trans- portation cost function. To formulate this problem, we proceed as follows. Let $S = \{s_1, \ldots, s_m\}$ denote a given set of m cities or points in a planar region. Let d_{jl} be the flow between two points j and l. Let $x_i, i = 1, \ldots, q$ where each $x_i \in \mathbb{R}^2$, be the set of variables of the problem: the hubs or centers location. The set of these hubs are represented by $X \in \mathbb{R}^{2q}$, and the assumption is that each pair of hubs is directly connected by the shortest distance route between them.

Concerning the hub-and-spoke problem under consideration, the connections between each pair of points j and l, have always three parts: from the origin point j to a first hub center i_1, from i_1 to a second hub center i_2 and from i_2 to destination point l. There are no network structure constraining connections, the only constraint being that connections between cities must be done through hubs. However, the first and the second hubs can be coincident (i.e., $i_1 = i_2$), meaning that a unique hub is used to connect the origin point j and the destination point l. Multiple allocation is permitted, meaning that any given point can be served by one or more hubs.

The unitary flow cost associated to a general connection (j, i_1, i_2, l) is equal to a weighted distance obtained by the sum of three Euclidian distances, with a reduction factor for the second part between hubs:

$$z_{ji_1i_2l} = \|s_j - x_{i_1}\|_2 + \alpha \|x_{i_1} - x_{i_2}\|_2 + \|x_{i_2} - s_l\|_2, \qquad (1)$$

where α is the reduction factor: $0 \leq \alpha < 1$.

The unitary flow cost from the origin point j to the destination point l is taken as the minimum value of all connections:

$$z_{jl} = \min_{i_1, i_2 = 1, \ldots, q} z_{ji_1i_2l}, \qquad (2)$$

or

$$z_{jl} = \min_{i_1, i_2 = 1, \ldots, q} \|s_j - x_{i_1}\|_2 + \alpha \|x_{i_1} - x_{i_2}\|_2 + \|x_{i_2} - s_l\|_2. \qquad (3)$$

3. SMOOTHING THE CONTINUOUS HUB-AND-SPOKE PROBLEM

The continuous hub-and-spoke problem consists of minimizing the total flow cost between all pairs of cities taking the unitary cost value, given by (2), for all connections:

$$\begin{aligned} \min \ & \sum_{j=1}^{m}\sum_{l=1}^{m} d_{jl} z_{jl} \\ \text{subject to} \quad & z_{jl} = \min_{i_1,i_2=1,\ldots,q} z_{ji_1i_2l}, \qquad j,l=1,\ldots,m. \end{aligned} \tag{4}$$

So, this hub-and-spoke problem has a structure named $min-sum-min$, with nondifferentiable and nonconvex characteristics, having a myriad of local minimizers. A series of transformations will be performed in order to obtain a continuous formulation. First, considering its definition, each z_{jl} must necessarily satisfy the following set of inequalities:

$$z_{jl} - z_{ji_1i_2l} \leq 0, \qquad i_1,i_2 = 1,\ldots,q. \tag{5}$$

Substituting these inequalities for the equality constraints of problem (4), the relaxed problem becomes

$$\begin{aligned} \min \ & \sum_{j=1}^{m}\sum_{l=1}^{m} d_{jl} z_{jl} \\ \text{subject to} \quad & z_{jl} - z_{ji_1i_2l} \leq 0, \\ & \qquad i_1,i_2 = 1,\ldots,q; \quad j,l=1,\ldots,m. \end{aligned} \tag{6}$$

Since the variables z_{jl} are not bound from below, in order to obtain the desired equivalence, we must modify problem (6). We do so by first letting $\varphi(y)$ denote $max\{0,y\}$ and then observing that, from the set of inequalities in (6), it follows that

$$\sum_{i_1=1}^{q}\sum_{i_2=1}^{q} \varphi(z_{jl} - z_{ji_1i_2l}) = 0, \qquad j,l=1,\ldots,m. \tag{7}$$

Using (7) in place of the set of inequality constraints in (6), we would obtain an equivalent problem maintaining the undesirable property that $z_{jl},\ j,l=1,\ldots,m$ still has no lower bound. Considering, however, that the objective function of problem (6) will force each $z_{jl},\ j,l=1,\ldots,m,$ downward, we can think of bounding the latter variables from below by including an ε perturbation in (7). So, the following modified problem is obtained:

$$\begin{aligned} \min \ & \sum_{j=1}^{m}\sum_{l=1}^{m} d_{jl} z_{jl} \\ \text{subject to} \quad & \sum_{i_1=1}^{q}\sum_{i_2=1}^{q} \varphi(z_{jl} - z_{ji_1i_2l}) \geq \varepsilon, \qquad j,l=1,\ldots,m, \end{aligned} \tag{8}$$

for $\varepsilon > 0$. Since the feasible set of problem (4) is the limit of that of (8) when $\varepsilon \to 0_+$, we can then consider solving (4) by solving a sequence of problems like (8) for a sequence of decreasing values for ε that approaches 0.

Analyzing problem (8), the definition of function φ endows it with an extremely rigid nondifferentiable structure, which makes its computational solution very hard. In view of this, the numerical method we adopt for solving problem (1), takes a smoothing approach. From this perspective, let us define the function:

$$\phi(y,\tau) = \left(y + \sqrt{y^2+\tau^2}\right)/2 \tag{9}$$

for $y \in \mathbb{R}$ and $\tau > 0$.

Function ϕ has the following properties:

(a) $\phi(y,\tau) > \varphi(y), \quad \forall\, \tau > 0$;

(b) $\lim_{\tau\to 0} \phi(y,\tau) = \varphi(y)$;

(c) $\phi(.,\tau)$ is an increasing convex C^∞ function in variable y.

Therefore, function ϕ constitutes an approximation of function φ. By using function ϕ in the place of function φ, in (8), the problem

$$\begin{aligned} \min \ & \sum_{j=1}^{m}\sum_{l=1}^{m} d_{jl} z_{jl} \\ \text{subject to} \quad & \sum_{i_1=1}^{q}\sum_{i_2=1}^{q} \phi(z_{jl} - z_{ji_1i_2l}, \tau) \geq \varepsilon, \qquad j,l=1,\ldots,m, \end{aligned} \tag{10}$$

is produced.

To obtain a differentiable problem, it is necessary further to smooth the balanced distances $z_{ji_1i_2l}$. For this purpose, let us define the function

$$\theta(v,w,\gamma) = \sqrt{(w^1-v^1)^2 + (w^2-v^2)^2 + \gamma^2} \tag{11}$$

where $v,w \in \mathbb{R}^2$ and $\gamma > 0$.

Function θ has the following properties:

(a) $\lim_{\gamma\to 0} \theta(v,w,\gamma) = \|w-v\|_2$;

(b) θ is a C^∞ function.

By using function θ in place of the Euclidian distances, the completely differentiable problem

$$\begin{aligned} \min \ & \sum_{j=1}^{m}\sum_{l=1}^{m} d_{jl} z_{jl} \\ \text{subject to} \quad & \sum_{i_1=1}^{q}\sum_{i_2=1}^{q} \phi(z_{jl} - (\theta(s_j, x_{i_1},\gamma) + \\ & \alpha\,\theta(x_{i_1}, x_{i_2},\gamma) + \theta(x_{i_2}, s_l,\gamma)),\tau) \geq \varepsilon, \qquad j,l=1,\ldots,m, \end{aligned} \tag{12}$$

is now obtained.

So, the properties of functions ϕ and θ allow us to seek a solution to problem (8) by solving a sequence of subproblems like problem (12), produced by decreasing the parameters $\gamma \to 0$, $\tau \to 0$, and $\varepsilon \to 0$.

Since $z_{jl} \geq 0,\ j,l=1,\ldots,m,$ the objective function minimization process will work towards reducing these values to the utmost. On the other hand, given any set of hubs $x_i, i=1,\ldots,q,$ due to property (c) of the hyperbolic smoothing function ϕ, the constraints of problem (12) are a monotonically crescent function in z_{jl}. So, these constraints will certainly be active and problem (12) will ultimately be equivalent to the problem:

$$\min \sum_{j=1}^{m}\sum_{l=1}^{m} d_{jl} z_{jl} \tag{13}$$

$$\text{subject to } h_{jl}(z_{jl},x) = \sum_{i_1=1}^{q}\sum_{i_2=1}^{q} \phi(z_{jl} - (\theta(s_j,x_{i_1},\gamma) + \alpha\,\theta(x_{i_1},x_{i_2},\gamma) + \theta(x_{i_2},s_l,\gamma)),\tau) - \varepsilon = 0, \quad j,l = 1,\ldots,m.$$

The dimension of the variable domain space of problem (13) is $(2q+m^2)$.

Since, in general, the value of the parameter m, the cardinality of the set S of the consumer points s_j, is large, problem (13) has a large number of variables. However, it has a separable structure, because each variable z_{jl} appears only in one equality constraint. Therefore, as the partial derivative of $h_{jl}(z_{jl},x)$ with respect to z_{jl}, $j,l = 1,\ldots,m$ is not equal to zero, it is possible to use the Implicit Function Theorem to calculate each component z_{jl}, $j,l = 1,\ldots,m$ as a function of the hub location variables $x_i, i = 1,\ldots,q$. In this way, the unconstrained problem

$$\min\ f(x) = \sum_{j=1}^{m}\sum_{l=1}^{m} d_{jl} z_{jl}(x) \tag{14}$$

is obtained, where each $z_{jl}(x)$ results from the calculation of a zero of each equation

$$h_{jl}(z_{jl},x) = \sum_{i_1=1}^{q}\sum_{i_2=1}^{q} \phi(z_{jl} - (\theta(s_j,x_{i_1},\gamma) + \alpha\,\theta(x_{i_1},x_{i_2},\gamma) + \theta(x_{i_2},s_l,\gamma)),\tau) - \varepsilon = 0, \quad j,l = 1,\ldots,m. \tag{15}$$

Due to property (c) of the hyperbolic smoothing function, each term ϕ above is strictly increasing with variable z_{jl} and therefore the equation has a single zero. Again, due to the Implicit Function Theorem, the functions $z_{jl}(x)$ have all derivatives with respect to the variables $x_i, i = 1,\ldots,q,$, and therefore it is possible to calculate exactly the gradient of the objective function of problem (14),

In this way, it is easy to solve problem (14) by making use of any method based on first order derivative information. Finally, it must be emphasized that problem (14) is defined on a $(2q)-$dimensional space, so it is a small problem, since the number of hubs, q, is, in general, small for real world applications.

The solution of the original hub-and-spoke problems can thus be obtained by using the Hyperbolic Smoothing Hub-and-Spoke Algorithm, described below in a simplified form.

Simplified HSHS Algorithm

Initialization Step: Choose initial values: x^0, γ^1 , τ^1 , ε^1.

Choose values $0 < \rho_1 < 1$, $0 < \rho_2 < 1$, $0 < \rho_3 < 1$; let $k = 1$.

Main Step: Repeat until a stopping rule is attained

Solve problem (14) with $\gamma = \gamma^k$, $\tau = \tau^k$ and $\varepsilon = \varepsilon^k$, starting at the initial point x^{k-1} and let x^k be the solution obtained.

Let $\gamma^{k+1} = \rho_1\,\gamma^k$, $\tau^{k+1} = \rho_2\,\tau^k$, $\varepsilon^{k+1} = \rho_3\,\varepsilon^k$, $k := k+1$. ■

Just as in other smoothing methods, the solution to the hub-and-spoke problem is obtained, in theory, by solving an infinite sequence of optimization problems. In the HSHS algorithm, each problem that is minimized is unconstrained and of low dimension. Notice that the algorithm causes τ and γ to approach 0, so the constraints of the subproblems it solves, given as in (12), tend to those of (8). In addition, the algorithm causes ε to approach 0, so, in a simultaneous movement, the problem (8) gradually approaches problem (4).

4. COMPUTATIONAL RESULTS

The computational results presented below were obtained from a preliminary implementation. The numerical experiments have been carried out on a PC Intel Celeron with a 2.7GHz CPU and 512MB RAM. The programs were coded with Compac Visual FORTRAN, Version 6.1. The unconstrained minimization tasks were carried out by means of a Quasi-Newton algorithm, employing the BFGS updating formula from the Harwell Library. The initial starting hubs $x_i^0, i = 1,\cdots,q$ are taken around the center of gravity of the set of points, by making random perturbations proportional to the standard deviation of this set. The value of τ^1 was taken as $1/100$ of this standard deviation. The following choices were made for the other parameters: $\varepsilon^1 = 4\,\tau^1$, $\gamma^1 = \tau^1/100$, $\rho_1 = 1/4$, $\rho_2 = 1/4$ and $\rho_3 = 1/4$.

In order to show the performance of the proposed algorithm, results obtained by using the German Towns instane, which uses the two Cartesian coordinates of 59 towns, originally presented by [4]. The instance reported here presents a symmetric demand matrix, with a required flow of one unit between all pairs of origin and destination cities: $d_{jl} = d_{lj} = 1$, $j,l = 1,\cdots,m$. So, problem (14) assumed the following formulation, further simplified:

$$\textit{minimize } f(x) = \sum_{j=1}^{m-1}\sum_{l=j+1}^{m} z_{jl}(x). \tag{16}$$

Table 1 presents the obtained computational results. Ten different randomly chosen starting points were used for each instance. The discount parameter has been fixed in $\alpha = 0,5$ The first column presents the specified number of hubs (q). The second column presents the best objective function value (f_{HSHS}) produced by the HSHS algorithm. The next three columns present the number of occurrences of the best solution $(Occ.)$, the average percentage error of the 10 solutions (E_{Mean}) in relation to the best solution obtained (f_{HSHS}) and the CPU mean time given in seconds (T_{Mean}). By defining f^r as the value of the objective function obtained at the starting point r, the percentage error is calculated by the expression:

$$E_{Mean} = 10 \sum_{r=1}^{10} (f^r - f_{HSHS})\,/\,f_{HSHS}. \tag{17}$$

q	f_{HSHS}	$Occur.$	E_{Mean}	T_{Mean}
2	0.171285E6	10	0.00	0.53
3	0.154629E6	6	0.01	2.48
4	0.139158E6	9	0.75	8.47
5	0.131453E6	4	2.03	18.23
6	0.126496E6	1	0.68	39.30
7	0.122636E6	2	0.73	76.28
8	0.119239E6	1	0.81	149.52
9	0.116583E6	1	1.25	246.99
10	0.113962E6	1	1.39	383.56

Table 1: Results for the German Towns Instance ($\alpha = 0.5$)

5. CONCLUSIONS

This paper shows how a preliminary implementation of the proposed algorithm is able to efficiently produce reliable and deep local minima. The motivation to solve large scale versions of continuous hub-and-spoke problems stems from, among other real world applications, aerial transportation and oil and gas off-shore exploration. We believe that this article's methodology is adequate enough for the requirements of such relevant applications.

6. REFERENCES

[1] M.E. O'Kelly and H.L. Miller, "The hub network design problem: A review and synthesis," *Journal of Transport Geography*, vol. 2, pp. 31–40., 1994.

[2] J. Klincewicz, "Hub location in backbone/tributary network design: A review," *Location Science*, vol. 6, pp. 337–335, 2001.

[3] H. Luna, *Network Planning Problems in Telecommunications*. M.G.C. Rezende and P.M. Pardalos (editors), Springer, New York, 2006, ch. Handbook of Optimization in Telecommunications, pp. 213–240.

[4] H. Späth, *Cluster Analysis Algorithms for Data Reduction and Classification*. Ellis Horwood, Upper Saddle River, NJ., 1980.

A hybrid method to solve a multi-product, multi-depot vehicle routing problem arising in a recyclable waste collection system

Tania Rodrigues Pereira Ramos [*] [‡] Maria Isabel Gomes [†] Ana Paula Barbosa-Povoa [‡]

[*] Instituto Universitario de Lisboa (ISCTE-IUL)
Avenida das Forcas Armadas, Edificio ISCTE, 1649-026 Lisboa, Portugal
tania.ramos@iscte.pt

[†] CMA - FCT, Universidade Nova Lisboa
Campus da Caparica, 2829-516 Caparica, Portugal
mirg@fct.unl.pt

[‡] CEG-IST, Universidade Tecnica de Lisboa
Avenida Rovisco Pais, 1049-001 Lisboa, Portugal
apovoa@ist.utl.pt

ABSTRACT

The present work aims to support tactical and operational decisions in recyclable waste collection systems, focusing on the delimitation of service areas in systems with more than one depot, and on vehicle routes definition. The problem is modelled as a multi-product, multi-depot vehicle routing problem. Due to problem solution complexity, a hybrid method based on two mathematical formulations and one heuristic procedure is developed as a solution method. The method proposed is applied to a large scale problem based on a real case study of a recyclable waste collection system, where three types of recyclable materials have to be collected.

Keywords: Multi-depot, Vehicle routing, Hybrid method, Recyclable waste collection system

1. INTRODUCTION

The present work aims to support tactical and operational decisions in recyclable waste collection systems with more than one depot, helping the decision making on the system delimitation of service areas and on the vehicle routes definition. When characterizing the recyclable waste collection system in study it can be said that this is responsible to collect, within a certain geographic area and in a regular basis, three types of recyclable materials used in packaging (paper, glass and plastic/metal) dropped by the final consumer into special containers. When these systems have more than one depot, in addition to the definition of the vehicle routes, it is also necessary to decide from which depot the collection is to be performed. This problem is modelled as a multi-product, multi-depot vehicle routing problem. A hybrid method is developed where a MIP solver is embedded in a heuristic framework. The hybrid method is applied to a large scale problem based on a real recyclable waste collection system.

2. LITERATURE REVIEW

MDVRP consists on defining a set of vehicle routes in such a way that: (1) each route starts and ends at the same depot, (2) each customer is visited exactly once by a vehicle, (3) the total demand of each route does not exceed the vehicle capacity, (4) the total duration of each route (including travel and service times) does not exceed a preset limit and (5) the total routing cost is minimized. For the MDVRP, there are several models developed (exact and approximate approaches). Due to its NP-hard combinatorial factor, the models presented in the literature are mostly heuristics-based and few exact algorithms cab be found in the literature. Laporte et al. [1], as well as Laporte et al. [2], developed exact branch and bound algorithms for solving the symmetric and asymmetric version of the MDVRP, respectively. Recently, Baldacci and Mingozzi [3] developed an exact method for solving the Heterogeneous Vehicle Routing Problem (HVRP) that is also capable to solve, among other problems, the MDVRP. This algorithm is based on the set partitioning formulation, where a procedure is applied to generate routes. Three bounding procedures are used to reduce the number of formulation variables. As for the approximate methods there are several heuristic algorithms developed for the MDVRP (Tillman and Cain [4], Golden, Magnanti and Nguyen [5], Renaud et al.[6], Salhi and Sari [7], Lim and Wang [8], Crevier et al. [9], among others). Based on the literature review, we can conclude that few exact models for the multi-depot problems exist, while several heuristic procedures have been developed for the same problem. The combination of these two methods is also not well explored. Therefore, this work studies this opportunity and proposes a hybrid method which combines an exact formulation with heuristic procedures to solve the multi-product, multi-depot vehicle routing problem.

3. HYBRID METHOD

In Figure 1 a schematic diagram of the hybrid method proposed is shown. This involves three main steps.

The first step involves the relaxation of the Multi-Product, Multi-Depot VRP (with more than one product and vehicle routes restricted to start and finish at the same depot) into the Single-Product, Multi-Depot VRP with Multi-Depot Routes (with just one product and multi-depot routes allowed). By solving this model, we obtain some collection sites that belongs to feasible routes for the Single-Product, Multi-Depot VRP, meaning that they belong to a route that starts and finishes at the same depot; and some other collection sites whose route starts and finishes at different depots. For the "feasible" collection sites, we fix their assignment to the

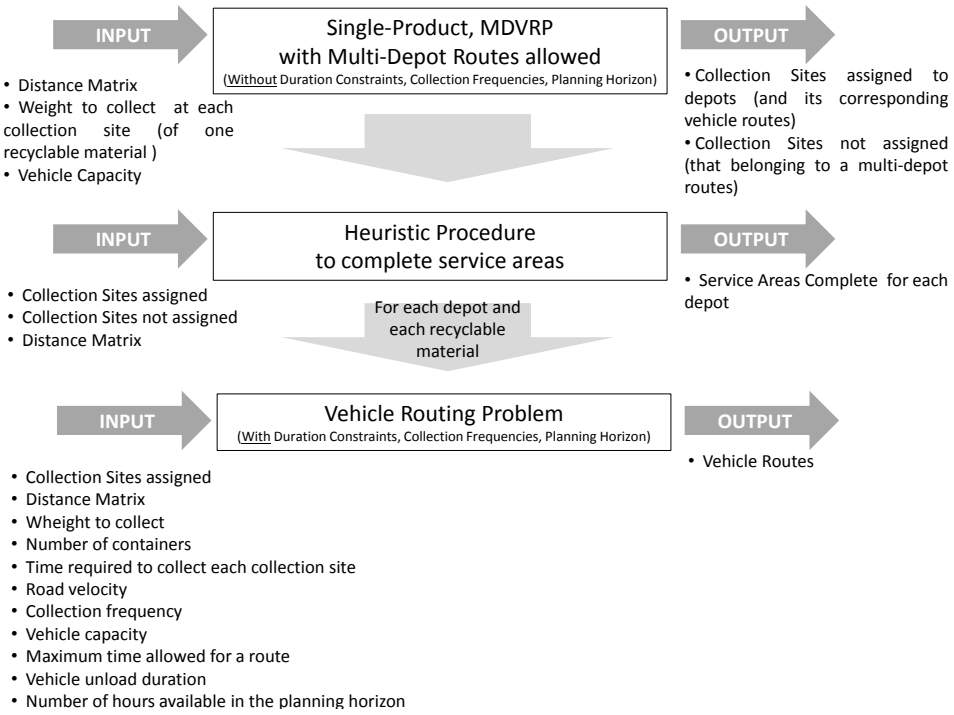


Figure 1: *Structure of the proposed hybrid method.*

depot (not to a particular route or vehicle, the assignment is done just to the depot) and then the a heuristic procedure at step two the procedure is run. This assigns the remaining collection sites and therefore completes the service areas by depot.

After the service areas being defined, an exact formulation is run to solve the Vehicle Routing Problem for each depot and for each recyclable material (third step). The relaxed constraints ,on the first step, are here considered: namely, the duration, the recyclable materials collection frequencies and the planning horizon constraints.

In the hybrid method, the service areas are established by the results obtained for one single recyclable material, at the first module. This module is also run for the other two recyclable materials to assess which one produces the best solution regarding the minimum total distance travelled. In order to provide further insights into the above steps, these will next be described with greater detailed and with supporting references.

1. Single-Product, MDVRP with Multi-Depot Routes
   In Multi-Depot VRP, vehicles are restricted to start and finish at the same depot. This can be relaxed and multi-depot routes are allowed, while minimizing the total distance travelled. The Multi-Depot Vehicle Routing Problem with Multi-Depot Routes has not received much attention from researchers. A similar problem is presented by Crevier et al. [9], entitled *Multi-Depot Vehicle Routing Problem with Inter-Depot Routes*, where inter-depot routes, that connect two different depots, are allowed. In this case, depots can act as intermediate replenishment facilities along the route of a vehicle, but the rotation of a vehicle always starts and ends at the same depot (the authors called "rotation" to the set of all routes assigned to a vehicle). In the Multi-Depot Vehicle Routing Problem with Multi-Depot Routes, the rotation concept doesnŽt exist since the vehicles donŽt have to return to their starting depot. The vehicle routes can then be Hamiltonian cycles or just paths between two depots.

   Our proposed formulation for the Multi-Depot VRP with Multi-Depot Routes is based on the two-commodity flow formulation for the CVRP, introduced by Baldacci et al. [10]. This formulation considers one real depot and one copy depot, and all vehicle fleet has to be used. In the proposed formulation instead of one real and one copy depot, we have a set of real depots and a set of copy depots, and we do not impose that all vehicles are to be used.

   Since this formulation intends to be a simplification of the master problem, multi-depot routes are allowed and time constraints are not taken into account. We, therefore, relax the maximum time allowed for a route and the maximum time available over the timeframe. As input data, this module requires the distance between each node (collection sites and depots), the weight to be collected at each collection site (considering only one recyclable material) and the vehicle fleet capacity. The output will be a set of collection routes, where some routes start and end at the same depot (feasible routes) and some start and end at different depots (infeasible routes). This module is run for each recyclable material, in order to find the best solution regarding the minimum total distance travelled.

2. Heuristic Procedure to Complete Service Areas
   As mentioned above, the first module generates uncompleted service areas by depot. The aim of this second module is to complete those service areas by assigning to depots the collection sites that, in the previous module, were associated to infeasible routes. The assignment is done through a greedy heuristic rule where the collection site is assigned to the nearest service area.

3. Vehicle Routing Problem
   After phase two, service areas by depot are already defined. It is now necessary to solve a vehicle routing problem for each depot to accomplish the multi-depot vehicle routing problem. The mathematical formulation used to solve the VRP is based on the two-commodity flow formulation [10], taking into account the collection frequencies of the recyclable materials, the route duration limit and the number of hours available in the planning horizon.

4. APPLICATION TO A LARGE SCALE PROBLEM

The procedure developed is applied to a large scale problem, extracted from a real case study. This instance problem has 100 collection sites and 3 depots. We considered that each depot has one vehicle that could make several trips over the timeframe. The timeframe has four weeks and the collection frequencies considered for each recyclable material are once a month for glass, twice a month for plastic and once a week for paper. We do not set a limit to the number of trips that a vehicle can do over the timeframe, but the number of hours that a vehicle can work over the timeframe is limited to 160 hours (4 weeks $\times$ 5 days per week $\times$ 8 hours per day).

The first and third modules of the hybrid method are solved using the branch-and-bound method implemented in the solver of the CPLEX Optimizer 12.1.0. The branch-and-bound computation time is arbitrarily limited to 2 hours. The second module, with the heuristic procedure, was written in MATLAB. An Intel(R) Core (TM) i7 CPU 930 @ 2.80 GHz is used.

The model will produce three solutions regarding service areas by depot: the first one considers glass results from the first and second module; the second one considers paper results and the third one considers plastic/metal results. The solution that minimizes the total distance travelled is the one developed for the plastic/metal material (Figure 2).

5. CONCLUSIONS

This work studies the multi-product, multi-depot vehicle routing problem and proposes a hybrid method based on two mathematical formulations and one heuristic procedure. This is justified by the high complexity associated to this kind of problems. The method proposed is applied to a large scale problem based on a real case study where a recyclable waste collection system that collects three types of recyclable materials is considered. The existence of multiple depots in such problems requires a service area definition by depot. Each depot is responsible to collect a set of collection sites (which have the three recyclable materials to be collected) and to define the collection routes by recyclable material. To accomplish the service areas by depot, the hybrid method produces three solutions considering independently paper, glass and plastic/metal results. Service areas based on plastic/metal results revealed to produce the minimum total distance to be travelled.

As future work, the heuristic procedure will be improved and the hybrid method will be applied to the real case study with 212 collection sites and 5 depots.

6. REFERENCES

[1] G. Laporte, Y. Nobert, and D. Arpin, "Optimal solutions to capacitated multi-depot vehicle routing problems," *Congressus Numerantium*, vol. 44, pp. 283–292, 1984.

[2] G. Laporte, Y. Nobert, and S. Taillefer, "Solving a family of multi-depot vehicle-routing and location-routing problems," *Transportation Science*, vol. 22, no. 3, pp. 161–172, 1988.

[3] R. Baldacci and A. Mingozzi, "A unified exact method for solving different classes of vehicle routing problems," *Mathematical Programming*, vol. 120, no. 2, pp. 347–380, 2009.

[4] F. Tillman and T. Cain, "Upperbound algorithem for single and multiple terminal delivery problem," *Management Science Series a-Theory*, vol. 18, no. 11, pp. 664–682, 1972.

[5] B. Golden, T. Magnanti, and H. Nguyen, "Implementing vehicle routing algorithms," *Networks*, vol. 7, no. 2, pp. 113–148, 1977.

[6] J. Renaud, G. Laporte, and F. Boctor, "A tabu search heuristic for the multi-depot vehicle routing problem," *Computers and Operations Research*, vol. 23, no. 3, pp. 229–235, 1996.

[7] S. Salhi and M. Sari, "A multi-level composite heuristic for the multi-depot vehicle fleet mix problem," *European Journal Operational Research*, vol. 103, no. 1, pp. 397–402, 1997.

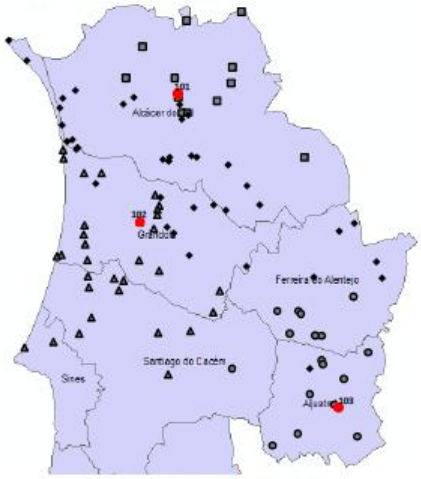
[8] A. Lim and F. Wang, "Multi-depot vehicle routing problem: A one-stage approach," *IEEE Trans Autom Sci Eng.*, vol. 2, no. 4, pp. 397–402, 2005.

[9] B. Crevier, J. Cordeau, and G. Laporte, "The multi-depot vehicle routing problem with inter-depot routes," *European Journal Operational Research*, vol. 176, no. 2, pp. 756–773, 2007.

[10] R. Baldacci, E. Hadjiconstantinou, and A. Mingozzi, "An exact algorithm for the capacitated vehicle routing problem based on a two-commodity network flow formulation," *Operations Research*, vol. 52, no. 5, pp. 723–738, 2004.

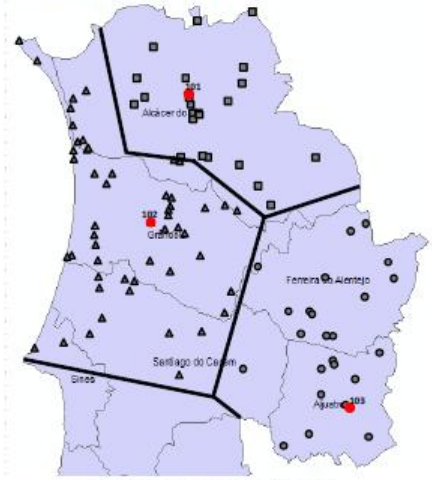
1) Single-Product, MDVRP with Multi-Depot Routes allowed:

N. Collection Sites assigned to Depot 101: 12
N. Collection Sites assigned to Depot 102: 32
N. Collection Sites assigned to Depot 103: 17
N. Collection Sites not assigned: 39



2) Heuristic procedure to complete service areas:

N. Collection Sites assigned to Depot 101: 23
N. Collection Sites assigned to Depot 102: 53
N. Collection Sites assigned to Depot 103: 24
N. Collection Sites not assigned: 0



3) Vehicle Routing Problem for each depot and each recyclable material, with duration constraints:

	Glass		Paper		Plastic/Metal		Total Distance
	Opt.Value	GAP	Opt.Value	GAP	Opt.Value	GAP	Travelled
Depot 101	247	0%	949	0%	574	0%	*1770 Km*
Depot 102	440	10%	1377	1%	821	7%	*2638 Km*
Depot 103	326	0%	1115	0%	792	0%	*2233 Km*
Total	*1013 Km*	-	*3441 Km*	-	*2187 Km*	-	*6641 Km*

Figure 2: *Results obtained for each module of the hybrid method, considering plastic/metal results to define service areas.*

Design and Planning of Supply Chains with Integrated Forward and Reverse Decisions

Sónia R. Cardoso * Ana Paula F. D. Barbosa-Póvoa * Susana Relvas *

* CEG-IST, UTL
Av. Rovisco Pais, 1049-001 Lisboa
{sonia.cardoso, apovoa, susanaicr}@ist.utl.pt

ABSTRACT

Markets increasing competition, coupled with a growing concern with the environment has created a need to increase supply chains' sustainability. To achieve this, the supply chain should integrate reverse logistics activities. In this paper, a mixed integer linear programming formulation is developed for the design and planning of supply chains while considering simultaneously production and reverse logistics activities with the goal of maximizing the net present value. The model is applied to a case study where forward and reverse activities are considered. A sensitivity analysis is performed in order to assess the resulting changes on the optimal solution.

Keywords: Reverse Logistics, Optimisation, Design, Planning

1. INTRODUCTION

Markets increasing competition, coupled with a growing concern with the environment, has created a new way of thinking when designing and planning supply chains. A need to increase supply chains' sustainability is emerging. To achive this, companies must invest in the design and operation of these systems in order to reduce their ecological footprint [1]. Therefore, the supply chain should be now seen as a close loop system [2] where reverse logistics activities are included encompassing the transportation and reprocessing of collected products. Investing in reverse logistics allows the achievement of cost savings in the procurement, disposal and transportation [3]. However, the establishment of a reverse network that is independent of the forward can increase infrastructure costs and can reduce the potential profit associated with remanufacturing [4]. So, for a better network design and planning it is necessary to consider simultaneously the forward and reverse flows. Several studies were published in this area, such as [5] who analyzed reverse logistics concluding that the research on this subject focuses only on separate aspects and there is no holistic analysis of the supply chain. As stated by [6] very few models combine, within a single formulation both forward and reverse flows and much less works consider the integration of the reverse chain as identified by [7] and [8]. Corroborating such conclusions, [9] mention that the number of published works where both forward and reverse flows are taken into account simultaneously is less than the ones that treat them separately. These authors present a generic model for the design and planning of supply chains and state that there are several research opportunities in this area. Thus, it is possible to conclude about the importance of the development of a model that deals with both flows at the same time in a realistic way.

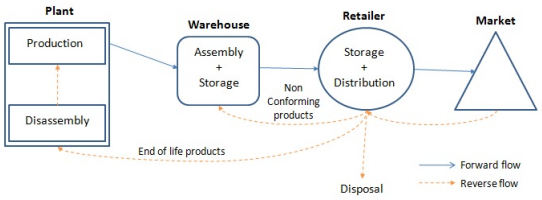


Figure 1: *Network representation.*

2. METHODOLOGY

In this paper, it is proposed an optimization model for the simultaneous design and planning of supply chains with forward and reverse flows, given a certain time horizon. The model representation uses Mixed-Integer Linear Programming (MILP) formulation and was developed based on the work by [10]. These authors studied the design and planning of supply chains, taking into account economical and environmental aspects. However, they only consider a three echelon supply chain and did not incorporate reverse logistics. Such model is generalized in the current work.

The problem addressed in this work has the objective to determine the configuration of the network along with planning decisions that maximize the net present value. The forward supply chain in study is formed by four echelons: plants with a set of available processes to be installed; warehouses where final products are assembled and stored; and retailers which are responsible to deliver the final products to markets, final level of the forward supply chain. It is also considered that factories can exchange all type of products amog them, including raw materials or intermediate products. In the reverse flow, products are sent from clients to retailers and then are sorted. Products that are too damaged are sent to disposal while products in their end of life are sent to factories to be disassembled to be reprocessed and non-conforming products are sent to warehouses to be repacked. In figure 1, it is possible to see a representation of the considered network.

The model determines the number, location and capacity of the processes that have to be installed in each plant, warehouses and retailers in order to maximize the net present value of the supply chain. It also allows to define the best production rates at the plants, forward and reverse flows between all nodes of the network, establishment of transportation links between all entities, inventory levels at warehouses and retailers. At the same time, it has to respect some constraints, namely mass balances at each node of the network, the material flows between each pair of entities have to be within the allowed boundaries, the capacity of each infrastructure cannot be exceeded and the products' bill of materials must be complied.

The model is applied to one example to show its applicability. This example is run for two different cases. The first case includes only

the forward flow while the second case analyses a supply chain where reverse logistics activities are included. For these examples the number of variables varies from 18119 for the first case to 163463 for the second. Although, and as expected, it was observed an increase in the problem complexity when introducing the reverse flow. Nevertheless, this fact did not resulted into difficulties at the solution level, the solution gap obtained was zero in both cases and the computational time was less than two seconds.

Since some assumptions on the example parameters were considered, a sensitivity analysis on the most critical parameters was performed. This allow us to assess the resulting changes on the optimal solution, infrastructures capacities and on other planning decisions.

3. FINDINGS

The MILP is developed for the design and planning of supply chains while considering simultaneously production and reverse logistics activities. The final results present details on the production levels, forward and reverse flow of products, inventory levels, establishment of transportation links and infrastructures capacities. The results obtained for the two situations analyzed are compared and it is possible to conclude that the inclusion of the reverse flows with the associated reverse logistics activities in the supply chain allows the achievement of a better net present value.

Furthermore, in terms of the supply chain robustness the structure designed appear quite robust since following a sensitivity analysis performed on the two most critical parameters, the minimum percentage of collection of end of life products and the percentage of demand satisfaction, no significant changes in the network structure were observed.

The MILP model was implemented in GAMS language, version 22.8, and solved using an IBM-ILOG's CPLEX branch and bound algorithm, version 11.0, in an Intel(R) Core(TM) i7 CPU 2.80 GHz computer with 6 GB RAM.

4. CONCLUSION

In this work it is proposed an optimization model for the design and planning of a four echelon supply chain with forward and reverse flows allowing for the simultaneous incorporation of both production and delivery of products as well as reverse logistics activities dealing with the recovery of non conforming products as well as of products in its end of life time. The model application shows that reverse logistics incorporation may result in an increment of economical benefits associated with environmental concerns, fact that can be seen as a business opportunity. As future work, it is intended to incorporate risk and environmental issues in the model with the goal of maximizing the NPV and simultaneously minimizing risk and environmental impacts. In addition, we also aim to apply the developed model to different types of supply chains so as to demonstrate its applicability to real case studies.

5. REFERENCES

[1] Barbosa-Póvoa, A., Salema, M. and Novais, A., *Design and Planning of Closed-Loop Supply Chains. In Supply Chain Optimization*, Editores L. Papageorgiou and M. C. Georgiadis, Wiley-VCH, Germany, 7, 187-218, 2007.

[2] Guide, D. and Van Wassenhowe, L., "The Reverse Supply Chain*", Harvard Business Review, 80(2), 25-26, 2002.*

[3] Krikke, H., Bloemhof-Ruwaard, J. and Van Wassenhowe, L., "Concurrent product and closed-loop supply chain design with an application to refrigerators", *International Journal of Production Research*, 41, 3689-719, 2003.

[4] Uster, H., Easwaran, G. and Çetinkaya, E., "Benders decomposition with alternative multiple cuts for a multi-product closed-loop supply chain network design model", *Naval Research Logistics*, 54(8), 890-907, 2007.

[5] Fleischmann, M., Bloemhof-Ruwaard, J., Dekker, R., Van der Laan, Van Nunen, J. and Van Wassenhove, L., "Quantitative models for reverse logistics: A review", *European Journal of Operational Research*, 103, 1-17, 1997.

[6] Goetschalckx, M., Vidal, C. and Dogan, K., "Modeling and design of global logistics systems: A review of integrated strategic and tactical models and design algorithms", *European Journal of Operational Research*, 143(1), 1-18, 2002b.

[7] Papageorgiou, L., "Supply chain optimisation for the process industries: Advances and opportunities", *Computers & Chemical Engineering*, 33(12), 1931-1938, 2009.

[8] Melo, M., Nickel, S. and Saldanha da Gama, F., "Facility Location and Supply Chain Management - A review", *European Journal of Operation Research*, 196, 401-412, 2009.

[9] Salema, M., Barbosa-Póvoa, A. and Novais, A., "Simultaneous design and planning of supply chains with reverse flows: A generic modelling formulation", *European Journal of Operational Research*, 203, 336-349, 2010.

[10] Guillén-Gosálbez, G. and Grossmann, I., "A global optimization strategy for the environmentally conscious design of chemical supply chains under uncertainty in the damage assessment model", *Computers & Chemical Engineering*, 34, 42-58, 2010.

Reverse Logistics Network Design for Household Plastic Waste

Xiaoyun Bing * Jacqueline Bloemhof Jack van der Vorst

* Wageningen University and Research Center, the Netherlands
Logistics, Decision and Information Sciences
P.O. Box 8130, 6700 EW Wageningen

xiaoyun.bing@wur.nl

ABSTRACT

This paper applies MILP methods to improve the network design of reverse logistics for household plastic waste based on the case of Netherlands. The purpose is to provide decision support for various stakeholders in choosing the most suitable recycling collection methods with an optimized network design that both balances their interests and improves the recycling efficiency. Separation method determines whether the quality and quantity of the plastics material is high enough to be economically efficient and environmentally effective. Currently, source separation (separation at households) is dominating as suggested by legislation. However, since the overall collection rate is not satisfying, municipalities are trying different ways to deal with plastic waste. There is a need to adopt the system according to the characteristics of the municipalities. This research follows the approach of scenario study. We start with the simulation of the current situation followed by investigating the impacts of various changes in the collection system. For each scenario, we suggest improvement in the network by repositioning the locations for separation, sorting and reprocessing sites.

Keywords: Reverse logistics, Network design, Mixed integer linear programming, Plastic recycling

1. INTRODUCTION

1.1. Purpose

This paper applies operations research methods to improve the network design of reverse logistics for household plastic waste based on the case of the Netherlands. The purpose is to provide decision support for various stakeholders in choosing the most suitable recycling collection methods with an optimized network design that both balances their interests and improves the recycling efficiency.

1.2. Problem and Research Question

Due to a higher volume to weight ratio in comparison to other recyclables, plastics have a larger number of kilometers traveled per tonne, meaning more emissions and less efficiency in transportation (Craighill and Powell, 1996)[1]. That is why only a few recycling and collection facilities exist compared to other types of recyclable packaging waste such as glass and paper (Waste online, 2010). However, the rising oil price and the cost reduction by using recycled plastics instead of virgin polymer-based plastics lead to a high demand for plastic recycling. As a result, there is a need to build an efficient network that improves the recycling system. Figure 1 illustrates the current flow of plastics recycling. Plastics recycling network in the Netherlands is characterized by various collection, separation and treatment systems. The first step of the processing system, separating plastics from other waste, can occur at households (source separation) or in separation centers (post-separation), making a difference in infrastructure, collection frequency, vehicle types, etc. Decisions on the choice of the system depend on issues like the type of municipality described by population density, geographical location, householders' behavior as well as the availability of resources. The separation method, together with the corresponding collection system and frequency, determine whether the quality and quantity of the plastics material is high enough to be economically efficient and environmentally effective.

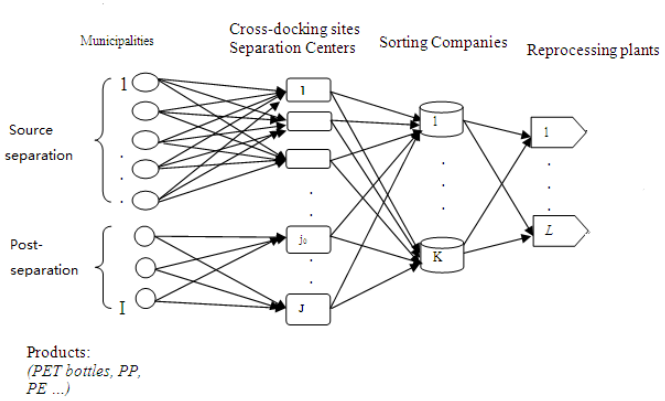


Figure 1: *Flow chart of reverse network for plastics waste*

A special characteristic of the Dutch network is that although the land area is not large, there are 441 different municipalities varying a lot in population density and housing types (Central Bureau of Statistics in Netherlands, 2009). For many municipalities, there is a mixture of different household types (apartments, houses, farms) within the municipality which results in not only population density difference but also diversity in plastic waste components. Not all the processing work is done inside the country, for instance, sorting facilities in Germany and reprocessing companies from all over the Europe are involved in the network as well. Currently, source separation is dominating. More than 90% of the municipalities are doing source separation as suggested by legislation. Since the overall collection rate is not satisfying, municipalities are trying different ways to deal with plastic waste. Households in urban municipalities have limited space at home for doing source separation. Therefore there is a need to adopt the system according to the characteristics of the municipalities.

The research question in this paper is

- What is the best reverse network design for plastic recycling in Netherlands that is both efficient and sustainable?

This includes the decision support for making the choice of source separation or post-separation in order to balance the interests and achieve the lowest overall transport cost from the point of collec-

tion to the final processing facility.

On contrary to normal distribution networks in which products assemble at the source or sometimes during the flow, plastic waste disassembles along distribution from the sources to the end processors. Many plastic fractions are collected together at the source mixed with dirt and moisture and even other municipal solid waste, depending on the collection method. Along the flow, separation and sorting are going on. In the end, different plastic fractions will be distributed to separated processors. The useless part out of each step of separation/sorting will be disposed through other channels, therefore quantity of plastics also reduces during distribution. Additionally, PET bottles are a special category of plastic waste. It has a special channel of recycling in the network other than the normal plastic waste. The network design should fit these special features.

1.3. Literature Review

The origin of research on reverse logistics can be dated back to 1995. Ever after that, there has been growing interest in research of this filed. Compared with forward logistics, Fleischmann et.al (1997) [2] identified a specialty of reverse distribution network, that is, it is not necessarily a symmetric picture of forward distribution. Most of them has a "many to few" network structure. In the case of plastic recycling network, the many municipalities as suppliers and the few sorting plants and reprocessors as customers form such a "many to few" structure. Rubio et.al(2006)[3] reviewed the characteristics of the research on reverse logistics during the period of 1995-2005 and pointed out that the majority of research focuses on the study of tactical and operational aspects like production planning and inventory management. Research on reverse logistics could be directed to the analysis on strategic aspects. Another trend in this research field is that environmental issues are becoming an important parameter in logistics network design. Srivastava (2007)[4] reviewed green supply chain management taking a reverse logistics angle. The new concept of green supply chain leads to a shift from minimizing cost to a balance between cost and environmental impact. In line with these research directions, this paper deals with the interaction between available technology(separation and sorting) and possible collection methods. Through network planning and scenario study, the purpose is to provide decision support for stakeholders in choosing the most suitable and sustainable recycling strategy.

1.4. Methodology

Mixed integer linear programming (MILP) models are used in this network design. To achieve the objective, the research follows the approach of scenario study by forming a list of scenarios first, then comparing the network modeling result of these scenarios. The modeling is conducted by using a graphical optimization tool IBM Logic Net Plus. Unlike the usual forward supply chain network model, we have all the plastic fractions together with dirt and moisture as various "products" in the model. Municipalities are the supplier of these "products". The distinctive "many to few" structure is built in and the special feature of product disassembling as waste disposal during the flow is simulated. The objective of the MILP model is to minimize the overall transportation cost of the four levels of Figure 1. In each scenario, different network layout, assumptions on the choice of collection channels and the characteristics of municipalities define the quantity of the products, their flow and the availability of facilities in the network which are constraints for the model. In this scenario study, we start with the simulation of the current situation based on source separation with separate PET bottle collection. Then we investigate the impacts of

- shifting to 100% post-separation;
- adopting PET collection system from other countries;
- choosing a collection method according to the population density of the municipality.

Modeling results are compared and discussed to answer the above mentioned research question. For each scenario, we suggest improvements in the network by repositioning the locations for sorting, separation and reprocessing sites.

1.5. Data and Data Sources

Main data used for building up the models and the data sources are as follows:

- Municipalities (population, quantity of plastic waste, location)
  Statistics can be collected through the annual reports of CBS (Central Bureau of Statistics in Netherlands)
- Processing facilities (function, location, capacity)
  Nedvang (Dutch packaging waste recycling association), has relevant information on the processing facilities in Netherlands. Another project partner Aachen University has more expertise on the system of Germany facilities.
- Plastic waste (components, quality, separation technology)
  Data is provided by one of the research anchors of the Kenniscentrum Nascheiding (KCN), an Expertise Center located at Wageningen University that investigates the technological and economical feasibility, as well as the environmental impact, of new technologies for the treatment of plastics (packaging waste) found in household waste.

2. FINDINGS

The modeling results are expected to give the following findings

- The logistics bottleneck of applying source separation/post-separation in a country
- The impact of choosing separation systems that fit the characteristics of municipalities
- For a chosen combination of source separation and post-separation, the location allocation of the separation and processing facilities
- The influence on transportation efficiency of separately collecting the PET bottles.

Current preliminary results show that in general, having a separate channel for PET bottles collection reduces the over all transportation cost. Among all the tested scenarios, doing source separation in rural ones and post-separation in urban areas while keeping the separate PET bottle refund system performs best in saving transportation cost.

3. RELEVANCE / CONTRIBUTION

This paper focuses on a niche scope of reverse logistics by applying operations research modeling techniques in reverse logistics on plastic recycling problem in specific with multiple objectives, multiple layers and multiple stakeholders. Traditional network design models are modified and extended to fit in the specific network features and to solve the applied problem.

4. ACKNOWLEDGEMENTS

This research is supported by Nedvang (Dutch packaging waste recycling association) in cooperation with KCN (an Expertise center in Wageningen University of plastic recycling). We would like to thank them for all the support they give and also the great help in collecting data for the research.

Many thanks to the reviewers!

5. REFERENCES

[1] A. Craighill and J. Powell, "Lifecycle assessment and economic evaluation of recycling: A case study," *RESOURCES CONSERVATION AND RECYCLING*, vol. 17, no. 2, pp. 75–96, AUG 1996.

[2] M. Fleischmann, J. Bloemhof-Ruwaard, R. Dekker, E. van der Laan, J. van Nunen, and L. VanWassenhove, "Quantitative models for reverse logistics: A review," *EUROPEAN JOURNAL OF OPERATIONAL RESEARCH*, vol. 103, no. 1, pp. 1–17, NOV 16 1997.

[3] S. Rubio, A. Chamorro, and F. J. Miranda, "Characteristics of the research on reverse logistics (1995-2005)," *INTERNATIONAL JOURNAL OF PRODUCTION RESEARCH*, vol. 46, no. 4, pp. 1099–1120, 2008.

[4] S. K. Srivastava, "Green supply-chain management: A state-of-the-art literature review," *INTERNATIONAL JOURNAL OF MANAGEMENT REVIEWS*, vol. 9, no. 1, pp. 53–80, MAR 2007.

Reverse Cross Docking

Juan Pablo Soto * Rosa Colomé Perales † Marcus Thiell *

* UniAandes School of Management
Bogotá, Colombia
{jps, mthiell}@adm.uniandes.edu.co

† ESCI, Universitat Pompeu Fabra
Pg. Pujades 1, Barcelona
osa.colome@esci.es

ABSTRACT

Nowadays companies are facing an important challenge in their distribution, as frequent deliveries and small order sizes are the common rule today. For this type of distribution, cross-docking is a logistics activity that generates several advantages like reduction in lead times and manipulation costs. In addition, Reverse Logistics (RL) has achieved more importance in recent years within the business world. In particular companies with fashion products are introducing RL activities to recover and, in most cases, resale the products through the same or through different channels of distribution like outlets, secondary markets, or internet, with the purpose to recapture value. Despite of the success of cross-docking in distribution, the concept has not been applied for the reverse flow so far. In this paper we propose a linear programming model that allows the use of cross-docking in a Reverse Logistics context, where returned products can be redirected to the outlets chain without storage.

Keywords: Reverse Logistics, Cross-docking

1. INTRODUCTION

With improvements in logistics operations, nowadays companies are facing a change in the way they are doing their distribution. Today it is very common to find frequent shipments with small order sizes. In such situations, cross-docking is a logistical activity that generates several advantages [1]in areas like inventory management, order picking, and transportation [2]. Products arriving to the cross dock are unloaded from inbound trailers, possibly reconsolidated with other products arriving from different destinations, and loaded into outbound trailers within less than 24 hours [3]. In practice, cross-docking is possible because the suppliers receive high quality information and organize the orders by final destination. At the moment goods arrive at the distribution center, it is then just necessary to translate them into a predefined position for the final client.

Reverse Logistics (RL), the second theoretical concept we refer to in this paper, has received more importance in recent years around the world. The implementation of environmental laws, an increasing customer awareness related to environmental issues, the reduction of the product life cycles and the creation of new business models based on returned products, are some of the drivers for the introduction of RL operations in supply chains [4]. As a consequence, many of the theories and practices developed in direct logistics have been adjusted properply to an environment that includes the returned products at the end of its life cycle [5]. As reverse flows of products are characterzed by high uncertainty in quantity and quality, those flows are like a “black box” until they arrive to the distribution center. Therefore the application of cross docking in the reverse logistics context is not prevalent so far. But in the special case of fashion retail companies, who have their own network of outlet stores, it seems to be possible to introduce cross docking in the reverse flow since product assortments can be created based on the available products which were unsold during the sales period. Although an "ideal" product assortment is planned for each outlet store, this is not unchangeable and allows companies to modify it with a certain degree of flexibility.

In this paper we propose a model designed for retail chains that have their own outlet stores to commercialize the returned products being called the “Reverse Cross-docking Model”. The next section treats a literature review related to this topic, followed by the explanation of the model in section 3, the mathematical formulation in section 4, and a conclusion as well as a research outlook in section 5.

2. LITERATURE REVIEW

Reverse Logistics (RL) has been defined as the process of planning, implementing and monitoring the effective and efficient flow of raw materials, in-process inventory, finished goods and all related information from the point of consumption to the point of origin with the purpose of recapturing value or arrange it properly [6]. In recent years, RL has gained more importance both in business and research. This growth is particularly significant in sectors with high rates of returns and high standards of customer service. Additionally, the growing concern about climate change and environmental impact of business activities, joint with the legislation changes towards a cleaner manufacturing environment, has enhanced companies to introduce different practices of product recovery. [1]

Cross-docking systems have been implemented since many years in the business world, being defined as a warehousing strategy that involves movement of material directly from the receiving dock to the shipping dock with a minimum time in between [7]. In other words, it is a practice of moving goods through distribution centers without storing it, increasingly used by enterprises for comprehensive management of its operations in the supply chain. One example for its application is Wal-Mart, who introduces the cross-docking system early in the ’90s. A big part of the competitive advantage and growth of Wal-Mart in the U.S., was due to the use of this strategy in its operations [8].

To the best of the authors knowledge no model was presented to deal with the implementation of cross-docking in RL so far. Due to the RL characteristics, mainly related to variety and uncertainty

[1] (see www.greenbiz.com)

of returned products, it can be assumed that it is difficult to apply cross-docking in the RL field. This paper contributes to the literature of RL with the creation of the first model dealing with the implementation of cross-docking in a RL context.

3. PROBLEM DESCRIPTION

Despite the general benefits derived from cross-docking in direct logistics, this use in RL is difficult due to the characteristics of the Reverse Supply Chain: outlet stores receive the products without having ordered them explicitely as their product offering depends on the return flow of the stores and the destination of the goods is not defined in advance, because the quality and the quantity are unknown in most cases. Sometimes it is not even know when the product arrives; normally, there are no orders from the "clients" of the chain, i.e. secondary markets where the commodity is sold. Even in the case that a company has control on the secondary markets, it is difficult to achieve a fit between these orders and the products that are returned from the main channel of sale.

In a situation where companies control both, the direct store chain and the outlet store chain, there is a way of benefiting from cross-docking practices in the returns flow. Fashion companies fit well with the described profile, since their product is affected by seasonal demands (autumn-winter and spring-summer), and its merchandise must rotate during each season. At the end of the season all merchandise that has not been sold is taken out of the shops and sent back to the distribution center. From this moment the products are no longer managed by the main distribution channel as a result of the intended variation of product assortments from season to season.

Traditionally, a returned box from a store follows several processes in order to be prepared for the distribution to the outlet stores. Main processes are: opening the boxes, sorting the products, organizing them by reference, size, color and other predefined features, storage them in the warehouse, and finally use them to fulfill the orders generated by the outlets. While this system works relatively well, it is quite costly, mainly due to the storage costs and time required to have a product ready to be sent to the outlet stores.

Usually, fashion companies create a desirable product assortment list for the outlet stores combining a product offer that is supposed to satisfy the final customer. To create this product assortment list, companies consider past sales and customers profiles for each outlet.

Cross docking can help to minimize operational costs, mainly of storage, retrieval and picking. In order to operate with cross-docking in the reverse channel it is necessary to have the critical information available (i.e. desired product assortment from each outlet and product sent per box from the direct stores). With this information at hand, it is possible to create a matching lists in which the company can see if there is a box that coincides with the desired products of an outlet. As it is difficult to find a box that fits 100% with the order of an outlet store, we assume that the company has some flexibility to change the desirable products of a given outlet store. But this flexibility is limited as the outlet stores do not intend to have an excess of unwanted products. In other words, this leads to the question: How many unwanted products is the company able to send to the outlet stores?

If the company sends many undesired products to the outlets, those products will be returned with a certain probability after passing a period of time in the outlet stores, causing corresponding costs. If the company does not have a certain degree of flexibility, just a few boxes can be sent through the reverse cross docking operation making it finally inefficient. To deal with this situation, we created an optimization model where an optimal percentage of matching is established. This Matching Percentage is computed for every box-outlet pair. Afterwards, the optimization model searches for the Global Percentage of Acceptance (GPA) that a box needs to achieve in order to be sent to an outlet store. Figure (1) shows how the system works.

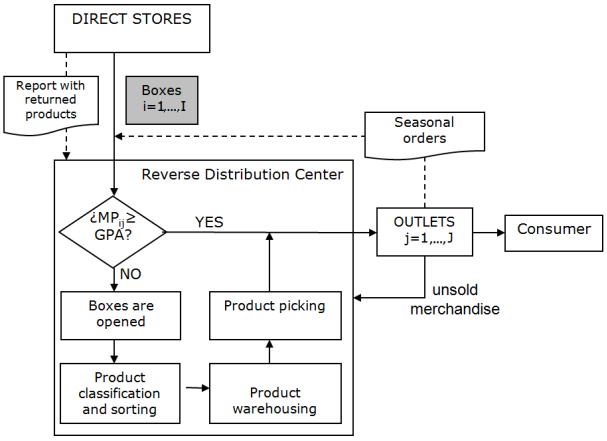


Figure 1: Sistem Operation

If a given box has a Matching Percentage which is above the GPA then the box is marked as candidate to be assigned to the corresponding outlet. The optimization model also considers the maximization of the sum of all the matching percentages, assuring that boxes will be sent to the outlets which fit better with its content. If the GPA is too low, then the estimated costs of taking back products from the outlets at the end of the season increase. If the GPA is too high, then only a few boxes are sent through reverse cross-docking, and traditional storage and picking must be performed which consequently increases the costs.

The objective function of the model minimizes the total costs and maximizes the sum of all matching percentages (1). Constraints of the model are: computation of products from a box (C_{ai}) which are in excess in comparison to the outlet order (O_{ai})or number of products of a given reference which are not enough to fulfill the outlet requirements (2); computation of the Matching Percentage (MP_{ij}) (3); Comparison between the Matching Percentage of a given pair box-outlet and the Global Percentage of Acceptance (GPA) (4);a box i is potentially assigned to an outlet j (PA_{ij}) if its matching percentage is greater than GPA (5); a box can be assigned to the traditional method (BT_i) only if it is not assigned for cross docking to any of the outlets (6) ; as a percentage, GPA and MP_{ij} must be less than or equal to 1 (7) and (8);computation of the cost of returned products at the end of the season. This takes into account an historical probability of returning a product a from the outlet j based on historical data (PR_{aj}). This is a parameter of the model. The estimation of products returned is the first integer number greater than the value obtained ($\hat{RP}_{aj}$) (9); binary, non-negativity and integer constraints (10),(11) and(12). The model was solved using the Lingo Software and results show that the costs of the total system can be reduced when the reverse cross docking practices are implemented.

$$Minz \sum_{a=1}^{A} C_{ai} \cdot BT_i \cdot UPC_a + \sum_{j=1}^{J} \hat{RP}_{aj} \cdot RLC_a + (M - \sum_{i=1}^{I} \sum_{j=1}^{J} MP_{ij} \cdot PA_{ij}) \quad (1)$$

$$\sum_{a=1}^{A} (C_{ai} - O_{ai}) \cdot Y_{ai} = SP_{ij} - LP_{ij} \forall i, j \quad (2)$$

$$MP_{ij} = 1 - \left(\frac{SP_{ij}}{\sum_{a=1}^{A} C_{ai}} \right) \forall i, j \quad (3)$$

$$MP_{ij} - GPA = CD_{ij} - CT_{ij} \forall i, j \quad (4)$$

$$PA_{ij} \geq MP_{ij} - GPA \forall i, j \quad (5)$$

$$BT_i \leq \sum_{j=1}^{J} PA_{ij} \forall i \quad (6)$$

$$GPA \leq 1 \quad (7)$$

$$MP_{ij} \leq 1 \quad (8)$$

$$\sum_{i=1}^{I} (PA_{ij} \cdot LP_{ij} \cdot C_{ai}) \cdot PR_{aj} \leq R\hat{P}_{aj} \forall a, j \quad (9)$$

$$BT_i, PA_{ij} \in \{0,1\} \quad (10)$$

$$SP_{ij}, LP_{ij}, MP_{ij}, CD_{ij}, CT_{ij}, GPA \geq 0 \quad (11)$$

$$\hat{RP}_{ij} \in \{integer\} \quad (12)$$

Where:

$a = \{1..A\}$ Set of articles

$i = \{1..I\}$ Set of boxes

$j = \{1..J\}$ Set of outlet stores

M =a big number greater than the maximum possible value of the sum of all matching percentages joint together

UPC_a =Processing cost per unit a in the distribution center.

MP_{ij} = Matching Percentage: Is the percentage in which box i, fulfill the order from outlet j.

SP_{ij} = Number of products in box i which are sent in excess to the outlet j.

LP_{ij} = Number of products in box i which are lacking to fulfill the order of outlet j.

CD_{ij} =Positive difference between the percentage of matching and the GPA.

CT_{ij} =Negative difference between the percentage of matching and the GPA.

4. CONCLUSIONS AND FUTURE RESEARCH

Cross-docking is a common strategy that have shown its benefits in reducing operational costs and time in business. In this paper we presented a model in which this practice can be used in a Reverse Logistics context. This model was applied to an environment where companies own their direct and outlet store channels. Prerequisit for the model application are two characteristics: availability of information about the desirable assortment of products from the outlet stores, and information about the products returned from the direct stores at box level. The model proposed established the optimum global percentage of acceptance in order to maximize the benefits for the company. The results obtained furthermore show that companies can obtain important reductions in operational costs from the application of this reverse cross docking model.

For future research, the consideration of a dynamic allocation of inventory could provide additional insights and probably lead to an improvement of the initial solution presented in this paper.

5. REFERENCES

[1] H. Yan and S.-L. Tang, "Pre-distribution and pos-distribution cross-docking operations," *Transportation Research Part E*, vol. 45, pp. 843–859, May 2009.

[2] R. Larbi, G. Alpan, P. Baptiste, and B. Penz, "Scheduling cross docking operations under full, partial and no information on inbound arrivals," *Computers and Operations Research*, vol. 38, pp. 889–900, 2011.

[3] G. Alpan, R. Larbi, and B. Penz, "A bounded dynamic programming approach to schedule operations in a cross docking platform," *Computers and Industrial Engineering*, vol. 60, pp. 385–396, April 2011.

[4] A. Diaz, M. J. Alvarez, and P. Gonzalez, *Logística Inversa y Medio Ambiente*. Madrid, Spain: Mc Graw-Hill Interamericana de Espana S.A., 2004.

[5] M. Fleischmann, J. M. Bloemhof-Ruwaard, R. Dekker, E. V. D. Laan, J. A. van Nunen, and L. N. V. Wassenhove, "Quantitative models for reverse logistics," *European Journal of operational research*, vol. 103, pp. 1–17, June 1997.

[6] D. Rogers and R. Tibben-Lembke, *Going Backwards: Reverse Logistics Trends and Practices*. Reverse Logistics Executive Council, Eds., 1998.

[7] U. M. Apte and S. Viswanathan, "Effective cross docking for improving distribution efficiencies," *International Journal of Logistics:Research and Applications*, vol. 3, no. 3, pp. 291–302, 2000.

[8] G. S. Jr., P. Evans, and L. E. Shulman, *Delivering Results: A new mandate for human resource professionals*. Boston, USA: HBSP, 1992, ch. Competing on Capabilities: The New Rules of Corporate Strategy, pp. 1–355.

Comparing Roster Patterns Within a Single Depot Vehicle-Crew-Roser Problem

Marta Mesquita * Margarida Moz † Ana Paias ‡ Margarida Pato §

* ISA-UTL, CIO
Tapada da Ajuda, 1349-017 Lisboa, Portugal
marta@math.isa.utl.pt

† § ISEG-UTL, CIO
Rua do Quelhas 6, 1200-781 Lisboa, Portugal
†mmoz@iseg.utl.pt §mpato@iseg.utl.pt

‡ DEIO-FCUL, CIO
Bloco C6, Piso 4, Cidade Universitaria, 1749-016, Lisboa, Portugal
ampaias@fc.ul.pt

ABSTRACT

The integrated vehicle-crew-roster problem aims to simultaneously determine minimum cost vehicle and daily crew schedules that cover all timetabled trips and a minimum cost roster covering all daily crew duties according to a pre-defined days-off pattern. This problem is solved by a heuristic approach based on Benders decomposition that iterates between the solution of an integrated vehicle-crew scheduling problem and the solution of a rostering problem. Computational experience with data from two bus companies in Portugal is used to compare two rostering patterns within vehicle-crew-roster solutions.

Keywords: Rostering, vehicle-scheduling, crew-scheduling, Benders decomposition

1. INTRODUCTION

The integrated vehicle-crew-roster problem aims to simultaneously assign drivers of a company to vehicles and vehicles to a set of predefined timetabled trips that cover passenger transport demand in a specific area, during a planning horizon. Due to the complexity of the corresponding combinatorial optimization problem, it is usually tackled on a sequential basis beginning with vehicle scheduling, followed by crew scheduling and, lastly, driver rostering. Vehicle scheduling produces daily schedules for the vehicles that perform all trips. The crew scheduling defines daily crew duties that cover the respective vehicle schedules. Finally, for the planning horizon, crew duties are assigned to the drivers of the company leading to a roster that must comply with labor and institutional norms. However, there is a high dependency among these three problems and despite its computational burden, some work has been reported on the integration of all or some of these problems, expecting to outperform the corresponding sequential approaches. Among other authors, [1], [2] and [3] have developed efficient algorithms for the integrated vehicle-crew problem. Crew-rostering integration has been devised by [4], [5] and by [6] albeit within other transport contexts (railway, airline and airport staff).

Following the idea that staff costs constitute more than 50% of operating costs, one wants to compare two different roster patterns in what concerns the resulting integrated vehicle-crew-roster solutions. A heuristic approach for an integrated vehicle-crew-roster problem with days-off pattern (VCRPat) is presented. The approach combines column generation and branch-and-bound techniques within a Benders decomposition and iterates between the solution of an integrated vehicle-crew scheduling problem and the solution of a rostering problem. A preliminary insight on this decomposition approach was already presented by the authors in [7]. Benders decomposition methods have been proposed by [8], [9], [10] and [11], although for airline operations problems. This paper is organized as follows: Section 2 introduces the VCRPat along with the two days-off patterns; Section 3 describes the mathematical model; Section 4 presents the decomposition algorithm and Section 5 gives some conclusions from preliminary tests.

2. PROBLEM DEFINITION

During a planning horizon H, partitioned into 49 days, a set M of drivers must be assigned to a fleet of vehicles housed at a depot d in order to perform a set of timetabled trips (trips for short). The location and the number of vehicles available at the depot as well as the set of trips to be performed on each day h are known. For each trip the starting and ending times and locations are given. Trips i and j are compatible if the same vehicle can perform both trips in sequence. Between compatible trips i and j a deadhead trip may occur where the vehicle runs without passengers. There are three types of deadhead trips: those between the end location of a trip and the start location of a compatible trip, those from the depot to the start location of a trip (pull-out) and those from an end location of a trip to the depot (pull-in). The set of timetabled trips and deadhead trips performed by a vehicle on day $h \in H$ defines a vehicle schedule. Each vehicle schedule starts and ends at the depot and is subdivided into points, called relief points, where a change of driver may occur. Two consecutive relief points define a task, that is, the smallest amount of work to be assigned to the same vehicle and crew.

A crew duty is a daily combination of tasks that respects labor law, union contracts and internal rules of the company. These rules depend on the particular situation under study and usually constrain the maximum and minimum spread (time elapsed between the beginning and end of a crew duty), the maximum working time without a break, the break duration, etc. Crew duties are assigned to the drivers to form their work schedules - the roster. This is usually done on a cyclic basis so as all workers have the same type of work and rest periods. In this paper, we deal with a group of drivers with more flexibility on the rosters. These drivers work according to a pre-defined cyclic days-off pattern where all drivers share the same type of rest periods, but not necessarily the same crew duties. For a time horizon of 7 weeks (49 days), this days-off pattern, PatI, includes 4 rest periods of 2 consecutive days-off and 2 rest periods

of 3 consecutive days-off. These 2 rest periods contain a Saturday and occur in sequence with 5 workdays in between. The remaining work periods have 6 consecutive workdays.

Table 1 displays PatI through a $0-1$ matrix, where 0 stands for day-off and 1 for workday. Each row of the matrix corresponds to a weekday. Seven consecutive columns correspond to the 7 weeks of the time horizon, being the last day in column i $(i = 1,...,6)$ followed by the first day in column $i+1$ and the last day in column 7 followed by the first day of column 1. Since a 49 days schedule may begin in row 1 of any column, this days-off pattern leads to a set S of 7 schedules $s_i, i = 1,...,7$. That is, for example, a driver assigned to schedule $s_4 \in S$ works according to columns 4, 5, 6, 7, 1, 2, and 3, during weeks, 1, 2, 3, 4, 5, 6, and 7, respectively.

	1	2	3	4	5	6	7
Mon	0	1	1	1	1	1	0
Tue	0	0	1	1	1	1	1
Wed	1	0	0	1	1	1	1
Thu	1	1	0	0	1	1	1
Fri	1	1	1	0	0	1	1
Sat	1	1	1	1	0	0	1
Sun	1	1	1	1	0	0	1

Table 1: Cyclic days-off pattern (PatI).

Usually, in public transit companies the workforce demand is constant from Monday to Friday but it decreases during the weekend. But, for the cyclic days-off pattern displayed in Table 1, in each weekday, there are always 2 schedules covering drivers days-off and 5 schedules covering drivers workdays. In order to minimize the number of drivers assigned to work we propose an additional schedule, s_8, with 7 rest periods of 2 consecutive days-off that always occur on Saturday and Sunday. During the planning horizon H, drivers assigned to s_8 work Monday through Friday and rest Saturday and Sunday. The 7 schedules in PatI plus schedule s_8 define the set S of schedules in pattern PatII, displayed in Table 2.

	1	2	3	4	5	6	7	8
Mon	0	1	1	1	1	1	0	1
Tue	0	0	1	1	1	1	1	1
Wed	1	0	0	1	1	1	1	1
Thu	1	1	0	0	1	1	1	1
Fri	1	1	1	0	0	1	1	1
Sat	1	1	1	1	0	0	1	0
Sun	1	1	1	1	0	0	1	0

Table 2: Cyclic/non cyclic days-off pattern - PatII.

PatII tends to counterbalance the lower demand during weekend. From Monday to Friday, the covering of drivers days-off and workdays is equal in PatI and PatII. However, on Saturday and Sunday PatII has 3 schedules covering drivers days-off.

A roster is an assignment of drivers to schedules in S covering all the crew duties defined for the planning horizon while satisfying labor and internal rules of the company. Any number of drivers may be assigned to each schedule.

Note that, both PatI and PatII satisfies a minimum number of days-off per week (1 day), the minimum number of consecutive days-off (2 days), a minimum number of Sundays/weekends off in the planning period (2 days) and a maximum of consecutive workdays (6 days). Besides these constraints, drivers must rest a minimum number of hours between two consecutive crew duties and, consequently, a crew duty starting in the morning cannot be assigned to a driver that had worked on a crew duty starting in the evening of the previous day. To avoid infeasible sequences of crew duties, for each day h, the set of crew duties, L^h, is partitioned into early crew duties, L^h_E, starting before 3:30 p.m. and late crew duties, L^h_A, starting after 3:30 p.m.

Moreover, for a roster to be well accepted in the company it should attain balanced workload. To balance workload, the set L^h is also partitioned into short duties, L^h_T, which have a maximum spread of 5 hours (without a break), normal duties, L^h_N, with spread $\in [5,9]$ hours, and long duties, L^h_O, with spread $\in [9,10.75]$ hours (with overtime).

The *VCRPat* aims to simultaneously determine a minimum cost set of vehicle schedules that daily covers all timetabled trips, a minimum cost set of crew duties that daily covers all vehicle schedules and a minimum cost balanced roster for the time horizon.

3. MATHEMATICAL MODEL

For each day h, we define the vehicle scheduling network $G^h = (V^h, A^h)$. The node set $V^h = N^h \cup \{d_s, d_t\}$ includes N^h corresponding to the timetabled trips to be performed on day h and $\{d_s, d_t\}$ corresponding to the depot d. The arc set $A^h = I^h \cup (d_s \times N^h) \cup (N^h \times d_t)$ contains I^h, the set of arcs representing the pairs of compatible timetabled trips, and the sets of arcs related with pull-out and pull-in trips. Each path on graph G^h, starting in d_s and ending in d_t, defines a vehicle schedule for a specific vehicle on day h. A set of paths from d_s to d_t disjoint on N^h, covering all nodes from N^h, defines a vehicle scheduling for day h. Decision variables z^h_{ij} indicate whether a vehicle performs trips i and j in sequence on day h, or not. In particular, $z^h_{d_s j}$ and $z^h_{id_t}$ represent, respectively, a pull-out from the depot to trip j and a pull-in from i to the depot. Vehicle costs c_{ij}, related with fuel consumption and/or vehicle maintenance, are associated to the corresponding arcs in G^h.

Daily vehicle schedules must be covered with daily crew duties. Since the depot as well as the end location of timetabled trips are potential relief points, we assume that task (i, j) corresponds either to the deadhead from trip i to trip j followed by trip j or to the deadhead from trip i to trip j followed by trip j and a pull-in. Let $L^h_{ij} \subseteq L^h$ be the set of crew duties covering task (i, j), on day h. Let variables w^h_ℓ indicate whether crew duty ℓ is selected on day h, or not. A cost e_ℓ is assigned to each crew duty ℓ. Cost e_ℓ usually includes a fixed cost (for example, a driver's salary) and operational costs related to overtime, evening periods, etc.

Each anonymous crew duty in the solution has to be assigned to a specific driver which works according to one of the pre-defined days-off schedules. Let $x^m_s = 1$ if driver m is assigned to schedule s, or 0 otherwise. Let $y^{mh}_\ell = 1$ if driver m performs crew duty ℓ on day h, or 0 otherwise.

The objective function includes different measures. Management often wishes to know the minimum workforce required to operate the fleet of vehicles, so as to transfer drivers to other departments of the company or to replace those absent. Such policy results in minimizing crew duty costs e_ℓ associated to variables $w^h_\ell = 1$ and costs r^m associated to $x^m_s = 1$. To balance workload, penalties λ_T and λ_O are associated with variables η_T and η_O that represent, respectively, the maximum number of short and long crew duties assigned to a driver during H. The integer linear programming formulation follows:

$$min \sum_{h\in H}\Big(\sum_{(i,j)\in A^h} c_{ij} z^h_{ij} + \sum_{\ell\in L^h} e_\ell w^h_\ell\Big) + \sum_{m\in M}\sum_{s\in S} r^m x^m_s + \lambda_T \eta_T + \lambda_O \eta_O \quad (1)$$

$$\sum_{i:(i,j)\in A^h} z^h_{ij} = 1, \quad j \in N^h, h \in H \quad (2)$$

$$\sum_{j:(i,j)\in A^h} z^h_{ij} = 1, \quad i \in N^h, h \in H \quad (3)$$

$$\sum_{i \in N^h} z^h_{d_s i} \leq v, \quad h \in H \tag{4}$$

$$\sum_{\ell \in L^h_{ij}} w^h_\ell - z^h_{ij} \geq 0, \quad (i,j) \in A^h, h \in H \tag{5}$$

$$\sum_{m \in M} y^{mh}_\ell - w^h_\ell = 0, \quad \ell \in L^h, h \in H \tag{6}$$

$$\sum_{s \in S} x^m_s \leq 1, \quad m \in M \tag{7}$$

$$\sum_{\ell \in L^h} y^{mh}_\ell - \sum_{s \in S} a^h_s x^m_s \leq 0, \quad m \in M, h \in H \tag{8}$$

$$\sum_{\ell \in L^h_A} y^{mh}_\ell + \sum_{\ell \in L^{(h-1)}_E} y^{m(h-1)}_\ell \leq 1, \quad m \in M, h \in H - \{1\}, \tag{9}$$

$$\sum_{\ell \in L^h_E} y^{mh}_\ell + \sum_{\ell \in L^{(h-1)}_A} y^{m(h-1)}_\ell \leq 1, \quad m \in M, h \in H - \{1\}, \tag{10}$$

$$\sum_{h \in H} \sum_{\ell \in L^h_t} y^{mh}_\ell - \eta_t \leq 0, \quad m \in M, t \in \{T, O\} \tag{11}$$

$$z^h_{ij} \in \{0,1\}, \quad (i,j) \in A^h, h \in H \tag{12}$$

$$w^h_\ell \in \{0,1\}, \quad \ell \in L^h, h \in H \tag{13}$$

$$y^{mh}_\ell \in \{0,1\}, \quad \ell \in L^h, m \in M, h \in H \tag{14}$$

$$x^m_s \in \{0,1\}, \quad s \in S, m \in M \tag{15}$$

$$\eta_T, \eta_O \geq 0 \text{ and integer.} \tag{16}$$

Constraints (2), (3) and (4) describe the scheduling of vehicles ensuring that each timetabled trip is performed exactly once by a vehicle. These constraints ensure that, for each day $h \in H$, graph G^h is partitioned into a set of disjoint paths, vehicle schedules, covering all vertex in N^h. Constraints (4), where v is the number of vehicles available at the depot, are related with the depot capacity. Constraints (5) link vehicle and crew duty variables ensuring that each task in a vehicle schedule is covered by one daily crew duty. Equalities (6) impose that each crew duty, in a solution, must be assigned to one driver. Constraints (7) state that each driver is assigned to one of the seven days-off schedules or is available for other service in the company. Constraints (8), where parameter $a^h_s = 1$ if h is a workday on schedule s, impose coherence between the assignment of a crew duty to a driver and the schedule assigned to this driver. Inequalities (9) forbid the sequence late/early duties to ensure that drivers rest a given minimum number of hours between consecutive duties. Furthermore, (9) and (10) impose a day-off period between changes of duty types. Inequalities (11) determine the maximum number of short/long duties per driver. Note that, these constraints along with the two last terms of the objective function ensure the integrality of variables η_T and η_O.

To deal with the VCRPat, a decomposition approach is suggested by three combinatorial structures included in this mathematical formulation. A network flow problem is related with the daily scheduling of vehicles. A set covering structure defines the set of crew duties that daily cover the vehicle schedules and a mixed covering-assignment problem with additional constraints defines the roster, for the planning horizon H, that covers all daily crew duties.

4. DECOMPOSITION ALGORITHM

Different temporal scheduling problems may be identified in the above mathematical formulation: $|H|$ daily integrated vehicle-crew scheduling problems, with variables z and w for the vehicle schedules and for the crew duties, respectively; and a rostering problem for the whole planning horizon, with variables y, x and η. These temporal scheduling problems share a set of variables in constraint set (6). To handle these linking constraints, we propose a heuristic approach based on Benders decomposition. The decomposition heuristic alternates between the solution of a master problem involving the z, w variables, a vehicle-crew scheduling problem for each day of H, and the solution of the corresponding subproblem involving the y, x, η variables, a rostering problem for H.

Concerning the master problem, a non-exact approach is proposed where, in each iteration, the Benders cuts are relaxed into the master objective function, associated with non-negative multipliers. An adequate choice for these multiplier values leads to a relaxed master problem that can be partitioned into $|H|$ independent integrated vehicle-crew scheduling problems, one for each day of the planning horizon H. In each iteration, crew duty costs are updated with information given by the rostering suproblem variables and the resulting integrated vehicle-crew scheduling problems are solved by an algorithm which combines a heuristic column generation procedure with a branch-and-bound scheme.

As for the subproblem, fixing the values of the z and w variables in VCRPat at values $\bar{z}$ and $\bar{w}$, given by the optimal solution of the master problem, one obtains a rostering problem. Exact standard algorithms are used to solve the linear relaxation of the rostering subproblem. Whenever the resulting solution is not integer, branch-and-bound techniques are applied to obtain a feasible roster. Integer solutions for the roster subproblem involve a large number of binary variables and a great amount of resources is needed to obtain these solutions. To overcome such drawback, different strategies have been incorporated into the branching process, yielding in most cases, to a "good" feasible roster in short computing time.

A computational experiment was performed using two real-world data set instances. In each iteration, the master problem vehicle-crew solution and the subproblem rostering solution, in case of being integer, are both included in a pool of feasible solutions for VCRPat for further analyses from different points of view.

Preliminary computational results concerning pattern PatI show that the decomposition algorithm adjust crew duties in the master problem, thus inducing better subproblem solutions in what concerns the number of drivers and/or the number of short and long crew duties assigned to a driver. Such improvement on the first iteration solution quality may be seen through the replacement of long and short crew duties by normal crew duties originating a fairer distribution of work among the drivers. Note that, the first iteration solution corresponds to the sequential solution.

5. CONCLUSIONS

This paper proposes a Benders decomposition based algorithm that generates a pool of feasible solutions for a single depot vehicle-crew-roster problem. The approach outperforms the traditional sequential scheme. The feedback given by Benders cuts guided the building of the daily vehicle-crew schedules thus leading to balanced workload rosters with fewer drivers.

Due to the weight of driver costs in the VCRPat overall cost, the methodology is now being used to analyze the influence of different roster patterns into the VCRPat final solutions.

6. ACKNOWLEDGEMENTS

This research was funded by POCTI/ISFL/152.

7. REFERENCES

[1] D. Huisman, R. Freling, and A. Wagelmans, "Multiple-depot integrated vehicle and crew scheduling," *Transportation Sci-*

ence, vol. 39, no. 4, pp. 491–502, 2005.

[2] M. Mesquita and A. Paias, "Set partitioning/covering-based approaches for the integrated vehicle and crew scheduling problem," *Computers & Operations Research*, vol. 35, no. 5, pp. 1562–1575, 2008.

[3] I. Steinzen, V. Gintner, L. Suhl, and N. Kliewer, "A time-space network approach for the integrated vehicle-and crew-scheduling problem with multiple depots," *Transportation Science*, vol. 44, no. 3, pp. 367–382, 2010.

[4] A. Caprara, M. Monacci, and P. Toth, "A global method for crew planning in railway applications," in *Computer-Aided Scheduling of Public Transport*, Lecture Notes in Economics and Mathematical Systems, Springer, 2001, pp. 17–36.

[5] R. Freling, R. Lentink, and A. Wagelmans, "A decision support system for crew planning in passenger transportation using a flexible branch-and-price algorithm," *Annals of Operations Research*, vol. 127, no. 1-4, pp. 203–222, 2004.

[6] S. Chu, "Generating, scheduling and rostering of shift crew-duties: applications at the hong kong international airport," *European Journal of Operational Research*, vol. 177, no. 3, pp. 1764–1778, 2007.

[7] M. Mesquita, M. Moz, A. Paias, and M. Pato, "An integrated vehicle-crew-roster problem with days-off pattern," CIO-Working Paper 7, Tech. Rep., 2010.

[8] J.-F. Cordeau, G. Stojković, F. Soumis, and J. Desrosiers, "Benders decomposition for simultaneous aircraft routing and crew scheduling," *Transportation Science*, vol. 35, no. 4, pp. 375–388, 2001.

[9] A. Mercier, J.-F. Cordeau, and F. Soumis, "A computational study of benders decomposition for the integrated aircraft routing and crew scheduling problem," *Computers & Operations Research*, vol. 32, no. 6, pp. 1451–1476, 2005.

[10] A. Mercier and F. Soumis, "An integrated aircraft routing, crew scheduling and flight retiming model," *Computers & Operations Research*, vol. 34, no. 8, pp. 2251–2256, 2007.

[11] N. Papadakos, "Integrated airline scheduling," *Computers & Operations Research*, vol. 36, no. 1, pp. 176–195, 2009.

Insights on the exact resolution of the rostering problem

Marta Rocha * José Fernando Oliveira * Maria Antónia Carravilla *

* DEIG, Faculdade de Engenharia da Universidade do Porto
Rua Dr. Roberto Frias, Porto, Portugal
{marta, jfo, mac}@fe.up.pt

ABSTRACT

The purpose of this paper is to present some findings on the rostering problem resolution through the analysis of a real case study. The problem is initially formulated as a mixed integer problem (MIP) and solved with CPLEX, using the ILOG OPL Studio environment. The achieved findings and results are the basis for the development of a constructive heuristic that consistently reaches a feasible solution, which is the optimal solution in this particular case, in a shorter period of time than the MIP model.

Keywords: Rostering, Staff scheduling

1. INTRODUCTION

Human resource management is one of the most concerning issues for any organization due not only to its significant impact in the total expenditure but also because it is highly constrained by human behavior aspects seeing that it deals with human beings. Having the right people doing the right task, at the right time, in the right place, at the minimum cost is typically the aim of staff scheduling or rostering problems. These problems are restricted by several constraints such as demand requirements, task skills specifications, legal or contractual obligations, employees preferences, among others. A detailed description can be found in [1] and [2].

2. CASE STUDY

2.1. Problem description

The present work addresses the rostering problem of an organization, leader in its market segment, that works continuously, around the clock, 365 days per year. The workforce is divided in teams which must be assigned to three eight-hour shifts: morning, afternoon and night. The workload shall be uniformly distributed among teams, no distinction is made concerning skills of the employees or shift types. The problem consists in determining which team will work on each shift in each of the planning horizon days and how the rest or break days shall be interposed between workdays.

2.2. Developed MIP model

We developed a mixed-integer formulation for this problem, where the objective function is to minimize the maximum number of the days that a team works in each shift. The decision variables assume binary values indicating when a team is assigned to a shift on a specific day. This objective function levels the working days of each team, leading to a solution in which each team works the same number of days in each shift.

The constraints to this problem guarantee that:

1. each day, every team has exactly one shift assigned, either a work shift or a break shift.
2. each day, every working shift has exactly one team assigned.
3. no team works more than a maximum number of consecutive days.
4. each team works at least a minimum number of consecutive working days.
5. the required shift sequence is followed.
6. all teams have the same schedule, but with an offset between them.

2.3. Findings

We ran this model in the ILOG OPL Studio v6.3 and, although it led to some feasible solutions, it revealed the difficulty in finding the right combination of input parameters: given a number of teams and a number of shifts, which is the appropriate planning horizon? And what shall the offset between teams be?

In fact, the parameters of this problem are connected in such a way that make it very tight and inflexible to allow large parameters variations. Based on the tests results, we realized, for instance, that in order to get a feasible solution, the number of days in the planning horizon must be a multiple of the number of teams and that an offset between teams equal to the ratio between the number of days and the number of teams usually leads to a good result. We detected that a key issue, to which the model proved to be very sensitive, is the number of available break days. This value can be exactly determined for a given number of teams, number of shifts and number of days in the planning horizon. Considering constraints (1) and (2), we know that the number of breaks in each day is given by the difference between the number of teams and the number of shifts. The total number of breaks can be obtained by multiplying this figure by the number of days in the planning horizon. The number of available break days for each team is then achieved by dividing the total number of breaks by the number of teams.

We verified the existence of two important conditions:

1. considering the scenario of having working-day blocks with the minimum possible length (equal to the minimum number of consecutive working days), if the minimum number of required break-days is equal or less than the number of available breaks, the problem has a feasible solution. Otherwise, it may have or not a feasible solution. This condition is thus sufficient but not necessary.
2. considering the scenario of having working-day blocks with the maximum possible length, if the minimum number of required break-days is greater than the number of available breaks, the problem has no feasible solution. This is a necessary condition but not sufficient.

2.4. Developed constructive heuristic

When the first condition is met we propose a constructive heuristic to build a feasible solution, consisting of only blocks of minimum consecutive days length or a combination of these with minimum length + 1 day blocks. The first step is thus to check whether it is possible to use only minimum length blocks or if there is the need to use a combination of minimum length blocks and minimum length plus 1 day blocks. Then, we assign the first blocks of the first shift to all the teams, assuming an offset equal to minimum consecutive days or equal to the ratio of number of days and the number of teams. If we use minimum length blocks, we assign one break after each block. If we use minimum length and minimum length + 1 day blocks we assign two breaks after each of the former blocks and one break after the latter. After assigning the other shifts working blocks, we insert the required breaks at the end of the last shift block to wait until the first shift is available again. This closes the first sub-cycle. The procedure is concluded with the replication of the first sub-cycle as many times as needed in order to fulfill the planning horizon for all the teams.

This heuristic was manually tested with several input parameters combinations, which fulfilled the initial assumptions: number of days multiple of the number of teams and an offset equal to the number of minimum consecutive days or to the ratio between the number of days and the number of teams, and met the condition (1), leading always to a feasible solution. The development of this constructive procedure aimed to define a consistent and reliable process for reaching a feasible solution. The results achieved so far show, yet, that when a feasible solution is found, it is the optimal solution.

3. FUTURE WORK

Future work involves the software code development in order to massively test the heuristic, making it possible to consolidate and generalize the achieved results. We are confident that this work will provide an important contribution to the research in staff scheduling and rostering problems.

4. REFERENCES

[1] A. T. Ernst, H. Jiang, M. Krishnamoorthy, and D. Sier, "Staff scheduling and rostering: a review of applications, methods and models," *Eur. J. Oper. Res.*, vol. 153, no. 1, pp. 3–27, 2004.

[2] P. De Causmaecker and G. Vanden Berghe, "Towards a reference model for timetabling and rostering," *Annals of Operations Research*, pp. 1–10–10, 03 2010. [Online]. Available: http://www.springerlink.com/content/8024n8h207807504/

Comparing Hybrid Constructive Heuristics for University Course Timetabling

Dario Landa-Silva * Joe Henry Obit †

* ASAP Research Group, School of Computer Science
University of Nottingham, United Kingdom
dario.landasilva@nottingham.ac.uk

† Labuan School of Informatics Science
University Malaysia Sabah, Malaysia
Research carried out while J. Obit was a PhD student in Nottingham.
joehenryobit@yahoo.com

ABSTRACT

This extended abstract outlines four hybrid heuristics to generate initial solutions to the University course timetabling problem. These hybrid approaches combine graph colouring heuristics and local search in different ways. Results of experiments using two benchmark datasets from the literature are presented. All the four hybrid initialisation heuristics described here are capable of generating feasible initial timetables for all the test problems considered in these experiments.

Keywords: Course timetabling, Hybrid heuristics, Event scheduling, Constructive heuristics

1. INTRODUCTION

We refer to the University course timetabling problem as described by Socha et al. [1] with: n events $E = \{e_1, e_2, \ldots, e_n\}$, k timeslots $T = \{t_1, t_2, \ldots, t_k\}$, m rooms $R = \{r_1, r_2, \ldots, r_m\}$ and a set S of students. Each room has a limited capacity and some additional features. Each event requires a room with certain features. Each student attends a number of events which is a subset of E. The problem is to assign the n events to the k timeslots and m rooms in such a way that all hard constraints are satisfied and the violation of soft constraints is minimised.

The *hard constraints* that must be satisfied for a timetable to be feasible are as follows. *HC1*: a student cannot attend two events simultaneously, i.e. events with students in common must be timetabled in different timeslots. *HC2*: only one event may be assigned per timeslot in each room. *HC3*: the room capacity must be equal to or greater than the number of students attending the event in each timeslot. *HC4*: the room assigned to an event must satisfy the features required by the event. The *soft constraints* that are desirable to satisfy in order to assess the quality of a timetable are as follows. *SC1*: students should not have only one event timetabled on a day. *SC2*: students should not attend more that two consecutive events on a day. *SC3*: students should not attend an event in the last timeslot of a day.

It has been shown in the literature that a *sequential heuristic* method can be very efficient for generating initial solutions [2, 3]. A sequential heuristic assigns events one by one, starting from the event which is considered the most difficult to timetable in some sense. The 'difficulty' of scheduling an event can be measured by different criteria (i.e. the number of other conflicting events or the number of students attending the event). However, a sequential heuristic alone does not guarantee that feasible solutions will be found even with the combination of more than one heuristic. For example, Abdullah et al. [4] proposed a method, based on a sequential heuristic, to construct initial timetables. However, their method failed to generate a feasible solution for the large instance of the Socha et al. problem instances [1].

We propose hybrid heuristics to create initial feasible timetables for the University course timetabling problem described above. We combine traditional graph colouring heuristics with various local search methods including a simple tabu search. In the experiments of this work we use the 11 benchmark data sets proposed by Socha et al. [1] and also the set of problem instances from the International Timetabling Competition (ITC) 2002 [5]. The proposed heuristics generate feasible timetables for all the instances in our experiments. However, these methods do not tackle the satisfaction of soft constraints. Then, we obtain feasible solutions that might still have relatively high number of soft constraint violations. The rationale for this is to allow flexibility for another algorithm, that seeks to improve the satisfaction of constraints, to start the search from the feasible timetables. This has proven to be beneficial in our related work helping the improving algorithm to achieve extremely good results [6, 7]. It is difficult to compare the results in this paper with the literature because most other works (e.g. [3]) incorporate the construction of initial timetables within the overall method to solve the problem, i.e. constructing initial solutions and improving them are combined into a single approach. The next section describes the proposed hybrid heuristics.

2. GENERATING INITIAL TIMETABLES

In order to develop effective algorithms for tackling hard constraints in the subject problem, we combine techniques such as graph colouring, local search and tabu search. We found that the search components incorporated in the hybrid methods are interdependent on their ability to produce a feasible timetable. In other words, when one of these components is disabled or removed, the remaining components are not able to produce feasible solutions in particular for medium and large instances. Therefore, the hybrids described next are effective tailored mechanisms to generate feasible timetables for the subject problem.

2.1. Largest Degree, Local Search and Tabu Search (IH1)

We adopted the heuristic proposed by Chiarandini et al. [8] and added the Largest Degree heuristic to Step one as described next. Largest Degree refers to the event with the largest number of conflicting events (events that have at least one student in common).

Step one - Largest Degree Heuristic. In each iteration, the un-

scheduled event with the Largest Degree is assigned to a timeslot selected at random without respecting conflicts between events. Once all events have been assigned into a timeslot, the maximum matching algorithm for bipartite graph is used to assign each event to a room. At the end of this step, there is no guarantee for the timetable to be feasible. Then, steps one and two below are executed iteratively until a feasible solution is constructed.

Step two - Local Search. We employ two neighbourhood moves in this step. Move one (M1) selects one event at random and assigns it to a feasible pair timeslot-room also chosen at random. Move two (M2) selects two events at random and swaps their timeslots and rooms while ensuring feasibility is maintained. That is, neighbourhood moves M1 and M2 seek to improve the timetable generated in Step one. A move is only accepted if it improves the satisfaction of hard constraints (because the moves seek feasibility). This step terminates if no move produces a better (closer to feasibility) solution for 10 iterations.

Step three - Tabu Search. We apply a simple tabu search using a slight variation of move M1 above. Here, M1 only selects an event that violates hard constraints. The motivation is that the algorithm should now only target events that violate hard constraints instead of randomly rescheduling other events like in Step two. The tabu list contains events that were assigned less than tl iterations before calculated as $tl = rand(10) + \delta \times n_c$, where $0 \leq rand(10) \leq 10$, n_c is the number of events involved in hard constraint violations in the current timetable, and $\delta = 0.6$. The usual aspiration criterion is applied to override tabu status, i.e. accept the move when a best known solution is found. This step terminates if no move produces a better (closer to feasibility) solution for ts iterations.

2.2. Saturation Degree, Local Search and Tabu Search (IH2)

This heuristic uses Saturation Degree, which refers to the number of resources (timeslots and rooms) still available to timetable a given event without conflicts in the current partial solution. In the previous heuristic IH1 the assignment of events in Step one is done without checking conflicts. The difference in heuristic IH2 is that we first check conflicts between the unassigned event and those events already assigned to the selected timeslot. If there are timeslots with no-conflicting events already assigned (saturation degree of the event to assign is greater than zero), the event is assigned to a feasible timeslot selected at random. If there are no such timeslots (saturation degree of the event to assign is zero), the events already assigned to the timeslot are ejected and put in a list of events to re-schedule. The heuristic then attempts to re-assign these ejected events into conflict free timeslots if possible. Otherwise, these ejected events are put into random timeslot-room, even if conflicts arise, then later the local and tabu search of Step two and Step three as described above, will deal with these ejected events and the remaining conflicting assignments. In essence, in addition to using Saturation Degree instead of Largest Degree, this second heuristic IH2 tries to fix some conflicts in the timetable before starting Steps two and three.

2.3. Largest Degree, Saturation Degree, Local Search and Tabu Search (IH3)

This heuristic incorporates both Largest Degree and Saturation Degree. The difference with heuristic IH2 is that in Step one, events are first sorted based on Largest Degree. After that, we choose the unassigned event with the Largest Degree and calculate its Saturation Degree. Then, Step one of this heuristic IH3 proceeds as in heuristic IH2, but when attempting to re-assign the ejected events, only those ejected events with Saturation Degree greater than zero (still available timeslots and room) are assigned to any feasible timeslot-room. All ejected events with Saturation Degree zero are moved from the re-schedule list to the list of unscheduled events. After each re-assigning, we re-calculate the Saturation Degree for all ejected events in the re-schedule list. This process in Step one continues and if after some given computation time there are still events in the unscheduled list, these events are then assigned to random timeslot-room without respecting conflicts. Steps one and two as described above follow implementing the local and tabu search respectively. In essence, compared to heuristic IH2, this heuristic IH3 combines Saturation Degree and Largest Degree in Step one trying to re-scheduled ejected events with less resources first. Algorithm 1 shows the pseudo-code for the hybrid heuristic IH3, which in a sense, is the most elaborate one among methods IH1, IH2 and IH3.

2.4. Constraint Relaxation Approach (IH4)

In this fourth heuristic approach, we introduce extra *dummy* timeslots to place events with zero Saturation Degree and in this way enforce the no-conflicts constraint by relaxing the availability of timeslots. The number of extra dummy timeslots needed is determined by the size of the problem instance. This heuristic works as follows. First, we sort the events using Largest Degree. The event with the Largest Degree is chosen to be scheduled first. If the event has zero Saturation Degree, the event is assigned randomly to one of the extra dummy timeslots. Once the algorithm assigns all events in the valid timeslots plus the extra *dummy* timeslots without conflicts, we then perform great deluge search [6] using moves M1 and M2 to reduce the number of timeslots down to 45 valid timeslots if necessary. In this local search, only the 45 valid timeslots are considered, so no events are allowed to move into any of the extra dummy timeslots. This hybrid heuristic is much slower that the other three methods above, mainly due to the great deluge search. Algorithm 2 shows the pseudo-code for the hybrid heuristic IH4, which in a sense, is the most different among all methods described here.

3. RESULTS AND DISCUSSION

The proposed hybrid heuristic initialisation methods were applied to the Socha et al. [1] instances and also to the ITC 2002 instances [5]. We did not impose time limit as a stopping condition, each algorithm stops when it finds a feasible solution.

All methods successfully generate initial solution for small instances in just few seconds. The medium and large Socha et al. instances are more difficult as well as all ITC 2002 instances. However, the proposed methods generated feasible solutions for all instances demonstrating that the hybridisation compensates weakness in one component with strengths in another one in order to produce feasible solutions in reasonable computation times.

Table 1 and Table 2 compare the performance of each method on the Socha et al. and the ITC 2002 instances respectively. The first column in each table indicates the problem instance. The next four columns give the best objective function value (soft constraints violation) obtained by each heuristic. The last column in each table indicates the best computation time in seconds and the corresponding heuristic.

The results show that none of the heuristics clearly outperforms the others in terms of the objective function value (soft constraints violation) obtained. Each of the four heuristics outperforms the other three in some of the problem instances. With respect to computation time we can see in Table 1 that for the Socha et al. problems, the heuristic that achieved the best objective value was almost never the fastest one (except in problem instance M2). However, for the ITC 2002 problems, we see in Table 2 that in several cases the heuristic producing the best objective value was also the

Algorithm 1: Initialisation Heuristic 3 (IH3)

Input: List of *Unscheduled* events E;
Sort E by non-increasing Largest Degree (LD);
while *(E is not empty)* **do**
  Choose event e from E with LD (random tie-break);
  Calculate *SD* for event e;
  if *($SD = 0$)* **then**
    Select a timeslot t at random;
    Move events scheduled (if any) in timeslot t that conflict with event e (if any) to the *Reschedule* list;
    Assign event e to timeslot t;
    for *(*each event in *Reschedule list* with $SD > 0$*)* **do**
      Select feasible timeslot t for event e at random;
      Re-calculate SD for all events in *Reschedule* list;
    end
    Move all events with $SD = 0$ that remain in *Re-schedule* list to the *Unscheduled* list E;
  end
  else
    Select a feasible timeslot t at random for event e;
  end
  if *($Unscheduled$* list E is not empty and *time* has elapsed*)* **then**
    One by one, place events from the *Unscheduled list* into any random selected timeslot without respecting the conflict between the events;
  end
end
S = current solution;
$loop = 0$;
while *(S not feasible)* **do**
  if *($loop < 10$)* **then**
    if *($coinflip()$)* **then**
      S^* = M1(S); // apply M1 to S
    end
    else
      S^* = M2(S); // apply M2 to S
    end
    if *($f(S^*) \leq f(s)$)* **then**
      $S \leftarrow S^*$ // accept new solution;
    end
  end
  else
    EHC = set of events that violate hard constraints;
    e = randomly selected from EHC;
    S^* = M1(S, e); // perform one Tabu Search iteration with move M1 using event e;
    if *($f(S^*) < f(S)$* **then**
      $S \leftarrow S^*$; // accept new solution
    end
    if *($loop >= ts$)* **then**
      $loop = 0$;
    end
  end
  $loop++$;
end
Output: S feasible solution (timetable);

Algorithm 2: Initialisation Heuristic 4 (IH4)

Input: List of *Unscheduled* events E;
Generate *dummy* timeslots according to problem instance;
Sort events in E by non-increasing Largest Degree (LD);
while *($Unscheduled$* list E is not empty*)* **do**
  Choose event e from E with the LD (random tie-break);
  Calculate *SD* for event e;
  if *($SD = 0$)* **then**
    Select *dummy* timeslot at random for event e;
  end
  else
    Chose any feasible timeslot for event e;
    Update the new solution;
  end
end
S = current solution;
Calculate initial cost function $f(S)$;
Initial water level $B = f(S)$;
ΔB = 0.01;
while *(dummy* timeslots are not empty*)* **do**
  if *($coinflip()$)* **then**
    S^* = M1(S); // apply M1 to S
  end
  else
    S^* = M2(S); // apply M2 to S
  end
  if *($f(s^*) \leq f(s)$) or ($f(s^*)(\leq B)$)* **then**
    $S \leftarrow S^*$; // accept new solution
  end
  $B = B - \Delta B$; // lower the water level
  if *(B - f(S)* $\leq$ *1)* **then**
    $B = B + 5$; // increase the water level
  end
end
Output: S feasible solution (timetable);

fastest. As indicated above, the hybrid initialisation heuristic (IH4) that uses *dummy* timeslots to deal with conflicts and then great deluge as the local search to bring the solution to feasibility, is never the fastest approach. However, this heuristic IH4 was capable of producing the best solutions for two of the Socha et al. instances and six of the ITC 2002 instances.

In our preliminary experiments, we implemented a sequential heuristic (see [2, 3]) but were able to generate feasible timetables only for the small instances of the Socha et al. dataset (in fact, these small instances are considered to be easy). Even after considerably extending the computation time, the sequential heuristic was not able to generate feasible solutions for the medium and large Socha et al. instances or the ITC 2002 datasets.

4. CONCLUSIONS

Many approaches have been proposed in the literature to tackle the University course timetabling problem. In this extended abstract we have outlined four variants of hybrid heuristics designed to generate initial feasible solutions to this problem. These hybrid approaches combine traditional graph colouring heuristics, like Largest Degree and Saturation Degree, with different types of local search. The four hybrid variants were tested using two sets of benchmark problem instances, the Socha et al. [1] and the International Timetabling Competition 2002 [5] datasets.

All the hybrid initialisation heuristics described here were capable of producing feasible timetables for all the problem instances.

Problem	IH1	IH2	IH3	IH4	Min Time
S1	**173**	198	207	200	0.077 (IH2)
S2	211	217	**189**	208	0.078 (IH2)
S3	**176**	190	188	209	0.062 (IH2)
S4	250	**174**	203	192	0.047 (IH1)
S5	229	238	226	**217**	0.078 (IH2)
M1	817	**772**	802	774	5.531 (IH3)
M2	793	**782**	784	802	6.342 (IH2)
M3	**795**	867	828	817	6.64 (IH3)
M4	**735**	785	811	795	5.828 (IH2)
M5	773	771	784	**769**	16.670 (IH1)
L	**1340**	1345	1686	1670	300.0 (IH1)

Table 1: Results obtained with each hybrid initialisation heuristic (IH1 to IH4) on the 11 Socha et al. problem instances, best results indicated in bold.

Problem	IH1	IH2	IH3	IH4	Min Time
Com01	805	**786**	805	805	1.93 (IH3)
Com02	**731**	776	**731**	778	1.36 (IH3)
Com03	**760**	812	**760**	777	1.14 (IH2)
Com04	1201	**1178**	1201	1236	4.46 (IH2)
Com05	1246	1243	1246	**1135**	2.11 (IH3)
Com06	1206	1219	1206	**1133**	1.33 (IH3)
Com07	1391	1388	1391	**1265**	2.10 (IH3)
Com08	1001	**968**	1001	1006	1.81 (IH2)
Com09	**841**	859	**841**	843	1.46 (IH1)
Com10	**786**	816	**786**	799	4.64 (IH3)
Com11	852	877	852	**839**	1.05 (IH1)
Com12	814	831	814	**788**	2.21 (IH2)
Com13	**1008**	1010	**1008**	1009	2.26 (IH1)
Com14	1040	**1032**	1040	1355	3.71 (IH2)
Com15	1165	1162	1165	**1161**	1.56 (IH3)
Com16	**887**	911	**887**	888	1.09 (IH3)
Com17	1227	**1032**	1227	1199	1.13 (IH2)
Com18	793	**724**	793	763	1.29 (IH3)
Com19	**1184**	1212	**1184**	1209	3.22 (IH3)
Com20	**1137**	1161	**1137**	1205	0.08 (IH3)

Table 2: Results obtained with each hybrid initialisation heuristic (IH1 to IH4) on the 20 ITC 2002 problem instances, best results indicated in bold.

None of the approaches showed to be clearly better that the others. For a given instance, the heuristic producing the best quality initial timetable is often not the fastest among the four approaches. However, for all the problem instances there is at least one hybrid heuristic capable of generating a feasible timetable in very short time, from less than a second to few seconds depending of the problem instance. The exception is the largest Socha et al. instance which is still regarded in the literature as a very challenging problem. Having some methods capable of generating feasible solutions for the University course timetabling problem is important because the effort of more elaborate methods can then be focused on tackling the violation of soft constraints in order to improve the timetable quality.

In a following more detailed description on this research, we intend to present a statistical comparison between the proposed initialisation heuristics, compare these approaches against other procedures to generate feasible solutions to the University course timetabling problem and analyse the effect of each component in the four hybrid heuristics.

5. REFERENCES

[1] K. Socha, J. Knowles, and M. Samples, "A max-min ant system for the university course timetabling problem," in *Ant Algorithms: Proceedings of the Third International Workshop (ANTS 2002), LNCS 2463*. Springer, 2002, pp. 1–13.

[2] E. Burke, B. A. McCollum, A. Meisels, S. Petrovic, and Q. Rong, "A graph based hyper-heuristic for educational timetabling problems," *European Journal of Operational Research*, vol. 176, pp. 177–192, 2007.

[3] P. Kostuch, "The university course timetabling problem with a three-phase approach," in *The Practice and Theory of Automated Timetabling V, LNCS 3616*. Springer, 2005, pp. 109–125.

[4] S. Abdullah, E. Burke, and B. McCollum, *Using a Randomised Iterative Improvement Algorithm with Composite Neighborhood Structures for University Course Timetabling*. Springer, 2007, pp. 153–172.

[5] B. Paechter, L. M. Gambardella, and O. Rossi-Doria. (2002) International timetabling competition 2002. Metaheuristics Network. [Online]. Available: http://www.idsia.ch/Files/ttcomp2002/

[6] J. H. Obit and D. Landa-Silva, "Computational study of nonlinear great deluge for university course timetabling," in *Intelligent Systems - From Theory to Practice, Studies in Computational Intelligence, Vol. 299*, V. Sgurev, M. Hadjiski, and J. Kacprzyk, Eds. Springer-Verlag, 2010, pp. 309–328.

[7] J. H. Obit, "Developing novel meta-heuristic, hyper-heuristic and cooperative search for course timetabling problems," Ph.D. dissertation, 2010.

[8] M. Chiarandini, M. Birattari, K. Socha, and O. Rossi-Doria, "An effective hybrid algorithm for university course timetabling," *Journal of Scheduling*, vol. 9, pp. 403–432, 2006.

Lower and upper bounds for large size instances of the optimal diversity management problem

Agostinho Agra * Jorge Orestes Cerdeira † Cristina Requejo *

* Department of Mathematics, University of Aveiro
3810-193 Aveiro, Portugal
{aagra, crequejo}@ua.pt

† Department of Sciences and Engineering of Biosystems
Instituto Superior de Agronomia, Technical University of Lisbon (TULisbon)
1349-017 Lisboa, Portugal
orestes@isa.utl.pt

ABSTRACT

We give procedures to derive lower and upper bounds for the optimal diversity management problem, especially conceived to deal with real instances that occur in the production of wire harness for the automotive industry. We report computational results to assess the quality of these bounds.

Keywords: Integer programming, Duality, Heuristics, P-median

1. INTRODUCTION

In the production of wire harness for the automotive industry decisions have to be made on the configurations that should be manufactured in order to satisfy, within reasonable production costs, a possible large variety of customers' requests. Specifically, cars are assembled with the necessary wire connections to activate the set of requested options such as airbags, air conditioned, etc. A configuration is the aggregate of minimum connections allowing to activate a given group of options. The set of requested options vary greatly depending on client's preferences. In theory, there can be millions of different combinations of options. Since wire harness is mainly manually assembled, in practice only a small number p of different configurations is settled and customers are often supplied with cars having wire harness including unnecessary wire connections. Clearly, this gives rise to production extra costs (associated with copper wires waste) making the selection of the p configurations to produce an important issue.

This problem, called the Optimal Diversity Management Problem (ODMP), is a special case of the well known p-median problem. The p-median problem [1, 2] seeks to select p vertices (the medians) of a digraph with weights on the arcs, in order to minimize the sum of the weights of the arcs linking each non-median vertex to one of the selected medians.

The ODMP, that was shown to be NP-hard [3, 4], is the p-median for transitive digraphs. This is the case of the graph resulting from the wire harness application above, which, in addition, usually consists of several connected components.

An extensive study on the ODMP is developed in the PhD thesis of Briant [3], which is the first substantial work on this problem. Briant [3] pointed out that the large size of instances of real problems is a serious barrier to the efficiency of the algorithms. Dealing with the huge size instances that appear in real problems is the main concern in studies on the ODMP [5, 6, 7, 4].

In this study we give ways to obtain lower and upper bounds on the values of optimal ODMP solutions, specifically meant to deal with the huge graphs arising from the wire harness application, and exploiting the fact that these graphs have several components. Computational results are reported to assess the quality of these bounds on real instances.

2. FORMULATION

To formulate the ODMP consider a weighted transitive digraph $G = (V,A,c)$, where the vertices of $V = \{1,\dots,n\}$ represent the configurations, and (u,v) is an arc of A if and only if every option that configuration u allows to activate could also be activated by v. We say that v covers u (or that u is covered by v). Each configuration v can be interpreted as the subset of options that v activates. Each arc $a = (u,v)$ of G has a cost c_a, which is the cost of using configuration v to substitute u.

An important property of real ODMP instances is that graph G has several connected components. We denote by $K = \{1,\dots,m\}$ the set of indices of the connected components, and by $G_k = (V_k,A_k)$ the subgraph induced by component k, with $k \in K$.

Let, for $v \in V$, y_v be a 0-1 variable indicating whether vertex v is selected ($y_v = 1$) or not ($y_v = 0$) to be a median. Let, for $(u,v) \in A$, x_{uv} be a 0-1 variable indicating whether configuration represented by vertex v will replace ($x_{uv} = 1$) or not ($x_{uv} = 0$) the configuration represented by u. Consider, in addition, for $k \in K$, a positive integer variable p_k, that indicates the number of medians in component k. With these variables the ODMP can be modeled as follows.

$$\min \sum_{v \in V} \sum_{u \in \delta^+(v)} c_{vu} x_{vu} \tag{1}$$

subject to

$$\sum_{u \in \delta^+(v)} x_{vu} + y_v = 1 \qquad v \in V \tag{2}$$

$$\sum_{v \in V_k} y_v = p_k \qquad k \in K \tag{3}$$

$$\sum_{k \in K} p_k = p \tag{4}$$

$$x_{vu} \le y_u \qquad v \in V, u \in \delta^+(v) \tag{5}$$

$$y_v \in \{0,1\} \qquad v \in V \tag{6}$$

$$x_{vu} \in \{0,1\} \qquad v \in V, u \in \delta^+(v) \tag{7}$$

$$p_k \in \mathbb{N} \qquad k \in K \tag{8}$$

where $\delta^+(v) = \{u \in V : (v,u) \in A\}$.

Equations (2) state that either v is a median or v must be linked to some vertex. Equations (3) and (4) guarantee that the medians in the m connected components sums p. Inequalities (5) express that if a node is not a median, its indegree is equal to zero. Finally (6)-(8) define the range of the variables

Note that if the values of p_k, say p_k^*, of an optimal ODMP solution were known, an optimal ODMP solution would result from the union of optimal p_k^*-medians of each component k.

Finding $p_1^*, p_2^*, \ldots, p_m^*$, with $p_1^* + p_2^* + \ldots + p_m^* = p$, and such that the union of optimal p_k^*-medians is ODMP optimal, is the decomposition problem for the ODMP [6].

The decomposition problem can be modeled as a particular case of the multiple choice knapsack problem ([8, 9]), and can be solved efficiently ([9, 10]). Hence, the ODMP problem can be decomposed in smaller similar subproblems. However, since p_k^* is not known in advance, each subproblem k has to be solved, in principal, with a number of medians equal to $1, \ldots, p-m+1$.

3. LOWER BOUNDS

Lower bounds for the ODMP are usually obtained by Lagrangean relaxation techniques [5, 11]. We use a heuristic for the dual of the linear relaxation of formulation (1) - (8), which is similar to the procedure proposed in [12] to solve the dual of a linear relaxation of the p-median problem.

Let λ_v, with $v \in V$, be the dual variables associated with equations (2); s_k, with $k \in K$, dual variables associated to (3); γ associated to (4) and t_{vu}, with $(v,u) \in A$, the non-negative dual variables corresponding to inequalities (5).

The dual of the linear programming relaxation of formulation (1) - (8) is as follows

$$\max \quad \sum_{v \in V} \lambda_v - p\gamma \tag{9}$$

subject to

$$\lambda_v - s_k + \sum_{(u,v) \in A} t_{uv} \leq 0 \quad v \in V_k, k \in K \tag{10}$$

$$\lambda_v - c_{vu} \leq t_{vu} \quad v \in V, u \in \delta^+(v) \tag{11}$$

$$s_k \leq \gamma \quad k \in K \tag{12}$$

$$t_{vu} \geq 0 \quad v \in V, u \in \delta^+(v) \tag{13}$$

The dual variables $s_k$and t_{vu}can be removed from the model, yielding

$$\max \quad \sum_{v \in V} \lambda_v - p\gamma \tag{14}$$

subject to

$$\gamma \geq \lambda_v + \sum_{(u,v) \in A} (\lambda_u - c_{uv}) \quad v \in V \tag{15}$$

The heuristic solution is defined assigning to γ the value of the r.h.s. of (15), with $\lambda_v := \min_{u \in \delta^+(v)} c_{vu}$.

4. UPPER BOUNDS

The greedy algorithm has been used to solve large size instances of the ODMP, and studies refer that the resulting solutions are normally quite good [3, 6, 4].

It is worth mention that in [9] it is shown that the following procedure:
Step 1: run the greedy algorithm for each connected component and for all the possible choices of medians;
Step 2: solve the decomposition problem using the values of greedy solutions obtained on each component,
provides the same objective function value than running the greedy algorithm over the entire graph.

We consider running a genetic algorithm for the ODMP in each connected component k of graph G, for different number of medians p_k, and use the approach in [10] to solve the resulting *restricted* decomposition problem to obtain a final ODMP solution.

Instead of considering all possible number of medians for component k, we restrict p_k to vary in the interval defined by the minimum and maximum number of medians that the greedy has determined for component k when solving the ODMP with p, $p-1$ and $p+1$ medians.

In order to take advantage from the knowledge of the greedy solution, in the implementation of the genetic algorithm, we included in the initial *population* (i.e., a collection of random selected p_k medians) for component k, what we call the *modified* k component greedy solution.

The modified k component greedy solution is a set of p_k medians on the connected component k resulting from adding (or deleting) uniformly selected vertices of component k to (from) the set of medians defined by the greedy for that component.

5. COMPUTATIONAL RESULTS

Here we report some computational experience carried out to evaluate the quality of the proposed lower and upper bounds on the optimal ODMP values.

All the computational tests were performed on a PC running on an Intel(R) Core(TM)2 Duo CPU 2.00 GHz processor and 1.99Gb of RAM. We used real data instances from the *Yazaki Saltano de Portugal*, a branch of Yazaki, the world's largest producer of wire harness, consisting of graphs with 3072, 10848, 15360, 22080, and 51840 vertices and, for each one, we tested $p \in \{50, 100, 150, 200\}$.

In order to obtain the optimal values we use the optimization software Xpress 7.1 with a limit for the computations on each instance equal to three CPU hours.

Table 1 reports the main computational results. The first two columns indicate the number of vertices (n) and the number of connected components of the graph (m). The third column specifies the number of medians (p). The remaining columns indicate the values found for the lower bounds (LB), the optimal solutions (OPT) when they were found, the greedy solutions (Greedy) and the upper bounds corresponding to the values of the solutions produced by the genetic algorithm (UB-Genetic). The values in brackets refer to the computational times, in CPU seconds, to determine the corresponding values.

It can be concluded from Table 1 that both greedy and dual solutions provide tight bounds for the real instances considered. The genetic algorithm was capable in most cases to slightly improve the greedy solutions.

It should be mentioned that the inclusion of the (modified) greedy solution in the initial population proved to be essential to obtain good solutions. Computational tests showed that the genetic algorithm working on an initial randomized population not including the greedy solution provides, in general, poor solutions when compared with the greedy solutions, and with a larger computational effort.

It is also worth noting that, when the greedy solution is compared with the optimal solution (for those instances it was obtained), in

n	m	p	LB	OPT	Greedy	UB-Genetic
3072	8	50	141 088 (1)	143 800 (24)	144 698 (1)	144 209 (40)
3072	8	100	79 838 (2)	82 035 (73)	82 365 (1)	82 074 (61)
3072	8	150	54 963 (1)	55 472 (69)	56 022 (1)	55 682 (85)
3072	8	200	40 155 (1)	40 829 (72)	41 066 (1)	40 908 (147)
10848	46	50	10 520 027 (6)	10 528 136 (41)	10 528 136 (1)	10 528 136 (1)
10848	46	100	3 216 662 (16)	3 389 989 (647)	3 508 534 (1)	3 431 425 (44)
10848	46	150	2 032 277 (16)	2 158 828 (2085)	2 253 169 (1)	2 203 339 (89)
10848	46	200	1 469 477 (19)	1 576 485 (3184)	1 657 811 (1)	1 617 045 (113)
15360	14	50	2 933 (626)	-	3 239 (8)	3 197 (5694)
15360	14	100	1 653 (688)	-	1 874 (17)	1 847 (406)
15360	14	150	1 174 (543)	-	1 314 (27)	1 298 (3442)
15360	14	200	925 (639)	-	1 013 (37)	1 000 (1047)
22080	16	50	2 099 492 (6374)	-	2 279 053 (63)	2 241 344 (1421)
22080	16	100	1 101 529 (3945)	-	1 248 647 (143)	1 212 547 (3199)
22080	16	150	752 955 (4020)	-	863 155 (202)	845 006 (1943)
22080	16	200	584 859 (12013)	-	662 480 (300)	653 190 (10295)
51840	60	100	2 354 812 (824)	-	2 396 102 (33)	2 358 739 (1049)
51840	60	150	1 364 717 (950)	-	1 432 499 (51)	1 401 920 (334)
51840	60	200	824 682 (1938)	-	1 043 544 (74)	1 018 434 (497)

Table 1: Computational results.

most cases, the numbers of medians per connected component coincide, and for most of the remaining cases, the differences do not exceed one. This means that greedy solutions give reliable estimative for the number of medians in each component of optimal solutions. Thus, solving the decomposition problem only considering a few number of medians close to the number of medians determined by the greedy solution on each connected component, is likely to be a good strategy for solving the ODMP.

6. CONCLUSION

The ODMP is a combinatorial optimization problem arising in the production industry of wire harness for the automotive. With special attention to the fact that the graphs arising from this application are very large and have several connected components, we proposed ways of obtaining lower and upper bounds. Lower bounds were obtained through a heuristic for the dual of the linear relaxation of a model for the ODMP. Upper bounds were obtained through a genetic algorithm running in each component of the graph, and benefiting from the knowledge of a greedy solution to combine the partial solutions into a feasible ODMP result. We intend to extend this approach further, exploiting a specific behavior of the ODMP objective function with respect to the number of medians, to determine a narrow range for the number of medians in the sub-problems that include an optimal decomposition.

7. ACKNOWLEDGEMENTS

The research of the second author was supported by Forest Research Centre (Centro de Estudos Florestais), the research of the other authors was supported by Center for Research and Development in Mathematics and Applications (CIDMA) both from the Portuguese Foundation for Science and Technology (FCT), cofinanced by the European Community Fund FEDER/POCI 2010.

8. REFERENCES

[1] P. Mirchandani and R. Francis, Eds., *Discrete Location Theory*. John Wiley & Sons, 1990.

[2] J. Reese, "Solution methods for the p-median problem: an annotated bibliography," *Networks*, vol. 48, pp. 125–142, 2006.

[3] O. Briant, "Étude théorique et numérique du problème de la gestion de la diversité," Ph.D. dissertation, Institute Nacional Polytechnique de Grenoble, Grenoble, France, 2000.

[4] A. Agra, D. Cardoso, J. Cerdeira, M. Miranda, and E. Rocha, "Solving huge size instances of the optimal diversity management problem," *Journal of Mathematical Sciences*, vol. 161, pp. 956–960, 2009.

[5] O. Briant and D. Naddef, "The optimal diversity management problem," *Operations Research*, vol. 52, no. 4, pp. 515–526, 2004.

[6] P. Avella, M. Boccia, C. D. Martino, G. Oliviero, A. Sforza, and I. Vasil'ev, "A decomposition approach for a very large scale optimal diversity management problem," *4OR*, vol. 3, pp. 23–37, 2005.

[7] P. Avella, A. Sassano, and I. Vasil'ev, "Computational study of large-scale p-median problems," *Mathematical Programming*, vol. 109, no. 1, pp. 89–114, 2007.

[8] H. Kellerer, U. Pferschy, and D. Pisinger, *Knapsack Problems*. Springer, 2004.

[9] A. Agra and C. Requejo, "The linking set problem: a polynomial special case of the multiple-choice knapsack problem," *Journal of Mathematical Sciences*, vol. 161, pp. 919–929, 2009.

[10] D. Cardoso and J. Cerdeira, "Minimum weight t-decomposition of an integer," to appear in Journal of Mathematical Sciences.

[11] A. Santos, "Solving large $p-$median problems using a lagrangean heuristic," 2009, Optimization Online.

[12] M. Captivo, "Fast primal and dual heuristics for the p-median location problem," *European Journal of Operational Research*, vol. 52, pp. 65–74, 1991.

Continuous Ant Colony System Applied to Optimization Problems with Fuzzy Coeficients

Luiza Amalia Pinto Cantão * Ricardo Coelho Silva † Akebo Yamakami †

* UNESP – Univ Estadual Paulista, Campus of Sorocaba, Environmental Engineering Dept.
Av. Três de Março, 511, 18087-180, Sorocaba – SP, Brazil
luiza@sorocaba.unesp.br

† UNICAMP – Univ. Estadual of Campinas, School of Electrical and Computer Engineering
P.O. Box 6011, 13083-970, Campinas – SP, Brazil
{rcoelhos, akebo}@dt.fee.unicamp.br

ABSTRACT

Heuristic algorithms based in ant colonies (named ant system – AS for short) were developed by Marco Dorigo to solve combinatorial optimization problems as the traveling salesman problem. This class of algorithms was also adapted by Seid H. Pourtakdoust and Hadi Nobahari for continuous optimization problems (Continuous Ant Colony Optimization Systems – CACS). In this work, an implementation of CACS was used for nonlinear continuous optimization problems with coefficients represented by fuzzy numbers. The fuzzy numbers are modelled through symmetric triangular membership functions, Possibility Measure — based on Didier Dubois and Henri Prade's work for comparison of functions with fuzzy values — and centroid defuzzification methods to obtain the ordinary value from function values in the pheromone evaluation step. Experiments with nine benchmark functions show a good agreement — considering the imprecise nature of the problem — between the fuzzy optima and their real counterparts.

Keywords: Ant Colony System, Optimization, Fuzzy Theory, Possibility Theory

1. INTRODUCTION

Ant Colony System was developed based in the *Traveling Salesman Problem*. The ideas behind the system make it suitable for high complexity combinatorial optimization problems demanding discrete solutions. The heuristic algorithm inspired in ant colonies was developed by M. Dorigo and coleagues as we can see in [1], [2], for example.

An extension of this algorithm for continuous function optimization was proposed by several authors, as in [9], [12] and [13], among others. The work of S.H. Pourtakdoust and H. Nobahari as in [8] and [7] is an example of such an extension, with the added bonus of a simpler structure for the application of fuzzy parameters on its formulation.

The purpose of this work is the introduction of fuzzy parameters into an Ant Colony System heuristic applied to Fuzzy Mathematical Programming. The fuzzy parameters are treated as fuzzy numbers (see [6]) with a double intent here: (i) to model the fuzzy parameters and (ii) make the fuzzy algebrical operations. Of course other changes are required in order to accomodate the fuzzy numbers, namely a convenient comparison operation between fuzzy quantities, an approach to evaluate the fuzzy function through ranking presented in [3] and [4], and finally a defuzzification process, as in [11].

The results are satisfactory, showing that Continuous Ant Colony Systems can be a valid alternative to treat Fuzzy Mathematical Programming problems.

2. PRELIMINARIES

In this section, we explain some topics of Fuzzy Theory used in this work.

Definition A fuzzy set $\widetilde{C}$ on R is a fuzzy number, if its *membership function* is defined as follows:

$$\mu_{(\widetilde{C})}(x) = \begin{cases} 0 & \text{if} \quad x \geq \underline{c} \\ \frac{x-\underline{c}}{c-\underline{c}} & \text{if} \quad x \in [\underline{c}, c] \\ \frac{\overline{c}-x}{\overline{c}-c} & \text{if} \quad x \in [c, \overline{c}] \\ 0 & \text{if} \quad x \geq \overline{c} \end{cases} \tag{1}$$

where $\mu_{(\widetilde{C})}(x) : R \to [0,1]$, c is the modal value, i. e., $\mu_{(\widetilde{C})}(c) = 1$, $\underline{c}$ and $\overline{c}$ are the inferior and superior limits, respectively.

We suppose that $f^L_{\widetilde{c}} : [\underline{c}, c] \to [0,1]$ and $f^R_{\widetilde{c}} : [c, \overline{c}] \to [0,1]$ are two continuous mappings from the real line R to the closed interval $[0,1]$. The former is a strictly increasing function and the later is a monotonically decreasing function. In this case, we assume that de fuzzy number is represented by a triangular function, i. e., $\widetilde{C} = (c, \underline{c}, \overline{c})$.

In order to facilitate the operations with fuzzy numbers, we assume that an exact membership function can be approximated by using piecewise linear functions based on α-level sets.

Definition [11] Let $\widetilde{C}$ be a fuzzy number. Its α-level sets $\widetilde{C}_\alpha$ or α-cuts are defined as

$$\begin{aligned} \widetilde{C}_\alpha &= \{x \in R \,|\, \mu_{\widetilde{C}}(x) \geq \alpha\} \\ &= [\min\{x \in R \,|\, \mu_{\widetilde{C}}(x) \geq \alpha\}, \max\{x \in R \,|\, \mu_{\widetilde{C}}(x) \geq \alpha\}] \\ &= [(x)^L_\alpha, (x)^U_\alpha] \quad 0 < \alpha \leq 1 \end{aligned} \tag{2}$$

Acording to Zadeh's extension principle [4], the fuzzy number $\widetilde{C}$ can also be expressed as

$$\widetilde{C} = \bigcup_\alpha \alpha \cdot \widetilde{C}_\alpha, \quad 0 < \alpha \leq 1. \tag{3}$$

The α-levels representation is used to operate with fuzzy numbers, as shown in [6]; all other operations follow the structure presented in this reference. They are also useful in the estimation of a representative ordinary number — a process known as defuzzification. In this particular setting we used the centroid defuzzification

method, as in [11]. This method defines the centroid of $\widetilde{C}$ as the x-axis value of the centroid as its defuzzification value, which can be expressed as:

$$D(\widetilde{C}) = \frac{\int_{\underline{c}}^{\overline{c}} x\mu_{\widetilde{C}}(x)\,dx}{\int_{\underline{c}}^{\overline{c}} \mu_{\widetilde{C}}(x)\,dx} \tag{4}$$

where

$$\begin{aligned}
\int_{\underline{c}}^{\overline{c}} \mu_{\widetilde{C}}(x)\,dx &= \frac{1}{2n}\left[\left((x)^U_{\alpha_0} - (x)^L_{\alpha_0}\right) + 2\sum_{i=1}^{n-1}\left((x)^U_{\alpha_i} - (x)^L_{\alpha_i}\right)\right], \\
\int_{\underline{c}}^{\overline{c}} x\mu_{\widetilde{C}}(x)\,dx &= \frac{1}{6n}\left[\left((x)^{2U}_{\alpha_0} - (x)^{2L}_{\alpha_0}\right) + 2\sum_{i=1}^{n-1}\left((x)^{2U}_{\alpha_i} - (x)^{2L}_{\alpha_i}\right)\right. \\
&\quad \left. + \sum_{i=1}^{n-1}\left((x)^U_{\alpha_i}\cdot(x)^U_{\alpha_{i+1}} - (x)^L_{\alpha_i}\cdot(x)^L_{\alpha_{i+1}}\right)\right].
\end{aligned} \tag{5}$$

In order to compare (or rank) fuzzy numbers, $\widetilde{C}_1$ and $\widetilde{C}_2$ for instance, we can apply a comparison measure built upon the Possibility Measure, as presented in [3] and [4]. In this context, if we want to decide wether $\widetilde{C}_1 > \widetilde{C}_2$ or not, we use the following mesure (remembering that $\widetilde{C}_1 = (c_1, \underline{c}_1, \overline{c}_1)$ and $\widetilde{C}_2 = (c_1, \underline{c}_2, \overline{c}_2)$):

$$\begin{aligned}
Pos(\widetilde{C}_1 \geq \widetilde{C}_2) &= \max\left(0, \min\left(1, 1 + \tfrac{(c_1 - c_2)}{\underline{c}_1 + \overline{c}_2}\right)\right) \quad \text{(PSE)} \\
Pos(\widetilde{C}_1 > \widetilde{C}_2) &= \max\left(0, \min\left(1, \tfrac{c_1 - c_2 + \underline{c}_1}{\underline{c}_1 + \underline{c}_2}\right)\right) \quad \text{(PS)}
\end{aligned} \tag{6}$$

where PSE stands for *exceedance possibility* and PS for *strict exceedance possibility*. Acording to [3], these formulas hold except when the sums of the spreads in the denominators are zero, which occurs when $\widetilde{C}_1$ and $\widetilde{C}_2$ are ordinary numbers.

So we assume that $\widetilde{C}_1 > \widetilde{C}_2$ when:

$$Pos(\widetilde{C}_1 \geq \widetilde{C}_2) \geq \alpha, \quad \alpha \in (0, 1] \tag{7}$$

and

$$\min\left[Pos(\widetilde{C}_1 > \widetilde{C}_2), Pos(\widetilde{C}_2 > \widetilde{C}_1)\right] < 1 \tag{8}$$

Condition (8) guarantees that $\widetilde{C}_1 \neq \widetilde{C}_2$.

The topics presented in this section were directly used in the computational implementation, so their descriptions are brief and without details about the theorethical foundations.

3. THE PROBLEM

As described in [5], a fuzzy function is used when some data about the problem is not precisely known. Here a function with fuzzy parameters can be denoted by:

$$\begin{aligned}
\min \quad & f(\widetilde{c}, x) \\
& x_i \in [a_i, b_i] \quad i = 1:n \\
& x \in R^n
\end{aligned} \tag{9}$$

where $x \in R$ and $\widetilde{c}$ is a vector whose entries are fuzzy numbers, such that $\widetilde{c} \in F(R)$, $F(R)$ is a fuzzy set over R and $f(\widetilde{c}, x) : [F(R) \times R^n] \to F(R)$. The interval $[a, b]$ is the region which the minimum value of the function, namely $\overline{x}$, occurs.

Even though problem (9) is an irrestrict on, we determine a search region for vector x where the nonlinear function is evaluated.

4. CONTINUOUS ANT COLONY SYSTEM (CACS) WITH FUZZY PARAMETERS

The heuristic method developed by [8] is a modification over the heuristic *Ant Colony System* — ACS, preserving all of its major characteristics. Some important aspects are related here.

4.1. Continuous Pheromone Model

As reported in [8] and [7], the pheromone deposition occurs over a continuous space. For fuzzy problem (9), this step involve only ordinary numbers, because it concerns only information about the vector x.

Consider a food source surrounded by several ants. The ant's aggregation around the food source causes the highest pheromone intensity to occur at the food source position. Then, increasing the distance of a sample point from the food source will decrease its pheromone concentration. This model uses a Probability Distribution Function (PDF), which determines the probability of choosing each point x within the interval $[a, b]$.

The normal PDF can be used at the state transition rule since the center of which is the last best global solution and its variance depends on the aggregation of the promising areas around the best one, so it contains exploitation behavior. In the other hand, a normal PDF permits all points of the search space to be chosen, either close to or far from the current solution, so it also contains exploration behavior.

4.2. Pheromone Update

At the start of the algorithm, there is no information available about the minimum point and the ants chose their destination only by exploration.

During each iteration, pheromone distribution over the search space will be updated using the acquired knowledge of the evaluated points by the ants. This process gradually increases the exploitation behavior of the algorithm, while its exploration behavior will decrease, i. e, the value of objective function is evaluated for the new selected points by the ants. Then, the best point found from the beginning of the trial is assigned to $x_{\min}$. Also the value of σ is update based on the evaluated points during the last iteration and the aggregation of those points around $x_{\min}$. Then a concept of weighted variance is defined as follows:

$$\sigma^2 = \frac{\sum_{j=1}^{k}\left(\frac{1}{D(f_j) - D(f_{\min})}\left(x_j - x_{\min}\right)\right)}{\sum_{j=1}^{k}\left(\frac{1}{D(f_j) - D(f_{\min})}\right)}, \tag{10}$$

for all j in which $D(f_j) \neq D(f_{\min})$, $D(\cdot)$ meaning the defuzzification presented in equation (4) and k is the number of ants. This strategy means that the center of the region discovered during the subsequent iterations is the last best point and the narrowness of its width is dependent on the aggregation of the other competitors around the best one. The closer the better solutions go to the best one, the smaller σ is assigned to the next iteration.

4.3. Algorithm

The algorithm description, based in [7], is shown below, including the modification for the fuzzy problem (9).

Step 1 choose randomly the initially guessed minimum point $\overline{x}_{\min}$ over the space and calculate the value of the function $f(\widetilde{c}, \overline{x}) = f_{\min}$, calcule $D(f_{\min})$ using (4). For each x_i use a uniform PDF over the interval $[a_i, b_i]$.

Step 2 Set the inicial value of weighted variance for each pheromone intensity distribution function: $\sigma_i = 3(b_i - a_i)$, $i = 1:n$. It will be large enough to approximately generate uniformly distributed initial values of x_i within the interval $[a_i, b_i]$.

Step 3 Send ants to points $(x_1, x_2, \ldots, x_n)_j$, $j = 1:k$. To generate these random locations, a random generator with normal PDF is utilized for each x_i, where its mean and variance are

$(x_j)_{\min}$ and σ_i respectively. If x_i is outside the given interval $[a_i, b_i]$, it is discarded and a new x_i is generated again.

Step 4 Evaluate f, at each discovered point, namely $f_1, f_2, \ldots, f_k$. Determine the minimum f_m and compare these value using (7) and (8) with the current minimum value $f_{\min}$ and determine the updated $f_{\min}$ and its associated $(x_1, x_2, \ldots, x_n)_{\min}$.

Step 5 If a stopping criterion is satisfied (usually the number of iterations) then **stop**, else update the weighted variance parameter σ_i for each variable x_i using (10); go back to **step 3**).

5. EXPERIMENTS

The following experiments employ test functions from [8] and one function from [10]. Also, each function will be presented on its own table, together the original result from their respective references.

All fuzzy numbers have 10% of uncertainty. They are represented as $[c, c-\overline{c}, c+\underline{c}]$, where c is the modal value of the number ($\mu_{\widetilde{c}}(x) = 1$), $c-\overline{c}$ and $c+\underline{c}$ are the inferior and superior values for the number $\mu_{\widetilde{c}}(x) = 0$, respectively.

Each table is composed by four columns, with the first one being the number of ants, followed by the best crisp result obtained on the cited reference, then the fuzzy objective function value and finally, on the last column, the defuzzified objective function number. The table was constructed upon an implementation using Scilab (`www.scilab.org`) version 5.3.0.

5.1. Function 1

$f_1(\widetilde{c}, x) = \widetilde{3905.93} - \widetilde{10}(x_1^2 - x_2)^2 - (1-x_1)^2$, $-2.048 \leq x_1, x_2 \leq 2.048$.

k	[10]	f_1	$D(f_1)$
100	−3905.93	[−3905.93, −3944.99, −3866, 87]	−3905.93
200		[−3905.93, −3944.99, −3866, 87]	−3905.93
500		[−3905.93, −3944.99, −3866, 87]	−3905.93

Table 1: Results for function f_1.

Note that, in [10] don't have information about the number of ants used in the tests, only the optimum value.

5.2. Function 2

$f_2(\widetilde{c}, x) = \widetilde{100}(x_1^2 - x_2)^2 + (1-x_1)^2$ and $-2.05 \leq x_1, x_2 \leq 2.05$.

k	[8]	f_2	$D(f_2)$
100	$1.6e-33$	$[1.9036e-19, -0.01, 0.01]$	0
200	$3.2e-22$	$[2.743e-21, -0.01, 0.01]$	0
500	$1.7e-12$	$[2.299e-20, -0.01, 0.01]$	0

Table 2: Results for function f_2.

5.3. Function 3

$f_3(\widetilde{c}, x) = \widetilde{1}x_1^2 + \widetilde{1}x_2^2 + \widetilde{1}x_3^2$, $-5.12 \leq x_1, x_2, x_3 \leq 5.12$.

k	[8]	f_3	$D(f_3)$
100	$3.6e-37$	$[1.394e-34, 1.255e-34, 1.534e-34]$	$1.394D-34$
200	$1.5e-20$	$[9.094D-36, 8.184e-36, 1.000-35]$	$9.094e-36$
500	$3.0e-09$	$[3.761e-35, 3.385e-35, 4.137e-35]$	$3.761e-35$

Table 3: Results for function f_3.

5.4. Function 4

$f_4(\widetilde{c}, x) = \widetilde{0.5} + \frac{\sin^2(x_1^2+x_2^2)^{1/2}}{\widetilde{1}+\widetilde{0.001}(x_1^2+x_2^2)}$, $-100 \leq x_1, x_2 \leq 100$.

k	[8]	f_4	$D(f_4)$
100	$7.8e-3$	$[0, -0.1611111, 0.1409091]$	−0.0050564
200	$7.7e-3$	$[0, -0.1611111, 0.1409091]$	−0.0050564
500	$1.4e-2$	$[0, -0.1611111, 0.1409091]$	−0.0050564

Table 4: Results for function f_4.

5.5. Function 5

$f_5(\widetilde{c}, x) = \widetilde{50} + \sum_{i=1}^{5}\left(x_i^2 - \widetilde{10}\cos(\widetilde{2\pi}x_i)\right)$, $-5.12 \leq x_i \leq 5.12$.

k	[8]	f_5	$D(f_5)$
100	4.9	$[0, -10, 10]$	$1.108e-14$
200	7.1	$[0, -10, 10]$	$1.108e-14$
500	9.4	$[0, -10, 10]$	$1.108e-14$

Table 5: Results for function f_5.

5.6. Function 6

$f_6(\widetilde{c}, x) = \widetilde{1} + \sum_{i=1}^{2}\frac{x_i^2}{\widetilde{4000}} - \prod_{i=1}^{2}\cos\left(\frac{x_i}{\sqrt{i}}\right)$, $-5.12 \leq x_i \leq 5.12$.

The number $\widetilde{4000}$ has only 1% of fuzzy uncertainty.

k	[8]	f_6	$D(f_6)$
100	$4.1e-3$	$[-9.375e-8, -0.1000001, 0.0999999]$	$-9.376e-8$
200	$2.7e-3$	$[-9.375e-8, -0.1000001, 0.0999999]$	$-9.376e-8$
500	$1.1e-3$	$[-9.375e-8, -0.1000001, 0.0999999]$	$-9.376e-8$

Table 6: Results for function f_6.

5.7. Function 7

$f_7(\widetilde{c}, x) = \widetilde{1} + \sum_{i=1}^{5}\frac{x_i^2}{\widetilde{4000}} - \prod_{i=1}^{5}\cos\left(\frac{x_i}{\sqrt{i}}\right)$, $-5.12 \leq x_i \leq 5.12$.

The number $\widetilde{4000}$ has only 1% of fuzzy uncertainty.

k	[8]	f_7	$D(f_7)$
100	$7.8e-3$	$[-9.375e-8, -0.1000001, 0.0999999]$	$-9.376e-8$
200	$7.7e-3$	$[-9.375e-8, -0.1000001, 0.0999999]$	$-9.376e-8$
500	$1.4e-2$	$[-9.375e-8, -0.1000001, 0.0999999]$	$-9.376e-8$

Table 7: Results for function f_7.

5.8. Function 8

$f_8(\widetilde{c}, x) = (x_1^2 + x_2^2)^{0.25}\left(\widetilde{1} + \sin^2\left(\widetilde{50}(x_1^2+x_2^2)^{0.1}\right)\right)$, $-100 \leq x_i \leq 100$.

k	[8]	f_8	$D(f_8)$
100	$2.5e-3$	$[2.176e-02, 1.935e-02, 2.176e-02]$	0.0176
200	$5.9e-2$	$[3.509e-03, 1.815e-03, 2.219e-03]$	0.00202
500	$3.8e-1$	$[4.228e-02, 1.945e-02, 2.377e-02]$	0.02168

Table 8: Results for function f_8.

5.9. Function 9

$f_9(\widetilde{c}, x) = -\widetilde{20}\exp\left(-\widetilde{0.2}\sqrt{\frac{1}{n}\sum_{i=1}^{30}x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{30}\cos(\widetilde{2\pi}x_i)\right) + \widetilde{20} + \widetilde{e}$, $-32 \leq x_i \leq 32$.

k	f_9	$D(f_9)$
100	$[1.421e-14, -4.27, 4.27]$	$5.547e-17$
200	$[1.421e-14, -4.27, 4.27]$	$5.547e-17$
500	$[1.421e-14, -4.27, 4.27]$	$5.547e-17$

Table 9: Results for function f_9. Optimum crisp solution $f_9(\mathbf{0}) = 0$

.

6. CONCLUSIONS

Fuzzy theory was proposed by L. A. Zadeh in 1965, as a tool to help the quantification of inherent imprecisions on the subject being studied. This theory quickly spreaded to several different fields, ranging from Engineering, Informatics and Mathematics to medical diagnosis, plague control e so on.

Mathematical Programming is by itself a very important decision tool in several application areas. Aggregation of Fuzzy characteristics to it allows the modelling of uncertainties when the avaliable data is not exactly know or has inherent inaccuracies, making Mathematical Programming an even more powerful resource.

In this sense, the quest for optimization methods that succesfully embraces Fuzzy Theory has been the focus of this work. Here, we have introduced uncertainty in the coefficients of the objective function. These uncertainties were modelled by fuzzy numbers. Its application on the Continuous Ant Colony System heuristics required some adaptations, as outlined here.

We tested 9 functions from the literature, with 77% of them giving equivalent or better results when compared with their crisp counterparts. Even the 23% of the test cases that performed worse, were kept inside de viable solution space and also relatively close to the crisp solution.

So, despite the fact that this implementation still needs some improvements — for instance, incorporating different ant colony heuristics, as proposed in [9] and [12] — its results are compatible with the ordinary ones, allowing a flexibilization in the modelling of real case, where crisp Mathematical Programming is not directly applicable.

7. ACKNOWLEDGEMENTS

To FUNDUNESP for the financial support and to Dr. Renato Fernandes Cantão for some invaluable hints.

8. REFERENCES

[1] M. Dorigo and L. M. Gambardella. Ant colony system: a cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, vol. 1(1), 53–66, 1997.

[2] M. Dorigo, V. Maniezzo, A. Colorni. Ant system: optimization by a colony of cooperative agents. *IEEE Transaction on Systems, Man and Cybernetics*, vol. 26(1), 29–41, 1996.

[3] D. Dubois and H. Prade. *Possibility theory: an approach to computerized processing of uncertainty*. Plenum Press, 1986.

[4] D. Dubois and H. Prade. Ranking fuzzy numbers in the setting of possibility theory. *Information Science*, vol. 30, 183-224, 1983.

[5] K. D. Jamison and W. A. Lodwick. Minimizing constrained fuzzy functions. *Fuzzy Sets and Systems*, vol. 103, 457-464, 1999.

[6] A. Kaufmann and M. M. Gupta. *Introduction to fuzzy arithmetic*. Van Nostrand Reinhold, 1991.

[7] H. Nobahari and S. H. Pourtakdoust. Optimization of fuzzy rule bases using continous ant colony system. *Proceeding of the First International Conference on Modeling, Simulation and Applied Optimization – ICMSAO*, 2005.

[8] S. H. Pourtakdoust and H. Nobahari. An extension of ant colony system to continous optimization problems. In *Proc. ANTS*, vol. 3172, *LNCS*, M. Dorigo, M. Birattari, C. Blum, L. M. Gambardella, F. Mondata and T. Sützle, Eds., 294-301, 2004.

[9] K. Socha and M. Dorigo. Ant colony optimization for continuous domains. *European Journal of Operational Research*, vol. 185(3), 1155-1173, 2008.

[10] A. de Vicente. O processo de otimização Ant System com redução no raio de busca. *TEMA Tend. Mat. Apl. Comput.*, vol. 7(1), 159-168, 2006 (*in portuguese*).

[11] Y.-M. Wang. Centroid defuzzification and the maximizing set and minimizing set ranking based on alpha level sets. *Computers & Industrial Engineering*, Vol. 57, 228-236, 2009.

[12] X.-M. Hu, J. Zhang, H. S.-Hung, Y. Li and O. Liu. SamACO: variable sampling ant colony optmization algorithm for continuous optimization. *IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics*, vol. 40(6), 1555-1566, 2010.

[13] X.-M. Hu, J. Zhang and Y. Li. Orthogonal methods based ant colony search for solving continuous optimization problems. *Journal of Computer Science and Technology*, vol. 23(1), 2-18, 2008.

A tree search procedure for forest harvest scheduling problems addressing aspects of habitat availability

Teresa Neto [*] Miguel Constantino [†] João Pedro Pedroso [‡] Isabel Martins [§]

[*] Escola Superior de Tecnologia de Viseu do Instituto Politécnico de Viseu
3504-510 Viseu, Portugal
tneto@estv.ipv.pt

[†] Centro de Investigação Operacional e Faculdade de Ciências da Universidade de Lisboa
Cidade Universitária, 1749-016 Lisboa, Portugal
miguel.constantino@fc.ul.pt

[‡] INESC Porto e Faculdade de Ciências da Universidade do Porto
Rua do Campo Alegre, 4169-007 Porto, Portugal
jpp@fc.up.pt

[§] Centro de Investigação Operacional e Instituto Superior de Agronomia da Universidade Técnica de Lisboa
Tapada da Ajuda, 1349-017 Lisboa, Portugal
isabelinha@isa.utl.pt

ABSTRACT

In the literature, the most referenced approaches for forest harvesting scheduling problems addressing environmental protection issues have focused mainly on including constraints on clearcut area. Nevertheless, these restrictions may not be sufficient to prevent the loss of habitat availability that endangers the survival of many wild species. This work presents a tree search procedure for finding good feasible solutions, in reasonable time, to forest harvest scheduling problems with constraints on clearcut area and habitat availability. We use two measures for habitat availability: the area of all habitats and the connectivity between them. For solving the problem, we use a tree search procedure: a process inspired in branch-and-bound, specifically designed for this problem. In each branch, a partial solution leads to two children nodes, corresponding to harvesting or not a given stand in a given period. Pruning is based on constraint violations or on unreachable objective values. Preliminary computational results are reported.

Keywords: Forest management, Harvest scheduling, Habitat availability, Tree search

1. INTRODUCTION

Forest management problems for timber production have been addressing concerns with resources other than timber, such as wildlife, soil, water and aesthetics values. Modeling approaches to confront these concerns have mainly involved the use of restrictions on the maximum clearcut area. However, the solution generated by these approaches typically has a dispersion of smaller clearcuts across the forest; it is known that forest fragmentation may have significant negative impacts on some wildlife species. Indeed, forest fragmentation generally implies a reduction of habitat availability that is, the total area of habitats (mature patches meeting a minimum target area or with an usable interior or core space with minimum area requirements) and the connections between them [1, 2]. Core area of a mature patch is determined by its size and shape and immediate surrounding conditions. Some animal species are more dependent on core area than total area of mature patches [3]. Connectivity between habitats enables wildlife movement through the forest, thus enhancing the probability of survival. It is considered a key issue for the biodiversity conservation and for the maintenance of natural ecosystems stability and integrity [4].

There are several works in forest planning that include mature patch size requirements, using exact integer programming approaches [5, 6, 7, 8, 9, 10, 11] or heuristic methods [12, 13, 14, 15, 16, 17, 18, 19, 20, 21]. To date, as far as we know, no method for forest harvest scheduling problems explicitly addressing the inter-habitat connectivity issue has been reported.

When full search is possible in reasonable time exact solution take advantage over heuristics, as they determine proved optimal solutions. When the problems are too large to be solved exactly, exact methods may be interrupted in the middle of the search. Tree search can used as an exact method, especially to solve academic problems [22, 23], but it also can be used as a heuristic [24].

This work presents a tree search approach for finding good feasible solutions, in reasonable time, to forest harvest scheduling problems with constraints on clearcut area and habitat availability. Every mature patch meeting a minimum target area is considered a habitat (*i.e.* core area is not considered). Several connectivity indices have been proposed for landscape conservation planning; we use the *probability of connectivity* index proposed by [25].

We report computational tests involving both real forests and generated benchmark instances.

2. PROBLEM

Basic forest harvest scheduling problems generally encompass the maximization of the net present value of timber harvested within a temporal horizon, subject to several non-spatial constraints. In this work, we consider lower and upper bounds on the volume of timber harvested in each period (constraints R_1^l and R_1^u, respectively) and a minimum average age for the forest at the end of the planning horizon (constraints R_2). Constraints on clearcut area and habitat availability are considered. Constraints R_3 impose a maximum in the area of each clearcut; constraints R_4 concern the minimum number of periods in which stands adjacent to a clearcut can not

be harvested, the so-called *greenup* restrictions. Constraints on habitat availability impose, in each period, a minimum in the total area of habitats (constraints R_5) and a minimum value for the probability of connectivity (constraints R_6). It is assumed that a stand may be harvested only once, and that a harvested stand may become mature within the time horizon.

To identify clearcuts or mature patches (maximal groups of contiguous stands) it is necessary to define adjacency between stands. For clearcuts, we consider that two stands are adjacent if they share a boundary that is not a discrete set of points (*strong* adjacency). For mature patches, we consider that it is sufficient to share at least a single point (*weak* adjacency).

Many indices have been used for connectivity analysis [26, 27, 28, 29, 30, 25]. The authors in [25] encourage the use of the probability of connectivity, an index that is based on the availability concept, dispersal probabilities between habitats and graph structures. This index uses an indicator p_{hr} of the possibility of a direct movement occurrence (without passing by any other intermediate habitat) between habitats h and r, obtained by a negative exponential function:

$$p_{hr} = e^{-Cd_{hr}},$$

where C is a constant greater than zero called the coefficient of dispersion (species dependent), and d_{hr} is the edge-to-edge distance between h and r (in km). This indicator expresses the possibility of an animal to disperse among habitats. The closer the indicator is to 1, the smaller is the inter-habitat distance, and the more favorable is the occurrence of a movement. In this work, the distance between two stands (represented as polygons) is simply computed as the minimum Euclidean distance between their vertices; the edge-to-edge distance between two mature habitats is approximated by the minimum distance between their stands.

A path between two habitats h and r, $h \neq r$, is made up of a sequence of direct movements from h to r in which no habitat is visited more than once. The *connectivity of a path* is given by the product of the indicators of direct movements that form the path. The largest connectivity among all paths between h to r is denoted by g_{hr}, and indicates the path with greatest chance of dispersion.

Let $\mathscr{H}_t$ be the set of all habitats in period t, s_h be the area of habitat $h, \forall h \in \mathscr{H}_t$ and H_t be the total habitat area. The probability of connectivity for period t is given by:

$$I_t = \frac{\sum_{h \in \mathscr{H}_t} \sum_{r \in \mathscr{H}_t} s_h s_r g_{hr}}{H_t^2}. \quad (1)$$

I_t expresses the possibility of two animal randomly placed into two habitats to fall into interconnected habitats. I_t ranges from 0 to 1, and increases with improving connectivity. It is equal to 1 when the forest is composed by a single habitat, and is equal to zero when there are no habitats, or all habitats are completely isolated (by being too distant).

3. TREE SEARCH

The tree search proposed in this work is inspired in a branch-and-bound designed specifically for this problem. The procedure consists of successive branching on partial solutions; more specifically, in each branch a partial solution can lead to two children solutions, corresponding to the decision of harvesting or not a stand in a given period.

Let T be the number of periods within the time horizon and n be the number of stands. The first step is to initialize a queue Q with the tree's root node, defined by the following elements:

- $\mathscr{S}_0$, the set of all pairs $(\text{stand } i, \text{period } t)$ such that i is available to be harvested in t, sorted by descending order of the net present value corresponding to i and t;
- a solution x^0 where no decision is taken ($x_i = T+1$ for all stands i);
- the net present value of x^0, $fnpv(x^0) = 0$;
- an upper bound ub_0 to the net present value of an optimal solution to the problem.

The maximum cardinality of $\mathscr{S}_0$ is $n \times T$, which happens when all stands are old enough to be harvested in any period.

At each tree node k, the first element (i_k, t_k) of $\mathscr{S}_k$ is selected. The partial solution x^k leads to two new partial solutions, corresponding to the decision of harvesting or not harvesting stand i_k in period t_k (left and right branches, respectively):

- x^{k+1}, where we fix $x^{k+1}_{i_k} = t_k$ and $x^{k+1}_i = x^k_i$, for all $i \neq i_k$;
- x^{k+2}, with $x^{k+2}_i = x^k_i$ for all i.

The sets corresponding to the two new branches are $\mathscr{S}_{k+1}$ and $\mathscr{S}_{k+2}$, initialized by removing (i_k, t_k) from $\mathscr{S}_k$. The set $\mathscr{S}_{k+1}$ is updated by removing any pair (i,t) such that harvesting stand i in period t violates the following restrictions: R_1^u; R_2; R_3 and R_4 if stands i and i_k are adjacent.

At any node k', restrictions R_1^l can only be fully checked when $\mathscr{S}_{k'}$ is empty (all the decisions were taken). In this case, if the corresponding solution $x^{k'}$ does not satisfy constraints R_1^l, k' is infeasible, otherwise k' is feasible. However, when $\mathscr{S}_{k'}$ is not empty, we check for period $t_{k'}$, and the next green-up periods, whether harvesting all stands still available gives a volume of timber greater or equal than the lower bound (infeasibility test). If not, node k' is infeasible, as can not lead to solutions meeting R_1^l. Otherwise, no conclusion is drawn about the infeasibility of k'.

We check nodes $k+1$ and $k+2$ with the infeasibilty test. If we do not conclude that node $k+1$ is infeasible, more updates are made: $fnpv(x^{k+1})$ is equal to $fnpv(x^k)$ plus the net present value of stand i_k in period t_k, and the upper-bound ub_{k+1} is calculated. If no conclusion is drawn about the infeasibility of $k+2$, the upper-bound ub_{k+2} is calculated. Any upper bound is on the optimal net present value of the forest harvest scheduling problem addressing aspects of habitat availability where the decisions already taken are incorporated.

Any node k' can be pruned by one of the following three reasons:

- k' is infeasible (either $\mathscr{S}_{k'}$ is empty or not);
- $\mathscr{S}_{k'}$ is empty and $x^{k'}$ is feasible;
- the upper-bound $ub_{k'}$ at node k' is not greater than the best net present value found so far.

The new (non-pruned) nodes are inserted into queue Q and the process continues from these elements. Tree search ends when Q is empty, or a certain CPU time limit is reached.

In this work, several types of upper-bounds are tested.

The method can be represented by a tree, as shown in figure 1. The tree has a maximum height of $n \times T + 1$ and a maximum number of nodes of $2^{(n \times T+1)} - 1$.

4. TREE SEARCH IMPLEMENTATION

Three strategies to guide the search on the tree were implemented: depth-first, best-first and beam search.

In depth-first search (DFS), the search descends on the tree until a leaf (pruned node) is reached. This is implemented though a

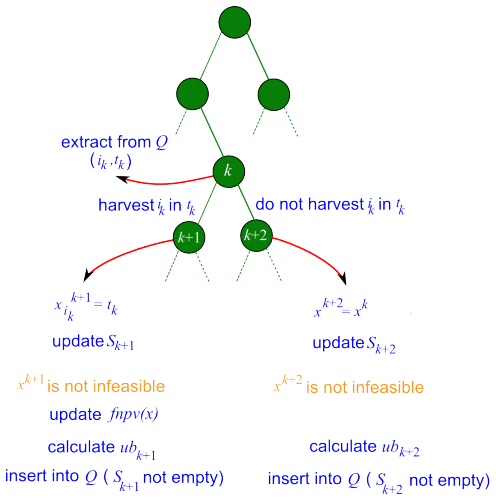


Figure 1: *Tree Search.*

last-in-first-out ($LIFO$) process on the queue Q. The right branch solution is inserted into Q first, followed by the left branch solution.

In best-first search (BFS), branch is made on the Q element that has the highest upper-bound.

In breadth-first search, all the solutions at the same level are searched before exploring the next level. In beam search (BS), breadth-first search is parameterized, by limiting the number of solutions to branch per level. At each level, the generated partial solutions are sorted by ascending order of the upper-bound, and the β last solutions are branched; the other solutions are pruned. This strategy reduces the memory requirements of breadth-first search.

On the first two strategies, when Q is empty the whole tree has been explored; in these cases, the best feasible solution is an optimal solution. Tree search is used as a heuristic when only a part of the tree is explored.

5. PRELIMINARY RESULTS

We report results for WLC and El Dorado instances (also available at the web site www.unbf.ca/fmos/), with 73 and 1363 stands, respectively. The El Dorado forest in the U.S.A. El Dorado is referred to in [31]. The test problem runs were made on a desktop computer with an Intel Core 2 - 2 GHz processor and 2 GB RAM. Tree search was implemented with Python language.

Different values are used for the minimum value of the probability of connectivity. The DFS and BFS strategies were allowed to run for two hours at most. The results show that the strategies with the different types of upper bounds were able to give feasible solutions for the instances. BS largely depends on the value of the parameter β.

6. REFERENCES

[1] L. D. Harris, "The fragmented forest: Island biogeography theory and the preservation of biotic diversity," University of Chicago, Chicago, Tech. Rep., 1984.

[2] M. Kurtilla, T. Pukkala, and J. Loikkanen, "The performance of alternative spatial objective types in forest planning calculations: a case for flying squirrel and moose," *Forest Ecology and Management*, vol. 166, pp. 245–260, 2002.

[3] E. Z. Baskent and G. A. Jordan, "Characterizing spatial structure of forest landscapes," *Canadian Journal of Forest Research*, vol. 25, no. 11, pp. 1830–1849, 1995.

[4] P. Taylor, L. Fahrig, K. Henein, and G. Merriam, "Connectivity is a vital element of landscape structure," *Oikos*, vol. 68, no. 3, pp. 571–573, 1993.

[5] J. Hof, M. Bevers, L. Joyce, and B. Kent, "An integer programming approach for spatially and temporally optimizing wildlife populations," *Forest Science*, vol. 40, no. 1, pp. 177–191, 1994.

[6] I. Martins, M. Constantino, and J. G. Borges, "Forest management models with spatial structure constraints," C.I.O./Faculdade de CiÍncias de Lisboa, Working Paper no. 2/1999, 1999.

[7] S. Rebain and M. E. McDill, "A mixed-integer formulation of the minimum patch size problem," *Forest Science*, vol. 49, no. 4, pp. 608–618, 2003.

[8] S. Rebain and M. McDill, "Can mature patch constraints mitigate the fragmenting effect of harvest opening size restrictions?" *Int. Trans. Operations Research*, vol. 10, no. 5, pp. 499–513, 2003.

[9] S. F. Tóth, M. E. McDill, and S. Rebain, "Finding the efficient of a bi-criteria, spatially-explicit, harvest scheduling problem," *Forest Science*, vol. 52, no. 1, pp. 93–107, 2006.

[10] Y. Wei and H. M. Hoganson, "Scheduling forest core area production using mixed integer programming," *Canadian Journal of Forest Research*, vol. 37, no. 10, pp. 1924–1932, 2007.

[11] K. Öhman and P. Wikström, "Incorporating aspects of habitat fragmentation into long-term forest planning using mixed integer programming," *Forest Ecology and Management*, vol. 255, pp. 440–446, 2008.

[12] K. Öhman and L. Eriksson, "The core area concept in forming contiguous areas for long term forest planning," *Canadian Journal of Forest Research*, vol. 28, no. 7, pp. 1032–1039, 1998.

[13] K. Öhman, "Creating continuous areas of old forest in long term forest planning," *Canadian Journal of Forest Research*, vol. 30, no. 11, pp. 1817–1823, 2000.

[14] A. Falcao and J. Borges, "Combining random and systematic search heuristic procedures for solving spatially constrained forest management scheduling models," *Forest Science*, vol. 48, no. 3, pp. 608–621, 2002.

[15] K. Öhman, *Multi-objective forest planning*. Kluwer Academic Publishers, 2002, ch. Spatial optimisation in forest planning: a review of recent Swedish research, pp. 153–172.

[16] F. Caro, M. Constantino, I. Martins, and A. Weintraub, "A 2-opt tabu search procedure for the multiperiod forest harvesting problem with adjacency, greenup, old growth, and even flow constraints," *Forest Science*, vol. 49, no. 5, pp. 738–751, 2003.

[17] H. M. Hoganson, Y. Wei, and R. H. Hokans, "Integrating spatial objectives into forest plans for minnesota national forests," in *Systems analysis in forest resources October 7-9 Stevenson WA, Proceedings of the 2003 Symposium*, M. Bevers and T. Barrett, Eds., USDA Forest Service - Rocky Mountain Research Station, 2004.

[18] I. Martins, M. Constantino, and J. G. Borges, "A column generation approach for solving a non-temporal forest harvest model with spatial structure constraints," *European Journal of Operational Research*, vol. 161, no. 2, pp. 478–498, 2005.

[19] K. Öhman and T. Lämås, "Reducing forest fragmentation in long-term forest planning by using the shape index," *Forest Ecology and Management*, vol. 212, pp. 346–357, 2005.

[20] A. H. Mathey, E. Kremar, and I. Vertinsky, "Re-evaluating our approach to forest management planning: a complex journey," *Forestry Chronicle*, vol. 81, no. 3, pp. 359–364, 2005.

[21] Y. Wei and H. M. Hoganson, "Tests of a dynamic programming-based heuristic for scheduling forest core area production over large landscapes," *Forest Science*, vol. 54, no. 3, pp. 367–380, 2008.

[22] A. Sbihi, "A best first search exact algorithm for the multiple-choice multidimensional knapsack problem," *Journal of Combinatorial Optimization*, vol. 13, no. 4, pp. 337–351, 2007.

[23] C. Artigues, M. Gendreau, L. M. Rousseau, and A. Vergnaud, "Solving an integrated employee timetabling and job-shop scheduling problem via hybrid branch-and-bound," *Computers & Operations Research*, vol. 36, no. 8, pp. 2330–2340, 2009.

[24] J. P. Pedroso and M. Kubo, "Heuristics and exact methods for number partitioning," *European Journal of Operational Research*, vol. 202, pp. 73–81, 2010.

[25] S. Saura and L. Pascual-Hortal, "A new habitat availability index to integrate connectivity in landscape conservation planning: comparison with existing indices and application to a case study," *Landscape and Urban Planning*, vol. 83, pp. 91–103, 2007.

[26] "Fragstats: spatial pattern analysis program for categorical maps," 1995. [Online]. Available: http//www.umass.edu/landeco/research/fragstats/fragstats.html

[27] N. H. Shumaker, "Using landscape indices ti predict habitat connectivity," *Ecology*, vol. 77, no. 4, pp. 1210–1225, 1996.

[28] T. H. Keitt, D. L. Urban, and B. Milne, "Detecting critical scales in fragmented landscapes," *Conservation Ecology*, vol. 1, no. 1, 1997. [Online]. Available: http//www.consecol.org/Journal/vol1/iss1/art4.

[29] A. G. Bunn, D. L. Urban, and T. H. Keitt, "Landscape connecivity: a conservation application of graph theory," *Journal of Environmental Management*, vol. 2, no. 10, pp. 265–278, 2000.

[30] S. S. L.P. Hortal, "Comparison and development of new graph-based connectivity indices: towards the priorization of habitat patches and corridors for conservation," *Landscape Ecology*, vol. 21, no. 7, pp. 959–967, 2006.

[31] M. Goycoolea, A. T. Murray, F. Barahona, R. Epstein, and A. Weintraub, "Harvest scheduling subject to maximum area restrictions: exploring exact approaches," *Operations Research*, vol. 53, no. 3, pp. 490–500, 2002.

Automatic Configuration of TPLS+PLS Algorithms for Bi-objective Flow-Shop Scheduling Problems

Jérémie Dubois-Lacoste * Manuel López-Ibáñez * Thomas Stützle *

* IRIDIA-CoDE, Université Libre de Bruxelles
Brussels, Belgium
{dubois-lacoste, manuel.lopez-ibanez, stuetzle}@ulb.ac.be

ABSTRACT

The automatic configuration of algorithms is a hot research topic nowadays, and it is rapidly having an increasing impact on the way algorithms are designed and evaluated. The main focus of automatic configuration tools has been so far the configuration of single-objective algorithms. However, these tools may be applied to the automatic configuration of multi-objective algorithms for Pareto-optimization by means of unary quality measures such as the hypervolume. This study shows that such an approach is able to outperform state-of-the-art multi-objective optimizers that were manually configured. The results presented here on five variants of multi-objective flow-shop problems show that the automatically configured algorithm reaches at least the same and often better final quality than the current state-of-the-art algorithm.

Keywords: Automatic configuration, Multi-objective, Flow-shop scheduling

1. INTRODUCTION

This paper presents a study of automatic algorithm configuration tools for improving the performance of multi-objective algorithms. Very recently, López-Ibáñez and Stützle [1] applied automatic configuration techniques to the configuration of multi-objective algorithms. In particular, they automatically configured a multi-objective ant colony optimization (MACO) framework, leading to new MOACO algorithms that outperform previously proposed MOACO algorithms for the bi-objective traveling salesman problem (bTSP). Despite the inherent interest for the research on MOACO algorithms, the results obtained by the new MOACO algorithms are still behind state-of-the-art algorithms for the bTSP. In this study, our aim is to configure in an automatic fashion a new state-of-the-art multi-objective optimizer for an $\mathscr{NP}$-hard problem. In particular, we tackle five bi-objective variants of the multi-objective flow-shop problem. The current state-of-the-art algorithm for these five bi-objective permutation flow-shop problems (bPFSPs) was already shown to outperform by a substantial margin all previously available algorithms for these problems [2] and, hence, we expected little room for improvement. Nonetheless, the results reported here show that automatic configuration leads to a significant improvement over the current state-of-the-art algorithm.

The current state-of-the-art algorithm for these five bPFSPs is a hybrid algorithm combining the two-phase local search (TPLS) [3] and the Pareto local search (PLS) frameworks [4]. TPLS tackles multi-objective problems by using efficient single-objective algorithms to solve a sequence of scalarizations (weighted sum aggregations) of the multi-objective problem. PLS is a local search method for multi-objective problems that uses the Pareto dominance criterion as an acceptance criterion in the local search. From these two frameworks, we have build a hybrid TP+PLS software framework.

The flow-shop scheduling problem (FSP) [5] is one of the most widely studied scheduling problems. In this work we study the bi-objective variants that arise from the minimisation of the following objectives: the makespan ($C_{\max}$, that is, the completion time of the last job), sum of flowtimes (*SFT*, that is the sum of the completion times of all jobs), weighted tardiness (*WT*, that is, the sum of the amount of time a job is late weighted by each job's priority) and the total tardiness (*TT*, that is the same as *WT* but all priorities are equal). We tackle the bi-objective PFSPs that result from five possible pairings of objectives (we do not consider the combination of the total and weighted tardiness): ($C_{\max}$, *SFT*), ($C_{\max}$, *TT*), ($C_{\max}$, *WT*), (*SFT*, *TT*) and (*SFT*, *WT*). These bi-objective problems have been the focus of intensive research, which is summarised in a recent review [6].

In bi-objective combinatorial optimization problems, candidate solutions are evaluated according to an *objective function vector* $\vec{f} = (f_1, f_2)$. Given two vectors $\vec{u}, \vec{v} \in \mathbb{R}^2$, we say that $\vec{u}$ *dominates* $\vec{v}$ ($\vec{u} \prec \vec{v}$) iff $\vec{u} \neq \vec{v}$ and $u_i \leq v_i,\ i = 1,2$. Without preference information about the objectives, the aim is, without loss of generality, to minimize the objective functions in terms of Pareto-optimality, that is, to find the set of solutions that are not dominated by any other feasible solution. This set is called the Pareto set, and its image in the objective space is called the Pareto front. Since this goal is in many cases intractable, the goal becomes to find a set of non-dominated solutions that approximates well the Pareto front.

The assessment of the relative quality of different Pareto front approximations is a difficult problem, since they are often incomparable in the Pareto sense. For this purpose, several unary quality indicators have been proposed that try to summarise the quality of a Pareto front approximation into a single scalar value. In this paper, we use one of the most widely used indicators, the hypervolume [7, 8]. In two dimensions, the hypervolume of a Pareto front approximation is the area dominated by at least one of its solutions, and bounded by a point that is larger in all objectives than all points in the Pareto front.

In what follows, we first describe the outline of the hybrid algorithm that we use, and we explain how we automatically configure it. We perform an experimental analysis that shows that the automatically configured versions reach state-of-the-art performance.

2. ALGORITHM DESIGN

The TP+PLS framework consists of the sequential execution of the TPLS and PLS algorithms. TPLS uses effective single-objective algorithms to solve a sequence of scalarized problems, that is, weighted sum aggregations of the multiple objective functions. We use a recent version called Adaptive Anytime Two-Phase Local Search (AA-TPLS) [9]. Contrary to TPLS, PLS does not rely on

weights, but is a local search method purely based on Pareto dominance. We briefly describe these two algorithms, and how we combine them into a final hybrid algorithm.

Adaptive Anytime Two-Phase Local Search TPLS in its original version [3] consists of two main phases. In the first phase, a high-quality solution is generated for one objective using an effective single-objective algorithm. This high-quality solution is the seed that initializes the second phase. During this second phase, a sequence of scalarizations are tackled. A scalarized single-objective problem is defined from the bi-objective one as follows: a normalized weight vector $\vec{\lambda} = (\lambda, 1-\lambda)$, $\lambda \in [0,1] \subset \mathbb{R}$ is used to compute the scalar value of a solution s with objective function vector $\vec{f}(s) = (f_1(s), f_2(s))$ as $f_\lambda(s) = \lambda \cdot f_1(s) + (1-\lambda) \cdot f_2(s)$. The single-objective algorithm that tackles these scalarizations uses as a seed the best solution found for a previous scalarization. In this way, TPLS can take advantage of any known effective algorithm for each single objective.

In a recent work [9], we have shown that TPLS can be effective but that it has few drawbacks. First, the computation time must be known in advance in order to distribute the computational effort equally in all directions; otherwise, if stopped earlier, the approximation to the Pareto front will be very poor in some regions. Second, it cannot adapt the computational effort to different Pareto front shapes. We proposed AA-TPLS [9] as an improved version of TPLS that has the anytime property, that is, it aims at producing an as high as possible performance at any moment of its execution; moreover, this improved version *adapts* to the shape of the Pareto front, focusing the search on those regions that would improve the overall quality of the Pareto front approximation. Here, we use this new AA-TPLS as a component of the TP+PLS algorithm.

We use an iterated greedy (IG) as the underlying algorithm of AA-TPLS. IG is a stochastic local search method, originally proposed for permutation flow-shop scheduling problems to minimize the makespan, for which it is state-of-the-art [10]. In recent work [11], we adapted IG to minimize other objectives, that is, total tardiness (weighted or not), sum of flowtimes, and scalarized problems arising from all possible pairwise combinations of these three objectives. Automatic configuration tools were used to find efficient parameter settings of IG for each problem. We use these settings here for IG, to focus on the automatic configuration of six parameters that control the behavior of our AA-TPLS framework, that is, the multi-objective part of the combination AA-TPLS &IG.

Pareto Local Search Pareto Local Search (PLS) can be seen as the extension of iterative improvement algorithms from the single to the multi-objective case [12]. In PLS, an acceptance criterion based on Pareto dominance replaces the usual single-objective acceptance criterion.

Given an initial archive of non-dominated solutions, which are initially marked as unvisited, PLS iteratively applies the following steps. A solution s is randomly chosen among the ones in the archive that are still unvisited. Then, the neighborhood of s is fully explored and all neighbors that are not weakly dominated by s or by any solution in the archive are added to the archive. Solutions in the archive dominated by the newly added solutions are removed in order to keep only non-dominated solutions in the archive. Once the neighborhood of s has been fully explored, s is marked as visited. When all solutions in the archive have been visited, the algorithm stops. Despite its relative simplicity, PLS is an important component of state-of-the-art algorithms for the bi-objective traveling salesman problem (bTSP) [13] and bi-objective permutation flow-shop scheduling problems (bPFSP) [11, 2]. As the neighborhood operator of PLS, in [2] we reported experiments using three different operators: two being based on either insertion or exchange moves, and the third being a combination of both (thus considering more solutions but requiring more time to do it). In this work we automatically configure the choice of this operator. The computation time required by PLS is unpredictable, and may depend on the instance and even on the order unvisited solutions in the archive are chosen. The version of PLS used in the final hybrid algorihm is *time bounded*, that is, it simply stops if the time limit is reached.

Hybrid TP+PLS Algorithm Our framework for the hybrid algorithm in this work is based on the two algorithmic schemes introduced above and it is the same as the one proposed in [2].

First single-objective algorithms (in our case, IG algorithms) find a high-quality initial solution for each single objective. Then we use AA-TPLS to perform a series of scalarizations that produces a set of high-quality, non-dominated solutions. This set is then further improved by a time-bounded PLS that uses appropriate neighborhood operators; in the specific case of the problems tackled in this paper, these are an insertion, an exchange operator, or a combination of both. The result is a hybrid TP+PLS algorithm. Through the particular choices of the underlying single-objective algorithms and the neighborhoods of PLS, we can instantiate the framework of the hybrid algorithm for virtually any bi-objective optimization problem. Here, these problems are five bi-objective PFSPs.

The seven parameters (six for AA-TPLS and one for PLS) of the TP+PLS framework are those that define the specific settings used by TPLS and PLS, that is, the multi-objective part of the final algorithm, and the relative duration of these phases. For more details on these parameters we refer to [2].

Automated Hybrid Configuration The automatic configuration tool that we use is I/F-Race [14]. More specifically, we use a new, improved implementation provided by the `irace` software [15]. This tool handles several parameter types: continuous, integer, categorical, and ordered. Continuous and integer parameters take values within a range specified by the user. Categorical parameters can take any value among a set of possibles ones explicitly given by the user, while an ordered parameter is similar to a categorical parameter with a pre-defined strict order of its possible values.

As proposed by López-Ibáñez and Stützle [1], I/F-Race may be used to automatically configure multi-objective algorithms by integrating the hypervolume indicator as the evaluation criterion.

For the automatic configuration process, we generated 500 training instances of each size, 50 jobs and 20 machines (`50x20`) and 100 jobs and 20 machines (`100x20`). These instances were produced following the same procedure described in [6]. I/F-Race is stopped after 5000 runs of TP+PLS, and each run is given a time limit proportional to the instance size of $0.1 \cdot n \cdot m$ seconds, that is, 100 seconds for instances of size `50x20` and 200 seconds for instances of size `100x20`.

We compare the configuration of TP+PLS found by I/F-Race with the configuration reported in the original publication, that are based on a careful experimental analysis to find the best possible parametrization of the algorithm "by hand" [2], to understand the effect of each algorithm component, and the best design choice for each of them. We call these original configurations $conf_{hand}$. In addition, we also run I/F-Race adding $conf_{hand}$ to the initial set of candidate configurations. We call $conf_{tun-rnd}$ the best configuration obtained from running I/F-Race without knowledge of the $conf_{hand}$ configuration, and we call $conf_{tun-ic}$ the best configuration obtained from running I/F-Race using $conf_{hand}$ as an initial configuration.

3. EXPERIMENTAL ANALYSIS

We compare the original configurations proposed in [2] ($conf_{hand}$), where it is shown that the hybrid algorithm using this parametrization greatly improves upon previous state-of-the-art algorithms, with the configurations obtained from the automatic configuration process ($conf_{tun-rnd}$ and $conf_{tun-ic}$).

For the experimental analysis of the configurations, we use 10 instances produced in the same way as the training instances, of size `50x20` and `100x20`. Each experiment is run until a time limit of $0.1 \cdot n \cdot m$ seconds to allow a computation time proportional to the instance size, as suggested by Minella et al. [6]. We repeat each experiment 10 times with different random seeds.

To normalize the hypervolume value across all instances, we first normalize all non-dominated points to the range $[1,2]$, and we compute the hypervolume of the set of normalized points, using as reference point $(2.1,2.1)$.

Graphical analysis To explore graphically the performance of each configuration, we examine their empirical attainment functions (EAF). The EAF of an algorithm provides an estimation of the probability for an arbitrary point in the objective space to be attained by (that is, dominated by or equal to) a solution obtained by a single run of the algorithm. Thus, examining the EAF allows to know with which frequency a region of the objective space is attained by a multi-objective algorithm. By examining the differences between the EAFs of two algorithms, one can not only identify regions of the objective space where one algorithm performs better than another but also know by which magnitude. The differences in favor of each algorithm can be plotted side-by-side and the magnitude of the differences be encoded in gray levels (the darker the color, the higher the difference). For more details, we refer to López-Ibáñez et al. [16].

Figure 1 presents the differences of the EAFs for $conf_{hand}$ and $conf_{tun-rnd}$ for 2 instances of size `50x20`. Other instances and objectives show the same trend: each algorithm performs better in different regions, but one can hardly assess that $conf_{hand}$ or one of the automatically derived configurations outperforms the other across the whole non-dominated front.

Statistical analysis To assess whether the performance differences among the configurations are significant, we perform a statistical test on the overall results. Table 1 presents the mean and standard deviation of the hypervolume for each problem and each configuration, for instances of size `50x20`. We perform a paired t-test with the null hypothesis of equal performance and a confidence level of 0.95, between the $conf_{hand}$ configuration and each of the other two. A bold face indicates that the difference is statistically significant in favor of one of the automatically derived configurations, and an italic face indicates that the difference is statistically significant in favor of $conf_{hand}$. The same test is performed for instances of size `100x20`, and results are reported in Table 2.

In all cases except one (*PFSP*-(*SFT*, *WT*) on Table 2), $conf_{hand}$ obtains the worst results of all the three configurations, the difference being often statistically significant. In particular, $conf_{tun-ic}$ improves in nine out of the ten cases significantly over $conf_{hand}$ (see Tables 1 and 2). Even if the absolute differences in hypervolume are not very large, this is a noteworthy result given the excellent performance that the hybrid TPLS+PLS using the $conf_{hand}$ configuration achieved when compared to previous state-of-the-art algorithms [2].

Table 1: Mean and standard deviation of the normalized hypervolume obtained by each configuration, evaluated over 10 runs and 10 instances of size `50x20`. A bold face indicates that there is a statistically significant difference (see text for details) in favor of a given configuration versus $conf_{hand}$, and an italic face that the difference is in favor of $conf_{hand}$.

	$conf_{hand}$		$conf_{tun-rnd}$		$conf_{tun-ic}$	
	mean	sd	mean	sd	mean	sd
($C_{\max}$, *SFT*)	0.974	0.036	0.982	0.038	**0.984**	0.034
($C_{\max}$, *TT*)	0.999	0.039	**1.005**	0.038	**1.002**	0.035
($C_{\max}$, *WT*)	1.037	0.026	**1.045**	0.024	**1.045**	0.023
(*SFT*, *TT*)	0.954	0.038	0.955	0.039	0.96	0.04
(*SFT*, *WT*)	1.022	0.028	1.024	0.03	**1.029**	0.026

Table 2: Mean and standard deviation of the normalized hypervolume obtained for each configuration, evaluated over 10 runs and 10 instances of size `100x20`. A bold face indicates that there is a statistically significant difference (see text for details) in favor of a given configuration versus $conf_{hand}$, and an italic face that the difference is in favor of $conf_{hand}$.

	$conf_{hand}$		$conf_{tun-rnd}$		$conf_{tun-ic}$	
	mean	sd	mean	sd	mean	sd
($C_{\max}$, *SFT*)	0.943	0.058	**0.968**	0.056	**0.971**	0.058
($C_{\max}$, *TT*)	1.005	0.043	1.008	0.045	**1.012**	0.038
($C_{\max}$, *WT*)	1.013	0.043	**1.028**	0.039	**1.025**	0.04
(*SFT*, *TT*)	0.621	0.129	**0.755**	0.117	**0.761**	0.133
(*SFT*, *WT*)	0.951	0.037	*0.922*	0.051	**0.962**	0.048

4. CONCLUSION

In this work, we automatically configured a new state-of-the-art algorithm for five bi-objective flow-shop problems. The hybrid TP+PLS algorithm that we automatically configure is the same as in [2]. In this previous study we proposed a new state-of-the-art algorithm for bi-objective permutation flow-shop scheduling, together with a highly effective parametrization that should be used for each instance size. In this work, we automatically configured this hybrid algorithm and showed that the configuration we obtained are as-good or even slightly better than the ones originally proposed. The hybrid multi-objective framework that we configure is generic and the same design procedure could be applied to different bi-objective combinatorial problems, potentially improving over the current state-of-the-art for different problems.

5. REFERENCES

[1] M. López-Ibáñez and T. Stützle, "Automatic configuration of multi-objective ACO algorithms," in *Ant Colony Optimization and Swarm Intelligence, 7th International Conference, ANTS 2010*, ser. Lecture Notes in Computer Science, M. Dorigo *et al.*, Eds. Springer, Heidelberg, Germany, 2010, vol. 6234, pp. 95–106.

[2] J. Dubois-Lacoste, M. López-Ibáñez, and T. Stützle, "A hybrid TP+PLS algorithm for bi-objective flow-shop scheduling problems," *Computers & Operations Research*, vol. 38, no. 8, pp. 1219–1236, 2011.

[3] L. Paquete and T. Stützle, "A two-phase local search for the biobjective traveling salesman problem," in *Evolutionary Multi-criterion Optimization (EMO 2003)*, ser. Lecture Notes in Computer Science, C. M. Fonseca *et al.*, Eds. Springer, Heidelberg, Germany, 2003, vol. 2632, pp. 479–493.

[4] ——, "Stochastic local search algorithms for multiobjective combinatorial optimization: A review," in *Handbook of Approximation Algorithms and Metaheuristics*, T. F. Gonzalez,

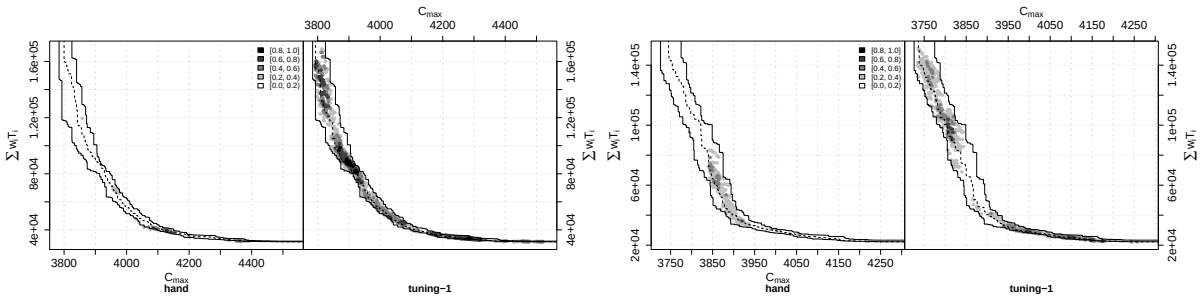


Figure 1: Differences of the empirical attainment functions estimated over 10 runs for $conf_{hand}$ and $conf_{tun-rnd}$, for two instances of size `50x20`, for $conf_{hand}$ (left) versus $conf_{tun-rnd}$ (right). The problem is *PFSP*-($C_{\max}$, *WT*).

Ed. Boca Raton, FL: Chapman & Hall/CRC, 2007, pp. 29–1—29–15.

[5] D. S. Johnson, "Optimal two- and three-stage production scheduling with setup times included," *Naval Research Logistics Quarterly*, vol. 1, pp. 61–68, 1954.

[6] G. Minella, R. Ruiz, and M. Ciavotta, "A review and evaluation of multiobjective algorithms for the flowshop scheduling problem," *INFORMS Journal on Computing*, vol. 20, no. 3, pp. 451–471, 2008.

[7] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. Grunert da Fonseca, "Performance assessment of multiobjective optimizers: an analysis and review," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 2, pp. 117–132, 2003.

[8] C. M. Fonseca, L. Paquete, and M. López-Ibáñez, "An improved dimension-sweep algorithm for the hypervolume indicator," in *Proceedings of the 2006 Congress on Evolutionary Computation (CEC 2006)*. Piscataway, NJ: IEEE Press, Jul. 2006, pp. 1157–1163.

[9] J. Dubois-Lacoste, M. López-Ibáñez, and T. Stützle, "Adaptive "anytime" two-phase local search," in *Learning and Intelligent Optimization, 4th International Conference, LION 4*, ser. Lecture Notes in Computer Science, C. Blum and R. Battiti, Eds. Springer, Heidelberg, Germany, 2010, vol. 6073, pp. 52–67.

[10] R. Ruiz and T. Stützle, "A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem," *European Journal of Operational Research*, vol. 177, no. 3, pp. 2033–2049, 2007.

[11] J. Dubois-Lacoste, M. López-Ibáñez, and T. Stützle, "Effective hybrid stochastic local search algorithms for biobjective permutation flowshop scheduling," in *Hybrid Metaheuristics*, ser. Lecture Notes in Computer Science, M. J. Blesa, C. Blum, L. Di Gaspero, A. Roli, M. Sampels, and A. Schaerf, Eds. Springer, Heidelberg, Germany, 2009, vol. 5818, pp. 100–114.

[12] L. Paquete, M. Chiarandini, and T. Stützle, "Pareto local optimum sets in the biobjective traveling salesman problem: An experimental study," in *Metaheuristics for Multiobjective Optimisation*, ser. Lecture Notes in Economics and Mathematical Systems, X. Gandibleux *et al.*, Eds. Springer, 2004, vol. 535, pp. 177–200.

[13] T. Lust and J. Teghem, "Two-phase Pareto local search for the biobjective traveling salesman problem," *Journal of Heuristics*, vol. 16, no. 3, pp. 475–510, 2010.

[14] P. Balaprakash, M. Birattari, and T. Stützle, "Improvement strategies for the F-race algorithm: Sampling design and iterative refinement," in *Hybrid Metaheuristics*, ser. Lecture Notes in Computer Science, T. Bartz-Beielstein, M. J. Blesa, C. Blum, B. Naujoks, A. Roli, G. Rudolph, and M. Sampels, Eds. Springer, Heidelberg, Germany, 2007, vol. 4771, pp. 108–122.

[15] M. López-Ibáñez, J. Dubois-Lacoste, T. Stützle, and M. Birattari, "The irace package, iterated race for automatic algorithm configuration," IRIDIA, Université Libre de Bruxelles, Belgium, Tech. Rep. TR/IRIDIA/2011-004, 2011.

[16] M. López-Ibáñez, L. Paquete, and T. Stützle, "Exploratory analysis of stochastic local search algorithms in biobjective optimization," in *Experimental Methods for the Analysis of Optimization Algorithms*, T. Bartz-Beielstein, M. Chiarandini, L. Paquete, and M. Preuss, Eds. Springer, Berlin, Germany, 2010, pp. 209–222.

Efficient paths by local search

L. Paquete * J.L. Santos † D.J. Vaz *

* CISUC, Department of Informatics Engineering, University of Coimbra
Pólo II, 3030-290 Coimbra
paquete@dei.uc.pt, dvaz@student.dei.uc.pt

† CMUC, Department of Mathematics, University of Coimbra
3001-454 Coimbra
zeluis@mat.uc.pt

ABSTRACT

In this article, we describe an experimental analysis on a given property of connectedness of optimal paths for the multicriteria shortest path problem. Moreover, we propose a local search that explores this property and compare its performance with an exact algorithm in terms of running time and number of optimal paths found.

Keywords: Multicriteria Optimization, Routing, Local Search, Shortest Path

1. INTRODUCTION

Multicriteria shortest path problems arise in many applications. For instance, GPS systems allow choosing different criteria such as time or cost. However, there is no shortest path that optimizes all criteria since the fastest path may not be the cheapest. For instance, highways are fast but expensive since they are tolled, whereas national roads are free of charge but slow. Hence, one has to develop algorithms that output a set of optimal paths representing the optimal trade-off between the several criteria, from which the user chooses the most preferable.

This work describes a large experimental analysis to understand the structure of the efficient paths that can be exploited from an algorithmic point of view. In particular, we aim to know whether those efficient paths are close to each other, according to a proper definition of "closeness". To know whether this holds for most of the instances is highly relevant, since we could use this information to develop even more effective algorithms [5]. Our experimental results reported indicate that a large number of instances present such property. Therefore, we propose a local search that explores this property and compare it against an exact approach described in the literature.

2. NOTATION AND DEFINITIONS

Let G be a network, $G = (V,A)$ and w a mapping that defines each arc's weight, $w : A \mapsto \mathbb{Z}^Q$. For the simplicity of notation, we will say that a path is a sequence of arcs or nodes, depending of the context. Let us also denote the set of feasible paths as P. The goal of this problem is to find the efficient set of paths as follows

$$\min_{p \in P} f(p) := \left(\sum_{a \in p} w^1(a), \ldots, \sum_{a \in p} w^Q(a) \right) \qquad (1)$$

D.J. Vaz acknowledges its grant BII-2009 from Fundação de Ciência e Tecnologia.

The meaning of operator min is as follows: We say that a feasible path p *dominates* another feasible path p' if and only if $f^j(p) \leq f^j(p')$ for $j = 1, \ldots, Q$, with at least one strict inequality. If there is no feasible path that dominates p, then we say that p is an *efficient path*. The set of all efficient paths is denoted by $\mathscr{N}^E$. The image of the feasible set P forms a set of distinct points in the criterion space. We say that a vector z is *non-dominated* if it is the image of some efficient path $p \in \mathscr{N}^E$. The set of all non-dominated vectors is called the *non-dominated set*. In Eq. (1), operator min finds the nondominated set.

A label correcting algorithm to solve this problem (or to find the efficient set) is given by Paixão and Santos [4], which consists of an adaptation of the algorithm given by Vincke [7]. Although this algorithm finds the efficient set, it is too slow for large networks. In this work, we propose a new local search algorithm that explores a given property of the efficient paths that may improve the running time. We say that two paths, p_1 and p_2, are *adjacent* if and only if, after removing the arcs in common, we obtain a single cycle in the resulting undirected graph [3]. Also, we define the *adjacency graph* G', such that G' has a vertex for each efficient path $p \in \mathscr{N}^E$ and an edge between two vertices if and only if the corresponding paths are adjacent. The algorithm that is reported here explores the *connectedness of efficient paths*, which is defined as the connectedness of G'. Although it is not necessarily true that the efficient set for a given network is connected [3], a large fraction of networks may satisfy this condition. To the knowledge of the authors, connectedness of the efficient set only holds for particular cases of knapsack problems [1, 6].

3. CONNECTEDNESS ANALYSIS

In the experimental investigation mentioned above, we used benchmark instances described in the literature [4]. Those instances are grouped in three categories according to their size: *small*, *medium* and *large*. In each of the categories, there are 7 classes: *RandomN*: Random network (randomly generated arc), with the number of nodes varying, and having constant density and number of criteria; *RandomD*: Random network with constant number of nodes and number of criteria, but varying density; *RandomK*: Random network with constant number of nodes and density, but varying the number of criteria; *CompleteN*: Complete network with constant number of criteria, but varying number of nodes; *CompleteK*: Complete network with constant number of nodes, but varying number of criteria; *GridN*: Grid (square mesh) with constant number of criteria, but varying number of nodes; *GridK*: Grid (square mesh) with constant number of nodes, but varying number of criteria. Each group corresponds to 50 distinct instances. For each class, there are 15-20 groups of 50 instances each. There are 19950 instances, from which 6600 are *small*, 6550 are *medium* and 6800

Algorithm 1: Local Search Algorithm

Input: Network $G = (V,A)$, $s,t \in V$.
Output: Set S.

$T,S := \emptyset$
Let $v : P \mapsto V$
for each crit. $q = 1,\ldots,Q$ **do**
  a) Find T_q, the reversed shortest path tree with root t on crit. q.
  b) Let $p \in T_q$ be path from s to t.
  c) Flag p as *not visited*.
  d) $v(p) := s$
  e) $S := S \cup p$
end for
for each path p in S that is *not visited* **do**
  a) Flag p as *visited*.
  for each node $i \in p$ from $v(p)$ to t **do**
    for each arc $(i,j) \in A$ **do**
      for each crit. $q' = 1,\ldots,Q$ **do**
        a) Let $p_{(s,i)}$ be a path from node s to i and $p_{(s,i)} \subseteq p$
        b) Let $r \in T_{q'}$ be the path from node j to t in crit. q'
        c) $p' := p_{(s,i)} \cup \{(i,j)\} \cup r$
        d) $v(p') := j$
        e) Flag p' as *not visited*
        f) $S :=$ Filter$(S \cup \{p'\})$
      end for
    end for
  end for
end for

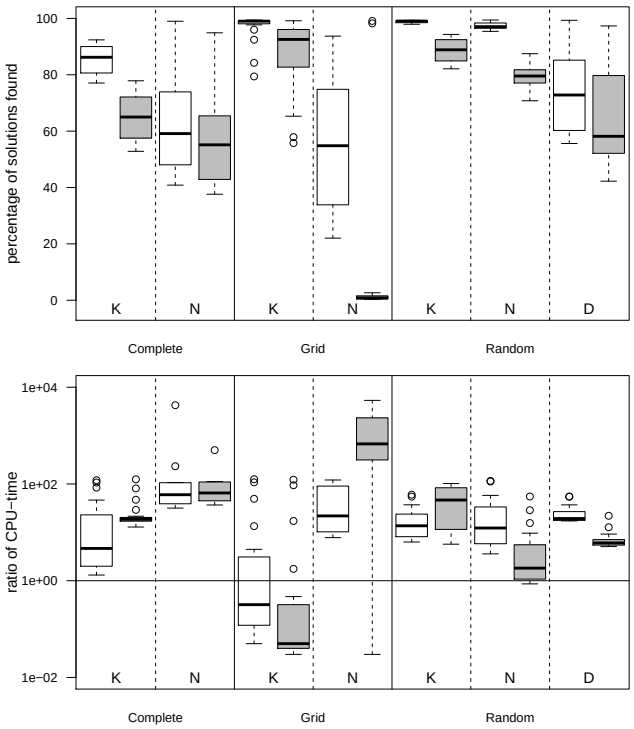


Figure 1: *Percentage of efficient solutions found (top) and ratio of CPU-time between label correcting and local search (bottom). The ratio is shown in logarithmic scale. The white and grey boxplots represent small and medium instances, respectively.*

are *large*. For each of those instances, the weight of each arc for each criterion is generated randomly according to an uniform distribution in the range of $[1,1000]$.

We developed an algorithm for detecting connectedness for a given set of efficient paths. This algorithm outputs the number of connected components of the adjacency graph. For detecting whether a given instance is connected according to the notion of connectedness described in Section 2 we ran the algorithm for finding the set of efficient paths as described by Paixão and Santos [4], and then used this set as input to the algorithm described above to determine if the set of efficient paths was connected. All the *small* and *medium* instances, along with some *large* instances that have been tested, were found to have the set of efficient paths connected.

4. LOCAL SEARCH ALGORITHM

The local search algorithm presented in this section generates candidate efficient paths that are neighbors with respect to the definition of adjacency given in Section 2. Note that the number of efficient and neighbor paths can be exponentially large [2]. Therefore, we focus on a subset of neighbors whose size only depends linearly on the number of criteria, number of nodes and/or arcs.

The local search works as follows. First, all the shortest paths from every node to the target and for each criterion are generated by using Dijkstra's algorithm. Then, for each one of these shortest paths, new paths are generated from a path p as follows: for each node i of p from s to the target t, deviate from p at node i through an arc (i,j) and then, for each criterion, follow the shortest path that was previously computed from node j to the target. With this procedure, further new candidates for efficient paths are generated. The algorithm iterates over the procedure above for all paths that are generated. To avoid generating repeated paths, at each new path p' generated from path p, the algorithm starts from the first node in p' where the detour occured. This node will be denoted by $v(p)$ and for the shortest path initially determined, we define $v(p) = s$. We also denote a path that follows path p from node s to node i by $p_{(s,i)}$. The resulting algorithm is shown in Algorithm 1. The procedure Filter(S) in the final step removes the dominated paths from set S At each iteration of the second loop, the algorithm uses a LIFO strategy to choose the next path from S. In order to define a stopping criterion, we use the following technique [5]: The algorithm flags each new path found as *not visited*; the path becomes *visited* when it is chosen to generate new paths. This algorithm stops when all paths in S are flagged as *visited*.

Figure 1 presents the experimental results obtained by using the local search algorithm as compared to the label correcting approach. The plot in the top gives a boxplot for the percentage of efficient paths found by the local search algorithm for each instance type and size. The plot in the bottom shows a boxplot for the ratio of CPU-time between the label correcting approach and the local search algorithm. The experimental results indicate that the local search algorithm behaved well in Random and Complete instances, where it finds over 80% of the efficient paths in RandomK and RandomN instances and between 50% and 90% in the remaining. For these classes of problems, the local search algorithm takes less than one tenth of the run-time of the exact approach. Additionally, in GridK instances, the local search finds more than 80% of the efficient paths, but it is slower than the exact approach. Finally, only a few portion of efficient paths was found by the local search algorithm in GridN instances.

5. CONCLUDING REMARKS

In this article, we performed an experimental analysis of connectedness for the multicriteria shortest path problem. The positive results obtained in this study suggest that local search algorithms may be an effective approach. We propose a local search algorithm that explores a stricter version of the neighborhood considered for

connectedness. From our point of view, the results obtained by our approach were quite positive for most of the instance types, both in terms of number of efficient paths found and running time.

This approach can even be improved in terms of solution quality, mainly for instances of type Grid with increasing size, by considering an extension of the neighborhood that is explored by our approach. However, it is an open question whether it would still be efficient in terms of running time as compared to exact algorithms for this problem. As for instances of type Grid with increasing number of objectives, a more efficient dominance check may improve our approach. Finally, we remark that this local search exploration can be also applied for other problems defined over networks, such as the multicriteria minimum spanning tree problem.

6. REFERENCES

[1] J. Gorski, L. Paquete, F. Pedrosa, Greedy algorithms for a class of knapsack problems with binary weights, Computers & Operations Research, 2011, in press.

[2] P. Hansen, Bicriterion path problems, In G. Fandel and T. Gal (Eds.), Multiple Criteria Decision Making Theory and Application, LNEMS 177, Springer, pp. 109–127, 1979.

[3] M. Ehrgott, K. Klamroth, Connectedness of efficient solutions in multiple criteria combinatorial optimization. European Journal of Operational Research, 97: 159–166, 1997.

[4] J.P. Paixão and J.L. Santos, Labelling methods for the general case of the multiobjective shortest path problem - a computational study. Working paper CMUC 07-42, University of Coimbra, 2007.

[5] L. Paquete, T. Stützle, On local optima in multiobjective combinatorial optimization problems. Annals of Operations Research, 156(1): 83–97, 2007.

[6] F. Seipp, S. Ruzika, L. Paquete, On a cardinality constrained multicriteria knapsack problem, Report in Wirtschaftsmathematik Nr. 133/2011, University of Kaiserslautern. 2011.

[7] P. Vincke, Problémes multicritères, Cahiers du Centre d'Etudes de Recherche Opérationelle 16, 425–436, 1974

Solving a Multiobjective Flowshop Scheduling Problem by GRASP with Path-Relinking

Iryna Yevseyeva [*] Jorge Pinho de Sousa [*] [†] Ana Viana [*] [‡]

[*] INESC Porto, FEUP campus,
Rua Dr. Roberto Frias, 378, 4200-465 Porto, Portugal
iyevseyeva@inescporto.pt

[†] Faculdade de Engenharia da Universidade do Porto,
Rua Dr. Roberto Frias, s/n, 4200-465 Porto, Portugal
jsousa@inescporto.pt

[‡] Instituto Superior de Engenharia do Porto,
Rua Dr. António Bernardino de Almeida, 431, 4200-072 Porto, Portugal
aviana@inescporto.pt

ABSTRACT

In this work, a hybrid metaheuristic for solving the biobjective flowshop problem with makespan and tardiness objectives is proposed. It is based on the well-known greedy randomized adaptive search procedure (GRASP) with path-relinking adapted to the multiobjective case. The proposed approach is tested on several flowshop instances and compared to existing results from literature with the hypervolume performance measures.

Keywords: Multiobjective, GRASP, Path-relinking, Scheduling, Flowshop

1. INTRODUCTION

Traditionally, scheduling problems are solved with one objective at a time. For instance, minimization of makespan or tardiness. However, in reality several usually conflicting objectives need to be optimized simultaneously. Solving multiobjective problems is not easy, since there exists no single optimal schedule, and the schedule which provides the minimal makespan has a larger tardiness when compared to the schedule with minimal tardiness but larger makespan. The multiobjective nature of the problem leads to the search of the Pareto set of nondominated solutions.

Optimizing any of these objectives is NP-hard [1] and exact methods are able to solve only small size problems. On the other hand, metaheuristics are able to find approximate good quality solutions within feasible computational time. In metaheuristics, the search is directed towards good (at least near optimal) approximate solutions by applying some exploitation (or intensification) techniques, e.g. local search. At the same time, the search space is studied in different directions. This is done by applying some exploration (or diversification) techniques, e.g., multi-start with different random initial points. Powerful metaheuristics are at the core of solving NP-hard multiobjective problems. However, there is a large experience in the development of heuristics for single objective scheduling problems. In this work, an attempt to benefit from both of these approaches is made by developing a multiobjective greedy randomized adaptive search procedure (GRASP) with path-relinking for flowshop scheduling.

Typically, in multiobjective optimization instead of the one best performing solution, a set of Pareto optimal solutions, relatively good according to all objectives, is of interest. Similarly to Evolutionary Algorithms and Scatter Search, GRASP can work with a population of solutions initialized by randomized heuristic(s). The quality of solutions constructed with some heuristic may still be improved with local search techniques. On the other hand, interlinking local optima with path-relinking contributes to the exploration of the search space between good solutions. The combination of multi-start local search that performs exploitation of the objective space with path-relinking that explores the objective space results in a powerful multiobjective metaheuristic discussed in this work.

2. BACKGROUND

2.1. Flowshop scheduling model

In a flowshop problem, n jobs have to be processed on m machines with processing time p_{ij} of each job $j \in J$ on each machine $i \in I$. Then, the total number of all possible schedules is equal to $(n!)^m$. The goal is to find the schedule that is optimal according to some objective function(s). Usually, it is supposed that each machine can process only one job at a time without interruptions and the order of jobs is the same on each machine. When the order of jobs in permutation is known, only $n!$ possible schedules can be constructed. This problem is called permutation flowshop scheduling problem (PFSP) and is the one considered in this paper.

In this work, the schedule is constructed such that it minimizes the maximum makespan (completion time of the last job) C_{max} and minimizes the tardiness T_j, simultaneously (see [1] for examples of other possible objectives). Processing of a job j on a machine i can start only after processing the same job on a machine $i-1$ is finished. Being C_j the completion time of a job j on the last machine, C_{ij} the completion time of a job j on a machine i, and d_j a due date of a job j, $C_{ij} = \max\{C_{i-1,j}; C_{i,j-1}\} + p_{ij}$; $C_{max} = \max\{C_j\}(\forall j)$, is the completion time of the last job on the last machine; and $T_j = \max\{C_j - d_j, 0\}$ is the tardiness of a job j.

2.2. Multiobjective optimization

At the outcome of multiobjective optimization there is a set of non-dominated solutions. Each of such solutions is "optimal" in the sense that improvements in one objective causes degradation in some other one(s). The Pareto set of solutions can be found

by evaluating all solution vectors in the objective space and finding nondominated solutions based on the *Pareto dominance relation*. This relation states that solution x_s is better than solution x_p ($x_s, x_p \in R^d$), if it is strictly better in at least one objective and not worse on the rest of the objectives. Assuming minimization of all objectives this can be written as follows: $f_k(x_s) \leq f_k(x_p) \forall k \in \{1,...,o\}$ and $\exists l \in \{1,...,o\} : f_l(x_s) < f_l(x_p)$, where $f_k(x_s), f_k(x_p)$, $k \in \{1,...,o\}$ are evaluations of x_s, x_p in the objective space R^o.

2.3. GRASP

The greedy randomized adaptive search procedure (GRASP) was developed by Feo and Resende in 1989 [2]. It is a multi-start heuristic that iteratively performs construction of solutions, e.g., with some heuristic(s), and makes a local search around solutions provided by the construction phase. Restarting these two consecutive phases with random initial solutions provides diversity of the final results. On the other hand, local search insures exploitation of neighbourhoods of the solutions found at the construction phase. GRASP with path-relinking was successfully applied to a single objective jobshop scheduling problems in [3], [4]. In this work, GRASP with path-relinking is adapted to the multiobjective flowshop.

3. MULTIOBJECTIVE GRASP WITH PATH-RELINKING FOR THE PFSP

Over the last decades, many multiobjective metaheuristics have been developed, such as multiobjective tabu search, multiobjective simulated annealing, multiobjective genetic algorithms, etc. Most of them have already been applied to flowshop scheduling problems. For a recent and comprehensive survey on solving multiobjective PFSP the reader is referred to [5].

The successful application of GRASP with path-relinking to single objective scheduling problems see e.g., [3], [4], has motivated the extension of GRASP to multiobjective PFSP in this work. GRASP has already been applied to the multiobjective PFSP in [6]. However, their approach is based on the aggregation of multiple objectives into a single one, and, finally, solving a single objective GRASP. On contrary, the multiobjective GRASP with path-relinking (moGRASP-PR) proposed in this work allows searching the Pareto set of non-dominated solutions.

3.1. Construction phase

The original GRASP works with one solution at each iteration, on contrary, here, a population of initial solution is constructed according to some heuristics. The idea of using a population of initial solutions is common to evolutionary algorithms and scatter search, and assumes working in parallel with some set of diverse solutions. Obtaining such solutions with heuristics guarantees that initial solutions are feasible and good according to at least one of the objectives.

In the scheduling literature, there is a long time tradition of developing efficient heuristics for different types of objectives based on dispatching rules. For instance, for the makespan objective, the shortest processing time (SPT) heuristic stands as the best performing one, while for the tardiness objective, the earliest due date (EDD) heuristics is reported to be the best [1]. On the other hand, more complicated heuristics were developed for each type of scheduling problems. For instance, for the flowshop, Nawaz, Dudek and Ham (NEH) heuristic [7] is considered to be the most efficient one [8].

In this work, several heuristics were selected for constructing an initial population. For the makespan and tardiness objectives the best solutions were obtained with the NEH heuristics with jobs initially ordered according to the LPT and EDD rules respectively. Then, to diversify the initial population, usually at the cost of quality of solutions, the rest of solutions is selected from two Restricted Candidate Lists (RCLs), constructed for the makespan objective according to the SPT rule and for tardiness with respect to the EDD rule.

Assuming minimization of the maximum makespan $C_j = C_{max}$, $\forall j$, only some jobs with the smallest makespan are selected into the RCL. The α-part of the best jobs is defined between the jobs with the minimal $\underline{C_j}$ and maximal $\overline{C_j}$ values of makespan. Consequently, $\underline{C_j} = \min(C_j | j \in J_a)$ and $\overline{C_j} = \max(C_j | j \in J_a)$, where J_a is the set of unscheduled jobs. Then, the RCL can be defined as follows

$$\text{RCL} = \{j \in J_a | \underline{C_j} \leq C_j \leq \overline{C_j} + \alpha(\overline{C_j} - \underline{C_j})\}, \qquad (1)$$

where the α parameter, such as $0 \leq \alpha \leq 1$ is selected depending on the degree of randomness desired. Similarly, the RCL for EDD is constructed.

Selection from the RCL of the job to be scheduled is random and guarantees diversification of solutions selected in the initial population from the same list. Half of the initial population is selected from the RCL constructed based on SPT and the other half is taken from the RCL constructed based on EDD. Such construction creates good solutions according to only one objective at a time, but not with respect to both objectives simultaneously.

3.2. Local search phase

Construction with a greedy-randomized approach does not guarantee optimality of the solutions, that is why local search can still improve the quality of solutions by exploring the neighbourhoods of the best solutions obtained at the construction phase. A *neighbourhood* $N(x)$ of a solution $x \in X$ is a set of solutions that are obtained by slightly changing x (by an operation called *move*) in some specific way for each particular type of problem. When compared to single objective optimization, where either the first improving solution better than the current one or the best among all possible solutions in the whole neighbourhood is accepted, in the multiobjective case, all non-dominated solutions with respect to the neighbourhood are accepted.

Due to the importance of the order of jobs in the flowshop problem, the most efficient neighbourhoods for it are those that destroy the order of jobs as less as possible. In this sense, the less destructive is the insertion neighbourhood that removes a job from its current position and inserts in some other random position. The swapping neighbourhood that exchanges positions of two randomly selected jobs is also shown to be efficient. Due to the quadratic growth of both neighbourhoods, when increasing the size of the problem solved, usually, only some fixed-size sub-neighbourhoods of solutions selected randomly are considered for evaluation. Exploring sub-neighbourhoods does not guarantee identification of local optima. However, it is compensated by consuming less of the available computation time.

3.3. Path-relinking

In the original GRASP, the local search exploits a region of the search space around some starting point, and exploration is compensated by the multiple restarts. However, from evolutionary computation it is known that taking two good solutions and swapping parts of them may result in new good solutions. Such operation is known as crossover. Glover suggested a more deterministic approach to trace solutions that are located on trajectories (or

paths) connecting good (or elite) solutions [9]. The basic idea is to transform one of the solutions (initial one) in the direction of the other (guiding) solution. Small changes to the initial solution allow to obtain new solutions located in between of the two relinked solutions. In [10], path-relinking was applied together with GRASP, and in [4], this approach was successfully applied to job shop scheduling.

There are different ways to perform path-relinking [11]. Here, a new solution is obtained by swapping elements of the initial solution with respect to the position of one of them in the guiding solution. The new solution is evaluated on objectives and compared to the initial solution. In case the new solution dominates the initial one it gets the chance to be added to the archive, see next section. In [12], it was demonstrated that for some problems, the solutions located near local optima are better than those located on the rest of the paths. This principle is used in the *truncating* path-relinking, where only sub-paths (and, consequently, sub-neighbourhoods) near to local optima are explored.

Depending on the size of the set to be interlinked, the neighbourhood created by path-relinking can be large, and, consequently, time-consuming. That is why here to reduce computational complexity and still keep the quality of solutions high, the truncating path-relinking is applied for all possible pairs of nondominated solutions.

3.4. Archive of solutions

All nondominated solutions found after construction, local search and path-relinking phases are stored in an external archive. The solution is added to the archive if it is non-dominated with respect to the solutions already stored in the archive. On the other hand, solutions stored in the archive that become dominated by the current solution are removed from it. An archive of solutions has already been used in the elitist multiobjective GRASP for the quadratic assignment problem in [13].

To reduce computational efforts, each new neighbour solution is first compared for the non-dominance with the original solution from which it stemmed from, either initial solution of the local search (LS) or initial and/or guiding solutions in case of path-relinking (PR). In case the new solution appears to be non-dominated, it is compared to all solutions stored in the archive for non-dominance. Such approach was suggested in [14] to avoid redundant computations.

3.5. moGRASP-PR routine

The general scheme of the main loop of the moGRASP-PR cycle applied to the current population is presented in Algorithm 1.

Algorithm 1: moGRASP-PR main loop

Require: Number of generation $NGEN$, population size $NIND$
$gen = 0$
$archive = \{\}$
while $gen < NGEN$ **do**
  $init\_pop = construct(NIND)$
  $archive \leftarrow non\_dominated(init\_pop \bigcup archive)$
  $neighbours = LS(archive)$
  $archive \leftarrow non\_dominated(neighbours \bigcup archive)$
  $neighbours = PR(archive)$
  $archive \leftarrow non\_dominated(neighbours \bigcup archive)$
  $neighbours = LS(archive)$
  $archive \leftarrow non\_dominated(neighbours \bigcup archive)$
  $gen = gen + 1$
end while

At the beginning, a population of solutions with size $NIND = 102$ is constructed. The two solutions are constructed with the NEH heuristics for makespan and tardiness, respectively, initially ordered according to the LPT and EDD heuristics. Then, for the predefined number of generations, $NGEN = 20$, the rest of the population is composed of 50 solutions, constructed from the best jobs selected randomly from the RCL based on the SPT rule, and 50 solutions selected from the RCL based on the EDD rule. The non-dominated solutions are selected from all constructions and preserved in the archive. Then, the archive is updated at each iteration after new neighbours are created with either local search or path-relinking.

3.6. Performance assessment

Among different performance measures available in the literature, the hypervolume [15] is considered to have good convergence properties [16]. The main disadvantages of the hypervolume is its computational complexity. Recently, several efficient algorithms that try to reduce computational time for calculating the hypervolume have been proposed. In this work, the hypervolume is computed with the improved dimension-sweep algorithm proposed in [17].

4. EXPERIMENTAL RESULTS

In this work, we compare the performance of the multiobjective GRASP with path-relinking applied to the PFSP with the results of 21 best algorithms obtained in [5]. The experiments presented in table 1 were performed on the well-known Taillard's benchmark for the PFSP proposed in [18] and modified for the makespan and tardiness objectives in [5]. The set of 110 instances is available for download at http://soa.iti.es/files/Taillard_DueDates.7z. In this work, the instances with different combinations $[n \times m]$ of n jobs and m machines are used, such as $[20,50] \times [5,10,20]$ and $[100 \times 5]$. The moGRASP-PR algorithm runs for 1 (for large size problems) to 3 (for small size problems) times on each instance (in total 226 problems are solved) with 20 GRASP iterations in each run. The results presented in table 1 are the averages of hypervolumes for each algorithm on all runs.

Table 1: Comparative results with the hypervolume criterion

Method	Hypervol	Method	Hypervol
MOSA_Varadhar	0.927	ε-NSGAII	0.71
MOGALS_Arroyo	0.861	moGRASP-PR	0.703
PESA	0.851	$(\mu+\lambda)$-PAES	0.605
PESAII	0.848	ε-MOEA	0.621
PGA_ALS	0.815	PAES	0.588
MOTS	0.795	MOSA_Suresh	0.851
MOGA_Murata	0.755	SA_Chakrav	0.515
CMOGA	0.741	PILS	0.43
NSGAII	0.725	ENGA	0.426
SPEA	0.724	A-IBEA	0.159
CNSGAII	0.722	SPEAII	0.159

A more detailed analysis shows that the algorithm proposed in this paper performs best compared to the rest of the algorithms for small data sets. However, for the large data sets it converges prematurely. That is why, the overall hypervolume is not as high as expected. Some diversity preservation mechanism is planned to be integrated into the algorithm in the near future.

The moGRASP-PR algorithm is implemented in Python 2.6 on a single Intel Core 2 Duo T9550 processor running at 2.66 GHz with

4 GB of RAM.

5. CONCLUSIONS AND FUTURE WORK

In this work, a multiobjective GRASP with path-relinking is applied to a biobjective flowshop problem with the makespan and the tardiness objectives. The approach was tested on several flowshop instances and compared to existing methods with the hypervolume performance measures. The initial results are promising for the small flowshop instances, but the performance of the algorithm should be investigated for the larger ones.

6. ACKNOWLEDGEMENTS

The authors are grateful to Nicolau Santos and Rui Rei for assisting with the Python code and to Gerardo Minella for comments on his paper.

7. REFERENCES

[1] M. L. Pinedo, *Scheduling: Theory, Algorithms, and Systems*. Springer, 2008.

[2] T. Feo and M. G. C. Resende, "A probabilistic heuristic for a computationally difficult set covering problem," *Operations Research Letters*, vol. 8, pp. 67–71, 1989.

[3] S. Binato, W. Hery, D. M. Loewenstern, and M. G. C. Resende, "A grasp for job shop scheduling," in *Essays and Surveys on Metaheuristics*. Kluwer Academic Publishers, 2000, pp. 59–79.

[4] R. M. Aiex, S. Binato, and M. G. C. Resende, "Parallel grasp with path-relinking for job shop scheduling," *Parallel Comput.*, vol. 29, no. 4, pp. 393–430, 2003.

[5] G. Minella, R. Ruiz, and C. M., "A review and evaluation of multiobjective algorithms for the flowshop scheduling problem," *INFORMS Journal on Computing*, vol. 20, no. 3, pp. 451–471, 2008.

[6] B. S. H. Khan, G. Prabhaharan, and P. Asokan, "A grasp algorithm for m-machine flowshop scheduling problem with bicriteria of makespan and maximum tardiness," *Int. J. Comput. Math.*, vol. 84, no. 12, pp. 1731–1741, 2007.

[7] M. Nawaz, J. E. E. Enscore, and I. Ham, "A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem," *Omega*, vol. 11, no. 1, pp. 91 – 95, 1983.

[8] E. Taillard, "Some efficient heuristic methods for the flow shop sequencing problem," *European Journal of Operational Research*, vol. 47, no. 1, pp. 65–74, July 1990.

[9] F. Glover, "Tabu search and adaptive memory programing – advances, applications and challenges," in *Interfaces in Computer Science and Operations Research*. Kluwer, 1996, pp. 1–75.

[10] M. Laguna and R. Martí, "Grasp and path relinking for 2-layer straight line crossing minimization," *INFORMS Journal on Computing*, vol. 11, pp. 44–52, 1999.

[11] M. G. C. Resende and C. C. Ribeiro, "Grasp with path-relinking: recent advances and applications," in *Metaheuristics: Progress as Real Problem Solvers*, T. Ibaraki, K. Nonobe, and M. Yagiura, Eds. Springer, 2005, pp. 29–63.

[12] M. G. C. Resende, R. Martí, M. Gallego, and A. Duarte, "Grasp and path relinking for the max-min diversity problem," *Comput. Oper. Res.*, vol. 37, no. 3, pp. 498–508, 2010.

[13] H. Li and D. Landa-Silva, "An elitist grasp metaheuristic for the multi-objective quadratic assignment problem," in *Evolutionary Multi-Criterion Optimization*, ser. Lecture Notes in Computer Science, M. Ehrgott, C. Fonseca, X. Gandibleux, J.-K. Hao, and M. Sevaux, Eds. Springer, 2009, vol. 5467, pp. 481–494.

[14] L. Paquete, M. Chiar, and T. Stützle, "Pareto local optimum sets in the biobjective traveling salesman problem: An experimental study," in *Metaheuristics for Multiobjective Optimization, Lecture*. Springer, 2004, pp. 177–200.

[15] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 4, pp. 257–271, 1999.

[16] J. Knowles and D. Corne, "On metrics for comparing nondominated sets," in *Proceedings of the 2002 Congress on Evolutionary Computation (CEC 2002)*, Honolulu, Hawaii, 2002, pp. 711–716.

[17] C. M. Fonseca, L. Paquete, and M. López-Ibáñez, "An improved dimension-sweep algorithm for the hypervolume indicator," in *IEEE Congress on Evolutionary Computation*, Vancouver, Canada, 2006, pp. 1157–1163.

[18] E. Taillard, "Benchmarks for basic scheduling problems," *European Journal of Operational Research*, vol. 64, no. 2, pp. 278–285, 1993.

Stabilized Column Generation for the Rooted Delay-Constrained Steiner Tree Problem

Markus Leitner * Mario Ruthmair * Günther R. Raidl *

* Institute of Computer Graphics and Algorithms, Vienna University of Technology
Favoritenstr. 9-11, 1040 Vienna, Austria
{leitner, ruthmair, raidl}@ads.tuwien.ac.at

ABSTRACT

We consider the rooted delay-constrained Steiner tree problem which arises for example in the design of centralized multicasting networks where quality of service constraints are of concern. We present a path based integer linear programming formulation which has already been considered in the literature for the spanning tree variant. Solving its linear relaxation by column generation has so far been regarded as not competitive due to long computational times needed. In this work, we show how to significantly accelerate the column generation process using two different stabilization techniques. Computational results indicate that due to the achieved speed-up our approach outperforms so-far proposed methods.

Keywords: Network design, Stabilized column generation, Delay-constrained Steiner tree

1. INTRODUCTION

When designing a communication network with a central server broadcasting or multicasting information to all or some of the participants of the network, some applications such as video conferences require a limitation of the maximal delay from the server to each client. Beside this delay-constraint minimizing the cost of establishing the network is in most cases an important design criterion. As another example, consider a package shipment organization with a central depot guaranteeing its customers a delivery within a specified time horizon. Naturally the organization aims at minimizing the transportation costs but at the same time has to hold its promise of being in time. Such network design problems can be modeled as *rooted delay-constrained Steiner tree problem (RDCSTP)*, which is an NP-hard combinatorial optimization problem [1]. The objective is to find a minimum cost Steiner tree of a given graph with the additional constraint that the total delay along each path from a specified root node to any other required node must not exceed a given delay bound.

More formally, we are given an undirected graph $G = (V,E)$ with a set V of n nodes, a fixed root node $s \in V$, a set $T \subseteq V \setminus \{s\}$ of terminal or required nodes, a set $S = V \setminus (T \cup \{s\})$ of optional Steiner nodes, a set E of m edges, a cost function $c : E \to \mathbb{Z}^+$, a delay function $d : E \to \mathbb{Z}^+$, and a delay bound $B \in \mathbb{Z}^+$. A feasible solution to the RDCSTP is a Steiner tree $G_S = (V_S, E_S)$, $s \in V_S$, $T \subseteq V_S \subseteq V$, $E_S \subseteq E$ satisfying the constraints $\sum_{e \in P_S(t)} d_e \leq B$, $\forall t \in T$, where $P_S(t) \subseteq E$ denotes the edge set of the unique path from root s to terminal t. An optimal solution G_S^* is a feasible solution with minimum costs $c(G_S^*) = \sum_{e \in E_S} c_e$.

2. PREVIOUS & RELATED WORK

There are many recent publications dedicated to this problem and its more special variants. Several metaheuristics have been applied to the RDCSTP, such as GRASP [2, 3], path-relinking [4] and variable neighborhood search [3]. More heuristic approaches can be found for the spanning tree variant with $T = V \setminus \{s\}$, e.g. GRASP and variable neighborhood descent (VND) in [5] and ant colony optimization and variable neighborhood search in [6]. Furthermore, preprocessing methods are presented in [6] to reduce the size of the graph significantly in order to speed up the solving process. Exact methods based on integer linear programming (ILP) have been explored by Leggieri et al. [7] who describe a compact extended node-based formulation using lifted Miller-Tucker-Zemlin inequalities. Since the used Big-M inequalities usually yield rather low linear programming (LP) relaxation bounds this formulation is improved by separating directed connection cuts. Several ILP approaches for the spanning tree variant have been examined by Gouveia et al. in [8] based on a path formulation solved by two different methods. Standard column generation (CG) turns out to be computationally inefficient while a Lagrangian relaxation approach together with a fast primal heuristic exhibits better performance. A third approach reformulates the constrained shortest path problem on a layered graph and solves it using a multi commodity flow (MCF) formulation. Since the size of the layered graph and therefore the efficiency of the according model heavily depends on the number of achievable discrete delay values this approach can in practice only be used for instances with a quite restricted set of achievable delay values. Additionally a MCF model usually suffers from the huge amount of flow variables used altogether leading to a slow and memory-intensive solving process. Nevertheless solving these layered graph models turned out to be very effective on certain classes of instances.

3. PATH FORMULATION

In this section we present a path based ILP formulation for the RDCSTP which is a straightforward modification of the model discussed by Gouveia et al. [8] for the spanning tree variant of the RDCSTP. In our directed formulation we use arc set A containing an arc (s, j) for each edge $\{sj\} \in E$ incident to the root node and two oppositely directed arcs (i, j), (j, i) for all other edges $\{ij\} \in E$, $i, j \neq s$. We further assume the edge cost and delay functions to be defined on the set of arcs too, i.e. $c_{ij} = c_e$ and $d_{ij} = d_e$, $\forall (i, j) \in A, e = \{ij\} \in E$. The *integer master problem* (IMP) defined by (1)–(6) is based on variables $x_{ij} \in \{0, 1\}$, $\forall (i, j) \in A$, which indicate arcs included in the directed solution. We further use path variables $\lambda_p \in \{0, 1\}, \forall p \in P = \bigcup_{t \in T} P_t$, where $P_t \subseteq 2^A$ is the set of feasible paths from the root node s to terminal t. Each path is represented by its arc set. A path $p \in P_t$ to terminal $t \in T$ is feasible if and only if it satisfies the delay bound, i.e.

$\sum_{(i,j)\in p} d_{ij} \leq B$. Variable λ_p is set to one if path $p \in P$ is realized.

$$\text{(IMP)} \quad \min \sum_{(i,j)\in A} c_{ij} x_{ij} \tag{1}$$

$$\text{s.t.} \quad \sum_{p\in P_t} \lambda_p \geq 1 \qquad \forall t \in T \tag{2}$$

$$x_{ij} - \sum_{p\in P_t | (i,j)\in p} \lambda_p \geq 0 \qquad \forall t \in T,\ \forall (i,j) \in A \tag{3}$$

$$\sum_{(i,j)\in A} x_{ij} \leq 1 \qquad \forall j \in V \tag{4}$$

$$x_{ij} \in \{0,1\} \qquad \forall (i,j) \in A \tag{5}$$

$$\lambda_p \in \{0,1\} \qquad \forall p \in P \tag{6}$$

Since the number of feasible paths for each terminal $t \in T$ and thus the total number of variables in the model is in general exponentially large, we apply CG – see e.g. [9, 10] – to solve the LP relaxation. We start with a small subset $\tilde{P}_t \subseteq P_t$, $\forall t \in T$, of path variables λ_p in the *restricted master problem* (RMP) where the integrality conditions on arcs (5) and paths (6) are replaced by (7) and (8), respectively. Further variables are added on demand according to the solution of the pricing subproblem.

$$x_{ij} \geq 0 \qquad \forall (i,j) \in A \tag{7}$$

$$\lambda_p \geq 0 \qquad \forall p \in P \tag{8}$$

Let $\mu_t \geq 0$, $\forall t \in T$, denote the dual variables associated to the convexity constraints (2) and $\pi^t_{ij} \geq 0$, $\forall t \in T$, $\forall (i,j) \in A$, denote the dual variables associated to the coupling constraints (3). Then the pricing subproblem is defined as

$$(t^*, p^*) = \text{argmin}_{t\in T, p\in P_t} -\mu_t + \sum_{(i,j)\in p} \pi^t_{ij}. \tag{9}$$

Hence we need to solve a resource constrained shortest path problem on a graph (V,A) with nonnegative arc costs π^t_{ij}, $\forall (i,j) \in A$, for each terminal $t \in T$. We solve each such problem in pseudo-polynomial time $O(B \cdot |A|)$ using the dynamic programming based algorithm from [8]. As long as path variables λ_p, $p \in P_t$, $t \in T$ with negative reduced costs $\bar{c} = -\mu_t + \sum_{(i,j)\in p} \pi^t_{ij}$ exist, we need to add at least one of them and resolve the RMP. This process is repeated until no further variable with negative reduced costs exists.

In each iteration we add for each terminal $t \in T$ multiple path variables using the approach from [8]: We consider all nodes $v \in V$ that are adjacent to terminal t and all delay bounds $b = 0, \ldots, B - d_{vt}$ for which a path from s to v in conjunction with arc (v,t) is a feasible path to t. In case a shortest path p to v of total delay b, $b = 0, \ldots, B - d_{vt}$, exists and $p' = p \cup \{(v,t)\}$ yields negative reduced costs, the corresponding variable is added to the RMP.

4. COLUMN GENERATION STABILIZATION

It is well known that basic CG approaches typically suffer from computational instabilities such as degeneracy or the tailing-off effect [11] which often increase the needed computational effort for solving them dramatically. Stabilization techniques to reduce the effects of these instabilities are usually classified into problem specific approaches such as the usage of dual-optimal inequalities [12, 13] and problem independent approaches, see e.g. [14, 15]. The latter are often based on the concept of stability centers and deviations from a current stability center are penalized, e.g. by using piecewise linear penalty functions. Recently, we showed how to significantly accelerate the CG process for a survivable network design problem without modifying the RMP by choosing alternative dual optimal solutions when solving the pricing subproblem [16, 17, 18]. In the following we will adapt this technique for the RDCSTP before we discuss an alternative stabilization approach based on piecewise linear penalty functions.

4.1. Alternative Dual Optimal Solutions

Let $\gamma_j \leq 0$, $\forall j \in V$, be the dual variables associated to constraints (4) and $\tilde{P} = \bigcup_{t\in T} \tilde{P}_t$ denote the set of paths for which corresponding variables have already been included in the RMP. Then the dual of the RMP is given by (10)–(15).

$$\max \sum_{t\in T} \mu_t + \sum_{j\in V} \gamma_j \tag{10}$$

$$\text{s.t.} \quad \sum_{t\in T} \pi^t_{ij} + \gamma_j \leq c_{ij} \qquad \forall (i,j) \in A \tag{11}$$

$$\mu_t - \sum_{(i,j)\in p} \pi^t_{ij} \leq 0 \qquad \forall t \in T,\ \forall p \in \tilde{P}_t \tag{12}$$

$$\mu_t \geq 0 \qquad \forall t \in T \tag{13}$$

$$\pi^t_{ij} \geq 0 \qquad \forall t \in T,\ \forall (i,j) \in A \tag{14}$$

$$\gamma_j \leq 0 \qquad \forall j \in V \tag{15}$$

Let (μ^*, π^*, γ^*) denote the current dual solution computed by the used LP solver when solving the RMP. It is easy to see that for arcs A' not part of any so far included path – i.e. $A' = \{(i,j) \in A \mid \nexists p \in \tilde{P} : (i,j) \in p\}$ – any values for the dual variables π_{ij} are optimal as long as $\sum_{t\in T} {\pi^t_{ij}}^* + {\gamma_j}^* \leq c_{ij}$, $\forall (i,j) \in A'$, since they do not occur in inequalities (12). Dual variable values ${\pi^t_{ij}}^*$, $t \in T$, may also be increased for arcs $(i,j) \in A \setminus A'$ if inequalities (11) are not binding. We conclude that any values $\pi^t_{ij} \geq {\pi^t_{ij}}^*$, $\forall (i,j) \in A$, $\forall t \in T$, are dual optimal if $\sum_{t\in T} \pi^t_{ij} \leq \sum_{t\in T} {\pi^t_{ij}}^* + \delta_{ij}$, $\forall (i,j) \in A$ holds, where $\delta_{ij} = c_{ij} + |\gamma_j| - \sum_{t\in T} {\pi^t_{ij}}^*$, $\forall (i,j) \in A$. Note that state-of-the-art LP solvers such as IBM CPLEX, which we use in our implementation, usually choose minimal optimal dual variable values, i.e. ${\pi^t_{ij}}^* = 0$, $\forall t \in T$, $\forall (i,j) \in A'$.

On the contrary to most other stabilization approaches we do not modify the RMP. Instead we aim to choose alternative dual optimal solutions which facilitate the generation of those path variables relevant for solving the LP relaxation of the IMP by increasing the dual variable values used as arc costs in the pricing subproblem. Obviously, we can simply split the potential increase δ_{ij} equally to all relevant dual variables, i.e. use alternative dual variables $\bar{\pi}^t_{ij} = {\pi^t_{ij}}^* + \frac{\delta_{ij}}{|T|}$, $\forall t \in T$, $\forall (i,j) \in A$. In our previous work for a survivable network design problem [16, 17, 18], however, it turned out to be beneficial to initially use different dual optimal solutions, one for each terminal t, which finally converge towards $\bar{\pi}^t_{ij}$, $\forall t \in T$, $\forall (i,j) \in A$. Hence, we consider two additional variants whose correspondingly used dual variables will be denoted as $\tilde{\pi}^t_{ij}$ and $\hat{\pi}^t_{ij}$, $\forall t \in T$, $\forall (i,j) \in A$, respectively. Equation (16) defines dual variable values $\tilde{\pi}^{t'}_{ij}$ used in the pricing subproblem when considering terminal $t' \in T$. Parameter $q \in \mathbb{N}$, $1 \leq q \leq |T|$, is initially set to one and gradually incremented by $\max\{1, \frac{|T|}{10}\}$ in case no negative reduced cost path has been found. After at most ten such major steps $\tilde{\pi}^{t'}_{ij} = \bar{\pi}^{t'}_{ij}$, for each terminal t'. Thus, we can terminate the CG process if $q = |T|$ and no path variables have been added.

$$\tilde{\pi}^{t'}_{ij} = \begin{cases} {\pi^t_{ij}}^* + \frac{\delta_{ij}}{q} & \text{if } t = t' \\ {\pi^t_{ij}}^* & \text{otherwise} \end{cases}, \forall (i,j) \in A. \tag{16}$$

Dual variable values $\hat{\pi}^t_{ij}$ correspond to $\tilde{\pi}^t_{ij}$ except for the fact that q is directly set to $|T|$ once no new negative reduced cost path can be found when using $q = 1$.

4.2. Piecewise Linear Stabilization

As mentioned above other successful stabilization techniques are often based on penalizing deviations from a current stability cen-

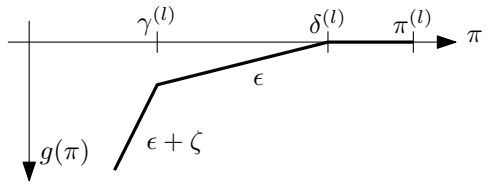


Figure 1: *Piecewise linear dual penalty function* $g(\pi)$.

ter by adding piecewise linear penalty functions to the dual problem. Among these, especially five-piecewise linear function have shown to frequently yield good results if all parameters are chosen carefully; compare [19]. In our case, however, preliminary tests with various settings and concrete parameter values showed that due to a large number of relatively time consuming updates of the stability center this concept does not seem to pay off. Since high dual variable values facilitate the generation of good path variables it is reasonable to penalize only small dual variable values. Hence we use a modified version of the approach from [14] where in each major iteration $l \in \mathbb{N}$, $l \geq 1$, only dual variable values smaller than the current stability center $\pi^{(l)} \in \mathbb{R}_+^{|T|\cdot|A|}$ are penalized according to vectors $\delta^{(l)}, \gamma^{(l)} \in \mathbb{R}_+^{|T|\cdot|A|}$, see Figure 1. Let $\pi_{ij}^{t\,*}$, $\forall t \in T$, $\forall (i,j) \in A$, denote the dual variable values after the CG approach on the penalized model at major iteration l terminates. If there exists at least one dual variable value in the penalized region – i.e. if $\exists t \in T \wedge (i,j) \in A : \pi_{ij}^{t\,*} < \delta_{ij}^{t\,(l)}$ – we need to update the stability center according to the current dual solution – i.e. $\pi^{(l+1)} = \pi^*$ – and correspondingly set $\delta^{(l+1)}$ and $\gamma^{(l+1)}$ and continue the CG process. As has been shown previously [14] this process, which needs to be repeated until each dual variable value lies within an unpenalized region, terminates yielding the LP relaxation value of the IMP after finitely many steps.

According to preliminary tests, the following settings have been chosen for our computational experiments. We set $\varepsilon = 0.3$ and $\zeta = 1$, the size of the inner trust region $T^{(l)} = \pi^{(l)} - \delta^{(l)} = 1$ while $\pi^{(l)} - \gamma^{(l)} = 5 \cdot T^{(l)}$ for all dimensions, i.e. $\forall t \in T$, $\forall (i,j) \in A$. Let A' denote the set of arcs used by the paths included in the initial model. We set $\pi_{ij}^{t\,(1)} = \pi_{ij}^{t\,*} + \delta_{ij}$, $\forall t \in T$, $\forall (i,j) \in A'$, $\pi_{ij}^{t\,(1)} = \pi_{ij}^{t\,*} + \frac{\delta_{ij}}{|T|}$, $\forall t \in T$,$\forall (i,j) \notin A'$.

5. COMPUTATIONAL RESULTS

All computational experiments have been performed on a single core of a multi-core system consisting of Intel Xeon E5540 processors with 2.53 GHz and 3 GB RAM per core. We used IBM CPLEX 12.2 as LP solver and applied an absolute time limit of 10000 CPU-seconds to all experiments. All preprocessing methods mentioned in [6] are used to reduce the input graphs prior to solving. To build an initial set of paths a simple construction heuristic is applied on Steiner tree instances: the delay constrained shortest paths to all terminal nodes are iteratively added to the tree dissolving possible cycles. On instances where $T = V \setminus \{s\}$ we apply the Kruskal-based heuristic followed by a VND both introduced in [5]. Tables (1) and (2) report average CPU-times in seconds and needed iterations for different instance sets. In both tables π^* denotes the unstabilized CG approach, and $\bar{\pi}$, $\hat{\pi}$, and $\tilde{\pi}$ refer to the three strategies discussed in Section 4.1 for using alternative dual optimal solutions in the pricing subproblem. The piecewise linear stabilization approach from Section 4.2 is denoted by PL, Lag_G and CG_G denote the Lagrangian and CG approach from [8], respectively. The results of the latter two have, however, been computed on a different hardware using an older CPLEX version for the CG approach and are thus not directly comparable.

		$\overline{\text{CPU time [s]}}$								$\overline{\text{Iterations}}$					
Set	B	Lag_G	CG_G	π^*	$\bar{\pi}$	$\hat{\pi}$	$\tilde{\pi}$	PL		CG_G	π^*	$\bar{\pi}$	$\hat{\pi}$	$\tilde{\pi}$	PL
r,100	100	493	4752	314	13	15	**10**	72		1041	189	**25**	39	92	115
	150	639	8215	111	10	**8**	**8**	48		12561	357	**26**	42	98	144
	200	288	10001	123	**4**	**4**	8	46		18736	904	**28**	41	102	238
	250	526	10001	261	5	**4**	9	71		24881	1676	**32**	44	115	325
c,100	100	809	10026	38	10	**9**	12	78		480	176	**31**	44	96	171
	150	544	10034	135	26	**15**	18	142		329	346	**41**	56	118	187
	200	711	10061	1151	50	37	**21**	367		314	697	**58**	69	123	311
	250	1066	10076	3779	43	27	**25**	500		327	2702	**68**	78	141	444
e,100	100	976	10033	481	90	75	**25**	598		239	208	**40**	64	115	307
	150	1817	10106	3980	705	356	**66**	2927		193	364	**52**	84	138	403
	200	2972	10096	9297	5148	2670	**177**	8607		209	397	**92**	123	172	459
	250	4008	10104	10000	7013	3489	**142**	9090		195	357	**98**	160	203	339
r,1000	1000	971	8064	25	7	**6**	11	25		891	119	**22**	39	96	84
	1500	1744	8538	112	12	**10**	16	60		4240	253	**27**	43	112	118
	2000	869	10002	220	**11**	**11**	20	70		15600	716	**28**	42	114	125
	2500	790	10007	535	14	**12**	20	89		18233	1527	**34**	48	124	156
c,1000	1000	668	8186	60	26	24	**18**	82		869	91	**26**	38	84	109
	1500	942	10024	112	30	**25**	33	111		418	163	**37**	46	104	122
	2000	2389	10037	788	68	57	**34**	235		451	401	**36**	58	109	188
	2500	1256	10037	1272	70	**44**	48	425		437	953	**53**	62	122	261
e,1000	1000	2846	10065	137	52	34	**25**	474		615	129	**34**	56	107	165
	1500	3041	10031	4540	711	378	**71**	2787		469	266	**53**	70	130	296
	2000	5882	10083	8423	1814	897	**134**	6418		396	254	**71**	95	172	443
	2500	5726	10070	10000	4583	2222	**183**	9468		385	176	**88**	136	181	439

Table 1: Results for instance sets from [8] consisting of five complete graphs with 41 nodes, $T = V \setminus \{s\}$, different graph structures (r, c, e), delay ranges (100, 1000), and bounds B.

		$\overline{\text{CPU time [s]}}$					$\overline{\text{Iterations}}$				
$\frac{\lvert T\rvert}{\lvert V\setminus\{s\}\rvert}$	B	π^*	$\bar{\pi}$	$\hat{\pi}$	$\tilde{\pi}$	PL	π^*	$\bar{\pi}$	$\hat{\pi}$	$\tilde{\pi}$	PL
0.3	30	19	**6**	**6**	10	36	143	**30**	36	92	84
	50	139	**15**	16	23	55	413	**41**	50	124	102
	100	2849	97	89	**55**	509	1345	**44**	55	149	194
0.7	30	77	**29**	**29**	34	171	142	**32**	46	93	198
	50	727	112	107	**80**	1091	561	**51**	62	130	475
	100	7942	819	923	**253**	7557	1361	**79**	92	182	958
1.0	30	213	77	**62**	67	630	184	**34**	54	98	797
	50	1807	302	328	**172**	5769	614	**56**	81	142	2039
	100	9615	2615	2196	**837**	10000	851	**86**	123	214	694

Table 2: Results for 30 randomly generated complete graphs with $|V| = 100$, different sets of terminal nodes, delays and costs uniformly distributed in [1,99] and delay bounds B.

We conclude that all stabilization methods based on alternative dual-optimal solutions lead to an enormous reduction of the necessary CPU-time. While $\hat{\pi}$ performs best for easier instances, $\tilde{\pi}$ clearly outperforms all other approaches on harder instances, i.e. on those which generally need more time. Stabilization based on piecewise linear penalty functions outperforms unstabilized CG in the majority of cases, but is clearly not competitive to our three approaches based on alternative dual-optimal solutions. We further observe that our unstabilized CG variant needs significantly less iterations than the conceptually identical one discussed by Gouveia et al. [8]. We believe that next to a different CPLEX version, these differences are mainly based on choosing a better set of initial path variables, more sophisticated graph preprocessing, and the fact that we use the dual simplex algorithm which turned out to perform better than the primal one in our case. Comparing the relative computational times of the Lagrangian approach from [8] to their CG approach with the speed-up achieved by our stabilization methods, we conclude that the proposed stabilized CG method also outperforms this method. All approaches based on dual-optimal solutions terminated before the time limit was met in all but one of the experiments reported in Table 1, while both unstabilized CG variants and the piecewise linear stabilization approach failed to do so for a number of experiments.

6. CONCLUSIONS & FUTURE WORK

In this paper we showed how to significantly accelerate a column generation approach based on a path formulation for the RDC-STP using alternative dual-optimal solutions in the pricing sub-problem. We conclude that this method does further outperform a stabilization method based on piecewise linear penalty functions as well as a previously presented approach based on Lagrangian relaxation [8]. We are currently extending the presented stabilized column generation towards a branch-and-price approach in order to compute proven optimal solutions to medium sized instances of the RDCSTP. In future, we also plan to consider additional pricing strategies – e.g. by restricting the total number of path variables to be included in each pricing iteration – and want to compare our approach to further stabilization techniques such as e.g. interior point stabilization [15]. Finally, we also want to study the impact of choosing better initial columns computed by metaheuristics which may lead to further significant speed-up as well as implement the Lagrangian relaxation approach from [8] for a fair comparison.

7. REFERENCES

[1] V. P. Kompella, J. C. Pasquale, and G. C. Polyzos, "Multicasting for multimedia applications," in *INFOCOM '92. Eleventh Annual Joint Conference of the IEEE Computer and Communications Societies, IEEE*, 1992, pp. 2078–2085.

[2] N. Skorin-Kapov and M. Kos, "A GRASP heuristic for the delay-constrained multicast routing problem," *Telecommunication Systems*, vol. 32, no. 1, pp. 55–69, 2006.

[3] Y. Xu and R. Qu, "A GRASP approach for the delay-constrained multicast routing problem," in *Proceedings of the 4th Multidisplinary International Scheduling Conference (MISTA4)*, Dublin, Ireland, 2009, pp. 93–104.

[4] N. Ghaboosi and A. T. Haghighat, "A path relinking approach for delay-constrained least-cost multicast routing problem," in *19th IEEE International Conference on Tools with Artificial Intelligence*, 2007, pp. 383–390.

[5] M. Ruthmair and G. R. Raidl, "A kruskal-based heuristic for the rooted delay-constrained minimum spanning tree problem," in *EUROCAST 2009*, ser. LNCS, R. Moreno-Díaz *et al.*, Eds., vol. 5717. Springer, 2009, pp. 713–720.

[6] ——, "Variable neighborhood search and ant colony optimization for the rooted delay-constrained minimum spanning tree problem," in *PPSN XI, Part II*, ser. LNCS, R. Schaefer *et al.*, Eds., vol. 6239. Springer, 2010, pp. 391–400.

[7] V. Leggieri, M. Haouari, and C. Triki, "An exact algorithm for the Steiner tree problem with delays," *Electronic Notes in Discrete Mathematics*, vol. 36, pp. 223–230, 2010.

[8] L. Gouveia, A. Paias, and D. Sharma, "Modeling and solving the rooted distance-constrained minimum spanning tree problem," *Computers & Operations Research*, vol. 35, no. 2, pp. 600–613, 2008.

[9] C. Barnhart, E. L. Johnson, G. L. Nemhauser, M. W. P. Savelsbergh, and P. H. Vance, "Branch-and-price: Column generation for solving huge integer programs," *Operations Research*, vol. 46, pp. 316–329, 1998.

[10] G. Desaulniers, J. Desrosiers, and M. M. Solomon, Eds., *Column Generation*. Springer, 2005.

[11] F. Vanderbeck, "Implementing mixed integer column generation," in *Column Generation*, G. Desaulniers, J. Desrosiers, and M. M. Solomon, Eds. Springer, 2005, pp. 331–358.

[12] H. B. Amor, J. Desrosiers, and J. M. V. Carvalho, "Dual-optimal inequalities for stabilized column generation," *Operations Research*, vol. 54, no. 3, pp. 454–463, 2006.

[13] J. M. V. de Carvalho, "Using extra dual cuts to accelerate convergence in column generation," *INFORMS Journal on Computing*, vol. 17, no. 2, pp. 175–182, 2005.

[14] H. B. Amor and J. Desrosiers, "A proximal trust-region algorithm for column generation stabilization," *Computers & Operations Research*, vol. 33, pp. 910–927, 2006.

[15] L.-M. Rousseau, M. Gendreau, and D. Feillet, "Interior point stabilization for column generation," *Operations Research Letters*, vol. 35, no. 5, pp. 660–668, 2007.

[16] M. Leitner, G. R. Raidl, and U. Pferschy, "Accelerating column generation for a survivable network design problem," in *Proceedings of the International Network Optimization Conference 2009*, M. G. Scutellà *et al.*, Eds., Pisa, Italy, 2009.

[17] M. Leitner and G. R. Raidl, "Strong lower bounds for a survivable network design problem," in *ISCO 2010*, ser. Electronic Notes in Discrete Mathematics, M. Haouari and A. R. Mahjoub, Eds., vol. 36. Elsevier, 2010, pp. 295–302.

[18] M. Leitner, G. R. Raidl, and U. Pferschy, "Branch-and-price for a survivable network design problem," Vienna University of Technology, Vienna, Austria, Tech. Rep. TR 186–1–10–02, 2010, submitted to Networks.

[19] H. B. Amor, J. Desrosiers, and A. Frangioni, "On the choice of explicit stabilizing terms in column generation," *Discrete Applied Mathematics*, vol. 157, pp. 1167–1184, 2009.

Heuristics for Discrete Power Control - A Case-Study in Multi-Carrier DSL Networks

Martin Wolkerstorfer [*] Tomas Nordström [‡] [†]

[*] Telecommunications Research Center Vienna (FTW)
Donau-City-Straße 1, A-1220 Vienna, Austria
{wolkerstorfer, nordstrom}@ftw.at

ABSTRACT

The performance of multi-user digital subscriber line (DSL) networks is limited by the electro-magnetic coupling between twisted pair cables. The adverse effect of this coupling can be reduced by controlling the transmit powers of all lines. The corresponding multi-user, multi-carrier power control problem can be modeled as a multi-dimensional nonlinear Knapsack problem which has previously motivated the application of various mathematical decomposition methods. These methods decompose the problem into a large number of combinatorial per-subcarrier problems. Our main contribution lies in the proposal and analysis of various low-complexity heuristics for these combinatorial problems. We provide insights in the parameter setting as well as simulation results on a large set of 6 and 30-user DSL scenarios. These show that simple randomized greedy heuristics perform well even in case of a very stringent complexity budget and that the heuristics' average suboptimality is dependent on the targeted data-rate.

Keywords: Power Control, DSL, Metaheuristics, Column Generation

1. MOTIVATION

Digital subscriber lines (DSL) are the most widely deployed broadband access technology today, with more than 320 million customers world-wide in 2010 [1]. DSL systems suffer from the electro-magnetic coupling between the twisted pair cables which induces so called "far-end crosstalk" noise at the DSL receivers. This in turn is the main limiting factor for the data-rate performance of current DSL modems. Furthermore, today's DSL systems are based on discrete multi-tone (DMT) modulation which splits the available frequency bandwidth into independent subchannels ("subcarriers"). We consider the problem of optimally controlling the transmitted power levels on each of these subchannels and hence the crosstalk noise as well as the conveyable total data-rate. This problem is also fundamental in multi-carrier wireless networks [2]. The classical objective is the maximization of a weighted sum of data-rates. Recently this technique has also been discovered useful for reducing the system power consumption in DSL [3, 4]. Current state-of-the-art multi-carrier power control algorithms for tens of subscriber lines are based on techniques such as user-iterative power updates, dual relaxation of transmission rate and sum-power constraints, as well as successive continuous and convex approximation, cf. [3, 5, 6] and the references therein. Dual relaxation results in the independent optimization of a large number of per-subcarrier problems. The distinctive feature of the nonlinear Dantzig-Wolfe decomposition [7, Ch. 23] based scheme in [8] is that it allows for the suboptimal solution of the independent per-subcarrier problems.

Our main contribution is the proposal of various heuristics for complexity reduction of solving the combinatorial per-subcarrier optimization subproblems, thereby expanding upon the work in [8]. We begin in Section 2 by reviewing the optimization problem of controlling the transmit power in DSL. In Section 3 we then turn to the main focus of this paper, namely the combinatorial per-subcarrier problems and various heuristics for their solution. Section 4 gives an example of the heuristics' performance when applied in conjunction with the framework in [8] to solve the main problem from Section 2. Our conclusions are summarized in Section 5.

This work has been supported in parts by the Austrian Government and the City of Vienna within the competence center program COMET.

2. BACKGROUND - GLOBAL PROBLEM

We denote the index sets of users and subcarriers by $\mathscr{U} = \{1,\dots,U\}$ and $\mathscr{C} = \{1,\dots,C\}$, respectively, where U and C are the total number of users and subcarriers, respectively. The optimization variables are the power levels p_u^c of user u on subcarrier c, where we will compactly write $\mathbf{p}^c \in \mathscr{R}_+^U$ for the power allocation of all users on subcarrier c. The data-rate of user u on subcarrier c is a nonlinear function $r_u^c(\mathbf{p}^c)$ [9] which notably depends on the power allocation of all users on that subcarrier. Again, we will compactly write $\mathbf{r}^c(\mathbf{p}^c) \in \mathscr{R}^U$ to denote all users' rates on subcarrier c. Reversely, the power allocation $\mathbf{p}^c(\mathbf{r}^c)$ for rates $\mathbf{r}^c$ can be computed as the unique [10] solution of a system of linear equations of size $U \times U$. Power levels are constrained by a regulatory power mask constraint $p_u^c \le \hat{p}_u^c$ and the implicit constraint $r_u^c(\mathbf{p}^c) \in \mathscr{B}, \forall u \in \mathscr{U}, c \in \mathscr{C}$, motivated by practical modulation schemes which only support a discrete set of data-rates $\mathscr{B}$. Altogether we may compactly write the set of feasible power allocations on subcarrier c as

$$\mathscr{Q}^c = \{\mathbf{p}^c | r_u^c(\mathbf{p}^c) \in \mathscr{B},\ 0 \le p_u^c \le \hat{p}_u^c, \forall u \in \mathscr{U}\}. \tag{1}$$

Additional to these per-subcarrier constraints the U users have minimum target-rates $\mathbf{R} \in \mathscr{R}_+^U$ dependent on the accepted service level, as well as technology-dependent maximum sum-power levels $\mathbf{P} \in \mathscr{R}_+^U$. Our optimization objective is defined as the sum of per-subcarrier objectives $\bar{f}_c(\mathbf{p}^c, \hat{\mathbf{w}}, \check{\mathbf{w}}), c \in \mathscr{C}$. These are given as the following weighted sum of users' transmit powers and rates

$$\bar{f}_c(\mathbf{p}^c, \hat{\mathbf{w}}, \check{\mathbf{w}}) = \hat{\mathbf{w}}^\mathsf{T}\mathbf{p}^c - \check{\mathbf{w}}^\mathsf{T}\mathbf{r}^c(\mathbf{p}^c),\ \forall c \in \mathscr{C}, \tag{2}$$

where the weights $\hat{\mathbf{w}}, \check{\mathbf{w}} \in \mathscr{R}_+^U$ allow us to trade-off between rate and energy optimization, i.e., we can consider rate-maximization and energy-minimization as special cases. We are now ready to formally write the optimization problem for multi-user power control in DSL as the following multi-dimensional nonlinear Knap-

sack problem [11]

$$\underset{\mathbf{p}^c \in \mathscr{D}^c, \forall c \in \mathscr{C}}{\text{minimize}} \sum_{c \in \mathscr{C}} \bar{f}_c(\mathbf{p}^c, \hat{\mathbf{w}}, \check{\mathbf{w}}) \tag{3a}$$

$$\text{subject to} \sum_{c \in \mathscr{C}} \mathbf{r}^c(\mathbf{p}^c) \succeq \mathbf{R}, \tag{3b}$$

$$\sum_{c \in \mathscr{C}} \mathbf{p}^c \preceq \mathbf{P}. \tag{3c}$$

3. THE COMBINATORIAL SUBPROBLEM

3.1. Subproblem Formulation

After Lagrange relaxation of constraints (3b) and (3c), respectively, one faces for each subcarrier $c \in \mathscr{C}$ an independent, nonlinear (and non-convex), wide-sense combinatorial [12, Sec. 4.4] pricing subproblem in the form of [8]

$$\underset{\{\mathbf{r}^c | \mathbf{p}^c(\mathbf{r}) \in \mathscr{D}^c\}}{\text{minimize}} f_c(\mathbf{r}^c) = \bar{f}_c(\mathbf{p}^c(\mathbf{r}^c), \hat{\mathbf{w}} + \boldsymbol{\nu}, \check{\mathbf{w}} + \boldsymbol{\lambda}), \tag{4}$$

where $\boldsymbol{\nu}, \boldsymbol{\lambda} \in \mathscr{R}_+^U$ are the Lagrange multipliers associated with constraints (3b) and (3c), respectively, cf. [8] for details. Note that for reasons of algorithm design we use the discrete vector $\mathbf{r}^c$ as our variables instead of the uniquely coupled power allocation $\mathbf{p}^c$ used above. In the following sections we study the solution of the subproblem in (4) and therefore omit subcarrier indices c for ease of notation. The search space we consider is $\times_{u \in \mathscr{U}} \mathscr{B}$, i.e., we only search over discrete rate allocations and do not explicitly consider the constraints in (4). An allocation $\mathbf{r}$ violating these constraints has by definition an objective $f(\mathbf{r}) = \infty$ and our algorithms thereby never traverse infeasible allocations. As mentioned above, in order to evaluate the objective $f(\mathbf{r})$ and to determine feasibility we need to solve a linear system of equations, cf. (1) and (4). Later we will introduce the number of evaluations of $\mathbf{p}(\mathbf{r})$ as a reproducible complexity measure to compare different algorithms.

The optimal solution of the problem in (4) was shown to have polynomial complexity in [8]. However, obtaining optimal solutions for practical values of U was found intractable for conventional branch-and-bound schemes [8, 13]. Furthermore, the number of these per-subcarrier problems is in the order of thousands in the newest generations of DSL technology. This altogether motivates our work on fast heuristics in the following sections.

3.2. Constructive Greedy Base Heuristics

In the full paper we review the greedy base heuristic as well as the sequential greedy heuristic in [8] and provide an analysis of various 6-user VDSL scenarios, cf. Section 3.5 for simulation details. This analysis shows that the suboptimality of the base heuristic is zero for all collocated network scenarios while the highest suboptimality appears in classical near-far type of scenarios. This insight will guide the parameter settings of randomized heuristics below. Basically two approaches will be taken in the following to improve upon purely greedy schemes, namely a) a randomization of greedy decisions, and/or b) randomized local searches.

3.3. Local Search

Local search schemes aim at iteratively *improving* a given solution $\mathbf{r}$. Their key ingredient is the definition of a neighborhood $\mathscr{N}(\mathbf{r}) \subseteq \times_{u \in \mathscr{U}} \mathscr{B}$ around $\mathbf{r}$ from which a next candidate allocation is picked, cf. [14] for various examples of local search schemes. Here we restrict ourselves to two possible neighborhood definitions: The first is a simple one-step neighborhood

$$\mathscr{N}^{(1)}(\mathbf{r}) = \{\tilde{\mathbf{r}} \in \times_{u \in \mathscr{U}} \mathscr{B} \mid \tilde{r}_u = r_u \pm \Delta, \tilde{r}_i = r_i, \forall i \in \mathscr{U} \setminus \{u\}, u \in \mathscr{U}\}, \tag{5}$$

which contains all allocations $\tilde{\mathbf{r}}$ that can be reached by perturbing a single element of $\mathbf{r}$ by Δ. The second used neighborhood is

$$\mathscr{N}^{(2)}(\mathbf{r}) = \mathscr{N}^{(1)}(\mathbf{r}) \cup \bar{\mathscr{N}}^{(2)}(\mathbf{r}), \tag{6}$$

$$\bar{\mathscr{N}}^{(2)}(\mathbf{r}) = \{\tilde{\mathbf{r}} \in \times_{u \in \mathscr{U}} \mathscr{B} \mid \tilde{r}_u = r_u \pm \Delta, \tilde{r}_{\bar{u}} = r_{\bar{u}} \pm \Delta, \tilde{r}_i = r_i, \forall i \in \mathscr{U} \setminus \{u, \bar{u}\}, u \neq \bar{u}, u, \bar{u} \in \mathscr{U}\}, \tag{7}$$

which contains all allocations $\tilde{\mathbf{r}}$ that can be reached by perturbing at most two different elements of $\mathbf{r}$ by Δ. Furthermore, two neighborhood search strategies are considered, namely the "first-improving" and the "best-improving" search strategy, cf. [14, Ch. 8].

Name	Abbr.	Reference
Joint Greedy Optimization	JOGO	[8]
Sequential Greedy Optimization	SEGO	[8]
Local Search	LS	Section 3.3
Rollout Algorithm	RA	[15]
Greedy Rand. Adapt. Search Proc.	GRASP	[14, Ch. 8]
Iterated Local Search	ILS	[14, Ch. 11]
Simulated Annealing	SA	[14, Ch. 10]
Ant Colony System	ACS	[16]
Randomized SEGO	rSEGO	Section 3.4
Randomized LS	rLS	Section 3.4
Solver "Couenne"	COU	[17]
Optimal Branch-and-Bound	OPT	[8]

Table 1: *Heuristics compared on the problem in (4).*

3.4. Heuristics Inspired by Meta-Heuristics

In the full paper we will present detailed descriptions of various heuristics for the bit-loading problem in (4) which are partly inspired by well-known meta-heuristics, cf. the overview of all studied algorithms in Table 1. Rollout algorithms and rSEGO/ant colony system algorithms are deterministic and randomized sequential decision making algorithms, respectively. GRASP is an extension of the greedy base heuristic using randomization, while iterated local search, randomized local search, as well as simulated annealing are randomized local search schemes.

3.5. Methodology, Simulations and Discussion

In order to be able to compare to optimal schemes as in [8] we restrict ourselves in this section to $U = 6$ users. We construct our network scenarios using a set of specified line lengths $\mathscr{L} = \{200, 400, 600, 800\}$ m, considering all U-combinations with repetitions. For example, for $U = 6$ this results in

$$m = \binom{|\mathscr{L}| + U - 1}{U} = 84, \tag{8}$$

generated network scenarios. Note that this allows us to identify scenarios where the given algorithms perform badly. Such scenarios were used to initially set the algorithmic parameters. Based on these settings various parameter changes were selected and the impact on the average performance studied by Monte-Carlo simulation. As in [8] we use equal Lagrange multipliers λ_u, ν_u, for all $u \in \mathscr{U}$. For setting the parameters of the heuristics we chose Lagrange multipliers $\lambda_u = 1, \nu_u = 0$ and weights $\check{w}_u = 0, \hat{w}_u = 1/U$, which leads to a maximum sum-rate in our 6-user scenarios [8].

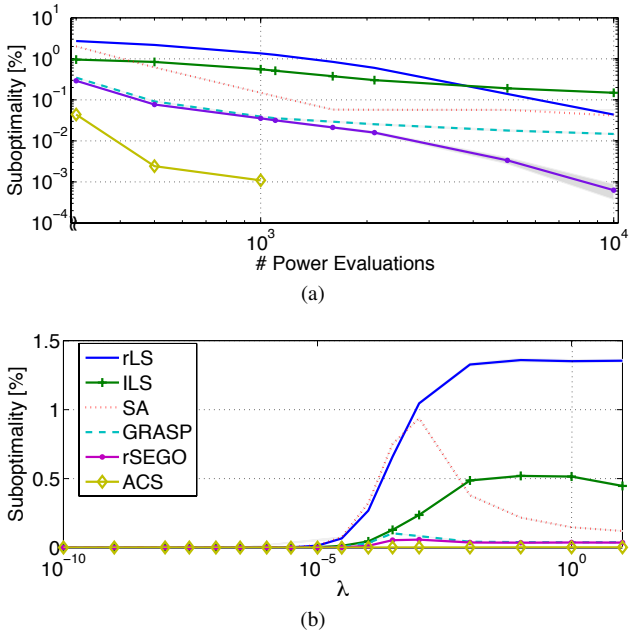


Figure 1: *Average suboptimality of randomized heuristics in* 6*-user VDSL scenarios; a) Dependency on the complexity budget; b) Dependency on Lagrange multipliers* $\lambda_u = \lambda, \forall u \in \mathscr{U}$.

We further make the practical assumption that there is a restriction in simulation time for solving the subproblems in (4). However, in order to make our results reproducible we will use the number of power evaluations $\mathbf{p}(\mathbf{r})$ by solving a linear system of equations [9] as the stopping criterion of all considered heuristics. Note that this metric also includes evaluations of infeasible allocations, cf. the discussion in Section 3.1. Simulation results provided in the full paper further motivate this complexity metric as it was found to preserve the comparability among different heuristics. While for six users we were able to compute the optimum of the problem in (4), for a higher number of users we either compare to the greedy base heuristic JOGO due to its simplicity, or to a lower bound being the optimal objective of a discrete, convex problem relaxation, cf. [8, Alg. 5] for an analytic solution with $O(U)$ complexity. We find that this lower bound gives a low gap to the optimal objective when the Lagrange multipliers $\boldsymbol{\lambda}$ (and therefore the users' rates) are low. Simulations were carried out using our DSL simulator available in [18] and using common parameters as in [8].

We investigated the solution quality of all presented heuristics for solving the subproblems in (4) in a VDSL system with 1635 subcarriers, where for the comparisons in this section we only select a subset of subcarriers $\mathscr{C} = 1, 51, \ldots, 1601$. As a benchmark for all our algorithms we use "Couenne" [17], a free branch-and-bound based solver for non-convex mixed-integer problems. As a base-line for our stochastic heuristics we added a randomized local search (rLS) scheme where the LS algorithm is reinitialized at random starting points $\mathbf{r}$ uniformly drawn from $\times_{u\in\mathscr{U}} \mathscr{B}$. In the full paper we provide the specific parameter settings and the intuitions behind these settings for all heuristics described in Section 3.4. Figure 1(a) depicts the average suboptimality of all randomized heuristics as a function of the complexity budget in various 6-user VDSL network scenarios. Intuitively, allowing the algorithms to test more solutions leads on average to a better performance. ACS performs best in these test scenarios, where its curve stops at 10^3 as it is optimal on the simulated points beyond that. Note that rLS eventually performs better than ILS and SA, which hints at insufficient diversification capabilities of these two schemes. Figure 1(b) similarly shows, for fixed complexity budget of 10^3 evaluations, the dependency of the heuristics' average suboptimality on

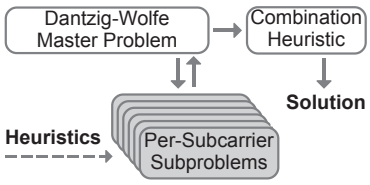


Figure 2: *Framework [8] for applying heuristics in DSL.*

the Lagrange multipliers $\lambda_u = \lambda, \forall u \in \mathscr{U}$, and hence on the targeted transmission rate as the average rate per user increases with these multipliers. JOGO, which is used as an initial incumbent for all schemes, was found to have a monotonously increasing suboptimality with λ. Also, the optimal rates do not change in most scenarios above $\lambda = 10^{-2}$. Differently to JOGO, all heuristics show a peak suboptimality for a specific multiplier value, however, at different values for different heuristics. Intuitively this can be explained by the fact that with increasing λ what matters most is the total number of bits achieved by all users. Then it matters less *how* the bits are distributed among the users as this distribution only influences the power consumption which has a comparably low weight in the objective for high λ. In 30-user VDSL scenarios with a complexity limit of $2 \cdot 10^4$ power evaluations per subcarrier problem and using the same parameter settings for all algorithms as above the picture is very different. The randomized heuristics GRASP, rSEGO and ILS perform now best, with an improvement upon the objective values achieved by the greedy base heuristic by on average 9.9 %, 9.8 %, and 9.2 %, respectively. Note however that the simple *deterministic* extension of the greedy constructive heuristic by a two-step local search improves the greedy heuristic already by on average 8 % while taking on average only $0.4 \cdot 10^4$ power evaluations. Notably, the maximum improvement in sum-objective over all 33 tested subcarriers encountered in any tested network scenario was as high as 32 %. Further insights in the performance of all heuristics in Table 1 for 6 and 30-user VDSL scenarios will be given in the full version of this paper.

4. PERFORMANCE EVALUATION FOR DSL

The purpose of this section is to provide evidence of the practical usefulness of the proposed approach based on heuristics. As these only target the subproblems in (4) we exemplarily apply the heuristic rSEGO in conjunction with the complexity reduction technique in [19] and the column generation framework in [8], cf. Figure 2. The algorithm consisting of these techniques is compared to state-of-the-art multi-carrier power control algorithms [5, 20]. When using the dual relaxation based, iterative spectrum balancing algorithm (ISB) in [5] we subsequently use the greedy central discrete bit-loading algorithm (CDBL) in [20] to obtain a discrete feasible solution. As an example of the DSL performance we consider a sum-rate maximization problem in a near-far downstream scenario with 50 collocated users, where 40 lines connect to the central office at a distance of 800 m and 10 lines connect to a closer remote cabinet at 200 m distance. Compared to CDBL we obtain a 2.1 % sum-rate increase, or more importantly an 8.3 % sum-rate increase for the lines connected to the central office. Comparing to ISB our results show an 8.2 % increase in total sum-rate and a 12.3 % increase in sum-rate for the lines connected to the central office. The full paper will provide simulation settings and extensive average performance and complexity comparisons.

5. CONCLUSIONS

We studied the application of various heuristics to a combinatorial, non-convex power allocation problem in digital subscriber lines (DSL). Parameter setting for various 6−user DSL scenarios allowed to obtain near-optimal results using several of the proposed randomized heuristics. Under various 30−user scenarios extending the greedy constructive heuristic by a proposed local search scheme gave already substantial improvements at low complexity. Randomized heuristics still gave slight improvements beyond that for moderate complexity limits. Summarizing, the proposed heuristics have shown an average gain in objective value compared to the greedy constructive heuristic of up to 10 %.

6. ACKNOWLEDGEMENTS

This work has been supported in parts by the Austrian Government and the City of Vienna within the competence center program COMET.

7. REFERENCES

[1] F. Vanier, "World broadband statistics: Short report Q3 2010," Point Topic Ltd., Tech. Rep., December 2010.

[2] C. Wong, R. Cheng, K. Letaief, and R. Murch, "Multiuser OFDM with adaptive subcarrier, bit, and power allocation," *IEEE Journal on Selected Areas in Communications*, vol. 17, no. 10, pp. 1747–1758, October 1999.

[3] M. Wolkerstorfer, D. Statovci, and T. Nordström, "Dynamic spectrum management for energy-efficient transmission in DSL," in *IEEE ICCS 2008*, Guangzhou, China, 19–21 November 2008.

[4] M. Guenach, C. Nuzman, K. Hooghe, J. Maes, and M. Peeters, "Reduced dimensional power optimization using class AB and G line drivers in DSL," in *IEEE GLOBECOM Workshops*, Miami, USA, 6–10 December 2010, pp. 1443–1447.

[5] W. Yu and R. Lui, "Dual methods for nonconvex spectrum optimization of multicarrier systems," *IEEE Transactions on Communications*, vol. 54, no. 7, pp. 1310–1322, July 2006.

[6] P. Tsiaflakis, M. Diehl, and M. Moonen, "Distributed spectrum management algorithms for multiuser DSL networks," *IEEE Transactions on Signal Processing*, vol. 56, no. 10, pp. 4825–4843, October 2008.

[7] G. Dantzig, *Linear programming and extensions*. Princeton University Press, 1963.

[8] M. Wolkerstorfer, J. Jaldén, and T. Nordström, "Complexity reduction strategies for the discrete multi-carrier power control problem," *Submitted to IEEE Transactions on Signal Processing*, January 2011.

[9] P. Golden, H. Dedieu, and K. Jacobsen, Eds., *Fundamentals of DSL technology*. Auerbach Publications, 2006.

[10] R. Yates, "A framework for uplink power control in cellular radio systems," *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 7, pp. 1341–1347, September 1995.

[11] T. Morin and R. Marsten, "Branch-and-bound strategies for dynamic programming," *Operations Research*, vol. 24, no. 4, pp. 611–627, July-August 1976.

[12] K. Price, R. Storn, and J. Lampinen, *Differential Evolution*. Springer, 2005.

[13] P. Tsiaflakis, J. Vangorp, M. Moonen, and J. Verlinden, "A low complexity optimal spectrum balancing algorithm for digital subscriber lines," *Signal Processing*, vol. 87, no. 7, pp. 1735–1753, July 2007.

[14] F. Glover and G. Kochenberger, Eds., *Handbook of Metaheuristics*. Kluwer Academic Publishers, 2003.

[15] D. Bertsekas, "Rollout algorithms for discrete optimization: A survey," in *Handbook of Combinatorial Optimization*, D. Zu and P. Pardalos, Eds. Springer, August 2010, to appear.

[16] M. Dorigo and L. Gambardella, "Ant colony system: A cooperative learning approach to the traveling salesman problem," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 53–66, April 1997.

[17] P. Belotti, "Couenne: A user's manual," Lehigh University, Tech. Rep., 2009.

[18] (2008, October) xDSL simulator v3.1. [Online]. Available: xdsl.ftw.at

[19] M. Wolkerstorfer and T. Nordström, "Coverage optimization in DSL networks by low-complexity discrete spectrum balancing," submitted to *IEEE GLOBECOM 2011*, Houston, Texas, USA, 5–9 December 2011.

[20] J. Lee, R. Sonalkar, and J. Cioffi, "Multi-user discrete bit-loading for DMT-based DSL systems," in *IEEE GLOBECOM 2002*, vol. 2, Taipei, Taiwan, China, 17–21 November 2002, pp. 1259–1263.

A HYBRID META-HEURISTIC FOR THE NETWORK LOAD BALANCING PROBLEM

Dorabella Santos [*] Amaro de Sousa [†] Filipe Alvelos [‡]

[*] Instituto de Telecomunicações
3810-193 Aveiro, Portugal
dorabella@av.it.pt

[†] Instituto de Telecomunicações, Universidade de Aveiro
3810-193 Aveiro, Portugal
asou@ua.pt

[‡] Centro Algoritmi / DPS, Universidade do Minho
4710-057 Braga, Portugal
falvelos@dps.uminho.pt

ABSTRACT

Given a capacitated telecommunications network with single path routing and an estimated traffic demand matrix, the network load balancing problem is the determination of a routing path for each traffic commodity such that the network load balancing is optimized, i.e., the worst case link load is minimized, among all such solutions, the second worst case link load is minimized, and so on... We discuss a meta-heuristic which runs a GRASP with Path Relinking procedure on a restricted search space defined by Column Generation. We discuss some computational results showing that, for the network load balancing problem, this approach is successful in obtaining good quality solutions in short running times.

Keywords: Load Balancing, GRASP with Path Relinking, Column Generation, Hybrid Meta-Heuristics

1. META-HEURISTIC PROPOSAL

Greedy Randomized Adaptive Search Procedure (GRASP) is a meta-heuristic first introduced for the set covering problem [1]. It is a multi-start local search method where, at each start, a solution is randomly generated (with some greediness) and local search is applied to it to find its closest local minimum solution (in a minimization problem). Path Relinking (PR), originally proposed as an intensification method applied to tabu search [2], is a method that tries to find better solutions by the combination of two initial solutions. The common combination of GRASP with PR (GRASP+PR) is to run PR at the end of each GRASP iteration between its local minimum solution and one solution randomly selected from a given list of elite solutions (please see [3] for a survey on many applications where GRASP with Path Relinking has been applied). The key idea of our approach is to run GRASP+PR on a restricted search space instead of running it on the complete solution space of the problem. The restricted search space aims to make the search more efficient provided that it contains good quality solutions. To manage the restricted search space, we use Column Generation (CG): the initial restricted search space is composed by the columns generated by CG solving the LP relaxation of the original problem; then, during the search, the restricted search space is modified by including new columns and/or excluding existing columns. New columns are generated by CG solving a perturbed problem which is defined based on the current incumbent solution of the GRASP+PR and on the LP value of the original problem. This meta-heuristic may be seen as an implementation of the general framework for combining CG and meta-heuristics entitled SearchCol (Meta-heuristic search by column generation) [4].

2. NETWORK LOAD BALANCING OPTIMIZATION

Consider a telecommunications network modeled on a graph $G(N,A)$ where N is the set of network nodes and A is the set of network links connecting nodes. The link between nodes $i \in N$ and $j \in N$ is denoted by $\{i,j\}$ and each link $\{i,j\} \in A$ has a given capacity $c_{\{ij\}}$. Consider a set of commodities K, where each commodity $k \in K$ is to be routed through a single path on the network and is characterized by its origin node $o_k \in N$, its destination node $d_k \in N$ and its demand $b_k > 0$.

Let P_k be the set of paths available on graph G between the end nodes of $k \in K$ and let $\delta^{pk}_{\{ij\}}$ be a binary parameter that is 1 if link $\{i,j\} \in A$ is in the path $p \in P_k$. To model the optimization problem, we consider the following decision variables: the binary variables φ^p_k which are 1 if path $p \in P_k$ is chosen as the routing path of commodity $k \in K$; and the real variables $\mu_{\{ij\}}$ accounting for the load on link $\{i,j\} \in A$. The following set of constraints defines the complete solution space:

$$lll \sum_{p \in P_k} \varphi^p_k = 1 \qquad \forall k \in K \tag{1}$$

$$\sum_{k \in K} \sum_{p \in P_k} b_k \delta^{pk}_{\{ij\}} \varphi^p_k = c_{\{ij\}} \mu_{\{ij\}} \qquad \forall \{i,j\} \in A \tag{2}$$

$$\phi^p_k \in \{0,1\}, \mu_{\{ij\}} \in [0,1] \tag{3}$$

Constraints (1) guarantee that exactly one path of P_k is chosen for each $k \in K$, constraints (2) account for the loads on each link, and constraints (3) are the domain constraints. The load balancing optimization problem uses the concept of lexicographical optimization. Given two vectors $a = (a_1, ..., a_m)$ and $b = (b_1, ..., b_m)$, vector a is lexicographically is said to be smaller than vector b if either $a_1 < b_1$ or there exists an index $l \in \{1, ..., m-1\}$ such that $a_i = b_i$ for all $i \in l$ and $a_l + 1 < b_l + 1$. Now consider the vector of link loads $\mu = (\mu_{\{ij\}} : \{i,j\} \in A)$ and let $[\mu]$ be the vector obtained from by rearranging its elements in a non-increasing order. The load balancing optimization problem can be defined in a non-linear

manner as:

$$lll\textbf{lexmin}\ [\mu] \quad (4)$$
$$\text{Subject to}$$
$$(1)-(3)$$

where **lexmin** denotes the lexicographical minimization of $[\mu]$, i.e., finding a vector $[\mu^*]$ which is lexicographically minimal among all possible vectors $[\mu]$. It is known that the solution of the load balancing optimization problem can be obtained by solving a sequence of mixed integer linear programming models. One such method is the conditional means approach (for details, please see [5-6]). The use of CG to solve the LP relaxation if this sequence of models is straightforward since constraints (1) define a subproblem for each commodity $k \in K$ whose solutions are paths. For this reason, CG is usually named Path Generation and the columns generated by CG for the subproblem associated to commodity k represent paths between its end nodes o_k and d_k in graph G.

3. EFFICIENCY OF THE PROPOSED META-HEURISTIC

The network load balancing problem was addressed in [7], where GRASP+PR was applied to the restricted search space given by the columns generated by CG solving the LP relaxation of the problem. In that approach, the restricted search space is not modified during the search and PR is done only between the local minimum solution of each GRASP iteration and the incumbent solution (i.e, the elite list is composed by a single solution). That approach was compared to the equivalent GRASP+PR applied to the complete solution space of the load balancing problem. The computational results showed that the constrained search space gives much better results because it contains good quality solutions and, due to its size, enables to find them in much shorter running times.

In order to test our meta-heuristic, we have defined a set of 24 instances based on the well known network topology of the NSF network with 26 nodes and 42 links. In all test instances, we have randomly generated a demand matrix with the aim of emulating different possible real scenarios.

In terms of PR, the computational results show that: (i) the use of an elite list does not provide significant improvements when PR is done only with one randomly selected elite solution; (ii) there is a significant improvement if PR is done to each of all elite solutions provided that the list size is not too large and the elite solutions are quite different; (iii) forward and backward PR was the best strategy in terms of heuristic efficiency.

Since the proposed method is a stochastic process, it gives different solutions in different runs. In order to generate useful data for comparison analysis, we have adopted the following methodology. Whenever we aim to compare two algorithms, we run both algorithms 10 times, giving the same runtime to each, and compare each pair of solutions. Then, we sum the number of times the second algorithm was better than the first algorithm. We average these numbers over the set of test instances of interest and calculate, in percentage, how many runs produce better results with the second algorithm when compared with the first one.

Globally, the best algorithm exhibits an efficiency performance improvement of 94.1% when compared to running GRASP+PR with no restricted search space modifications, showing that our meta-heuristic is far better than the simplest one exploited in [7].

4. REFERENCES

[1] T. Feo and M. Resende,“A probabilistic heuristic for a computationally difficult set covering problem", Operations Research Letters, 8, pp. 67-71, 1989

[2] F. Glover, “Tabu search and adaptive memory programming - Advances, applications and challenges", in Interfaces in Computer Science and Operations Research, R.S. Barr, R.V. Helgason, and J.L. Kennington, (Eds.), Kluwer, pp. 1-75, 1996

[3] M. Resende and C. Ribeiro, “GRASP with path relinking: recent advances and applications", in Metaheuristics: Progress as Real Problem Solvers, T. Ibaraki, K. Nonobe and M. Yagiura, (Eds.), Springer, pp. 29-63, 2005

[4] F. Alvelos, A. de Sousa and D. Santos, “SearchCol: metaheuristic search by column generation", in Hybrid Metaheuristics, M. Blesa, C. Blum, G. Raidl, A. Roli, M. Sampels (Eds.), Lecture Notes in Computer Science, Volume 6373, pp. 190-205, 2010

[5] W. Ogryczak and T. Śliwiński, “On solving linear programs with the ordered weighted averaging objective", European Journal of Operational Research, 148, pp. 80-91, 2003

[6] W. Ogryczak, M. Pióro and A. Tomaszewski, “Telecommunications network design and max-min optimization problem", Journal of Telecommunications and Information Technology, 3, 2005

[7] D. Santos, A. de Sousa, F. Alvelos and M. Pióro, “Link load balancing optimization of telecommunication networks: a column generation based heuristic approach", in Proc. of NETWORKS Conference, IEEE Xplore, 2010

Modeling the collision avoidance for the ATM by a mixed 0–1 nonlinear approach

Antonio Alonso Ayuso * Laureano F. Escudero * Francisco Javier Martín Campo *

* Department of Statistics and Operational Research
C/ Tulipán s/n, 28933, Móstoles, Madrid (Spain)
{antonio.alonso,laureano.escudero,javier.martin.campo}@urjc.es

ABSTRACT

A 0–1 nonlinear model for the Collision Avoidance in Air Traffic Management (ATM) problem is presented. The aim of this problem is deciding the best strategy for an arbitrary aircraft configurations such that all conflicts in the airspace are avoided where a conflict is the loss of the minimum safety distance that two aircraft have to keep in their flight plans. A mixed 0–1 nonlinear optimization model based on geometric constructions is developed knowing the initial flight plan (coordinates, angles and velocities in each time period) and minimizing the acceleration variations where aircraft are forced to return to the original flight plan when no aircraft are in conflict. A linear approximation by using iteratively Taylor polynomials is developed to solve the problem in linear terms, as well as a metaheuristic based on Variable Neigbourhood Search (VNS) in order to reduce the resolution time.

Keywords: Air Traffic Management (ATM), Collision avoidance, Mixed 0-1 nonlinear optimization

1. INTRODUCTION

Aircraft conflict detection and resolution is currently attracting the interest of many air transportation service providers and is concerned with the following question: Given a set of airborne aircraft and their intended trajectories, what control strategy should be followed by the pilots and the air traffic service provider to prevent the aircraft from coming too close to each other?

Several approaches can be found in the literature, where different works tackle the problem from different points of view. In Kuchar and Yang (2000) [1] can be found several approaches until 2000. However, the developments from thereafter are very interesting. Some of the most important works can be found in Martín-Campo (2010) [2].

The organization of the remainder of the note is as follows. First, Section 2 presents the problem description. Section 3 introduces the notation of the elements of the problem. Section 4 presents the mixed 0–1 nonlinear model. Section 5 gives the main ideas of the iterative procedure for problem solving as well the VNDS metaheuristic scheme. Section 6 shows the main computational results and, finally, section 7 concludes.

2. PROBLEM DESCRIPTION

A 0–1 nonlinear constrained model is developed by using the geometric and theoretical ideas from the Velocity Changes problem (VC) presented in Pallottino et al. (2002) [3] and the Velocity and Altitude Changes problem (VAC) presented in Alonso-Ayuso et al. (2010) [4] and Martín-Campo (2010) [2]. The VC and VAC models assume instantaneous changes in velocity to avoid a conflict. In the new model, so-called VCTP (Velocity Changes through Time Periods), continuous velocity changes are proposed by using the properties of a rectilinear movement and uniformly accelerating movement. The VCTP model can also assume nonlinear trajectories by considering the polygonal (in each time period) of the trajectory.

The model suppose that the preliminary trajectories of F aircraft are known and it can be extracted the aircraft configurations in T fixed time points. In these points the velocity and the position (abscissa and ordinate) of each aircraft in each point and the motion angles between two points are known. With these data we construct a new model for obtaining the optimal configuration by changing the aircraft accelerations and avoiding all conflicts between the aircraft.

3. NOTATION

Let the following notation for the formulation of the model:

Sets

$\mathscr{F} = \{1,\ldots,F\}$, set of aircraft in the airspace.
$\mathscr{T} = \{0,\ldots,T\}$, set of time periods.

Parameters

s, safety distance between aircraft, usually, 2.5 nautical miles.

e, distance bound to consider as a conflicting pair of aircraft.

w_1, w_2, weight (between 0 and 1) for each objective function term.

div, integer parameter greater than 1 to be considered for the bounds of some variables.

For all $t \in \mathscr{T}$:

I_t, length of the time period between times instants $t-1$ and t.

For all $f \in \mathscr{F}$ and $t \in \mathscr{T}$:

x_f^{*t}, y_f^{*t}, initial configuration of position, abscissa and ordinate, for aircraft f in time period t, respectively.

d_f^{*t}, covered distance for aircraft f during time period t in the initial configuration.

v_f^{*t}, initial velocity configuration for aircraft f in time period t.

a_f^{*t}, initial acceleration configuration for aircraft f in time period t.

r_f^t, safety radius for each aircraft f in time period t, usually 2.5 nautical miles (nm).

$\underline{v}_f^t, \overline{v}_f^t$, minimum and maximum velocities allowed for aircraft f in time period t, respectively.

$\underline{a}_f^t, \overline{a}_f^t$, minimum and accelerations allowed for aircraft f in time period t, respectively.

m_f^{*t}, direction of motion in $(-\pi,\pi]$ for aircraft f in time period t.

$\hat{x}_f^t, \hat{y}_f^t$, position parameters to be updated in the Taylor approximation for aircraft f in time period t.

$\hat{d}_f^t$, distance parameter to be updated in the Taylor approximation for aircraft f in time period t.

$\hat{v}_f^t$, velocity parameter to be updated in the Taylor approximation for aircraft f in time period t.

$c_{ft}^{a^+}, c_{ft}^{a^-}$, costs for positive and negative acceleration changes for aircraft f in time period t, respectively.

c_{ft}^v, costs for the difference between the initial and optimal velocity configuration for aircraft f in time period t.

c_{ft}^d, costs for the difference between the initial and optimal covered distance for aircraft f in time period t.

For all $f \in \mathscr{F}$:

$x_f^{*t_f^d}, y_f^{*t_f^d}, x_f^{*t_f^r}, y_f^{*t_f^r}$, departure and arrival positions (abscissa and ordinate) for aircraft f.

d_f^{tot}, total length of the polygonal of the trajectory for aircraft f.

t_f^d, t_f^r, scheduled departure and arrival times for flight f.

Data preprocessing

For all $f \in \mathscr{F}$ and $t \in \mathscr{T}$:

$\overline{x}_f^t, \overline{y}_f^t, \overline{d}_f^t$, upper bounds for variables x, y and d, respectively.

$\underline{x}_f^t, \underline{y}_f^t$, lower bounds for variables x and y, respectively.

For all $i, j \in \mathscr{F} : i < j$, for all $t \in \mathscr{T}$:
$t = \left\{ \max\{t_i^d, t_j^d\} + 1, \ldots, \min\{t_i^r, t_j^r\} - 1 \right\}$:

fc_{ij}^t, 0–1 parameter that detects if there is a "false conflict" between aircraft i and j in time period t.

p_{ij}^t, 0–1 parameter that will be 1 if the pair of aircraft i and j will not be taken into account in time period t for conflict resolution. This parameter depends on the criterion decided by the ATC. Notice that this parameter will be 1 if $fc_{ij}^t = 1$.

ip_{ijt}, intersection point between the straight line trajectories of the corresponding polygonal segment for time period t for aircraft i and j, if the trajectories are not parallel o coincident.

d_{ijt}^1, distance between the i aircraft position and ip_{ijt} in time period t.

d_{ijt}^2, distance between points $\left(x_i^t + \cos(m_i^{*t}), y_i^t + \sin(m_i^{*t})\right)$ and ip_{ijt} for aircraft i and j in time period t.

Variables

For all $f \in \mathscr{F}$ and for all $t \in \mathscr{T}$:

x_f^t, y_f^t, the position (abscissa and ordinate) of aircraft f in time period t, for $f \in \mathscr{F}$ and $t \in \mathscr{T}$, respectively.

a_f^t, acceleration variation for aircraft f in time period t, for $f \in \mathscr{F}$ and $t \in \mathscr{T}$. This variable is real and can be divided in two positive variables, say, $a_f^{t^+}$ and $a_f^{t^-}$, such that $a_f^t = a_f^{t^+} - a_f^{t^-}$ as a traditional way in LP.

$a_f^{t^+}$, positive acceleration variation for aircraft f in time period t, for $f \in \mathscr{F}$ and $t \in \mathscr{T}$.

$a_f^{t^-}$, negative acceleration variation for aircraft f in time period t, for $f \in \mathscr{F}$ and $t \in \mathscr{T}$.

v_f^t, velocity for aircraft f in time period t, for $f \in \mathscr{F}$ and $t \in \mathscr{T}$.

d_f^t, covered distance for aircraft f during time period t, for $f \in \mathscr{F}$ and $t \in \mathscr{T}$.

$\gamma_{ft}^1, \gamma_{ft}^2, \gamma_{ft}^3, \gamma_{ft}^4, \gamma_f^t$ auxiliary 0–1 variables to model the case of early or delay for aircraft f in time period t, for $f \in \mathscr{F}$ and $t \in \mathscr{T}$.

For all $i, j \in \mathscr{F} : i < j$, for all $t \in \mathscr{T}$:
$t = \left\{ \max\{t_i^d, t_j^d\} + 1, \ldots, \min\{t_i^r, t_j^r\} - 1 \right\}$ and $n = 1, \ldots, 8$:

δ_{ijt}^n, auxiliary 0–1 variables for modeling the or-constraints.

4. MIXED 0–1 NONLINEAR MODEL

Now, the full formulation for the VCTP model is presented below, including all the aspects that have been studied above.

$$\min w_1 \sum_{f\in\mathscr{F}} \sum_{t\in\mathscr{T}} \left(\frac{c_{ft}^{a^+} a_f^{t+}}{\overline{a}_f^t - \underline{a}_f^t} + \frac{c_{ft}^{a^-} a_f^{t-}}{\overline{a}_f^t - \underline{a}_f^t} \right) + w_2 \sum_{f\in\mathscr{F}} \sum_{t\in\mathscr{T}} c_{ft}^d \beta_f^t \tag{1}$$

subject to $\forall f \in \mathscr{F}, \forall t \in \mathscr{T} : t = \{t_f^d + 1, \ldots, t_f^r\}$

$$\underline{v}_f^t \leqslant \quad v_f^{t-1} + a_f^t I_t \quad \leqslant \overline{v}_f^t \tag{2a}$$

$$\underline{a}_f^t \leqslant \quad a_f^t \quad \leqslant \overline{a}_f^t \tag{2b}$$

$\forall f \in \mathscr{F}, \forall t \in \mathscr{T} : t = \{t_f^d + 1, \ldots, t_f^r\}$

$$d_f^t = v_f^{t-1} I_t + \frac{1}{2}(a_f^{t+} - a_f^{t-}) I_t^2 \tag{3}$$

$\forall i, j \in \mathscr{F} : i < j \wedge p_{ij} = 0, \forall t \in \mathscr{T} : t = \left\{ \max\{t_i^d, t_j^d\} + 1, \ldots, \min\{t_i^r, t_j^r\} - 1 \right\}$

$$v_i^t\left(\cos(m_i^{*t})(1 - pc_{ij}^t) - \sin(m_i^{*t}) pc_{ij}^t\right) - v_j^t\left(\cos(m_j^{*t})(1 - pc_{ij}^t) - \sin(m_j^{*t}) pc_{ij}^t\right) \leqslant (\overline{v}_i^t + \overline{v}_j^t)(1 - \delta_{ijt}^1) \tag{4a}$$

$$-v_i^t\left(h_i^t(1 - pc_{ij}^t) + h_i'^t pc_{ij}^t\right) + v_j^t\left(h_j^t(1 - pc_{ij}^t) + h_j'^t pc_{ij}^t\right) \leqslant \left((\overline{v}_i^t|h_i^t| + \overline{v}_j^t|h_j^t|)(1 - pc_{ij}^t) + (\overline{v}_i^t|h_i'^t| + \overline{v}_j^t|h_j'^t|) pc_{ij}^t\right)(1 - \delta_{ijt}^1) \tag{4b}$$

$$v_i^t\left(\cos(m_i^{*t})(1 - pc_{ij}^t) - \sin(m_i^{*t}) pc_{ij}^t\right) - v_j^t\left(\cos(m_j^{*t})(1 - pc_{ij}^t) - \sin(m_j^{*t}) pc_{ij}^t\right) \leqslant (\overline{v}_i^t + \overline{v}_j^t)(1 - \delta_{ijt}^2) \tag{4c}$$

$$v_i^t\left(k_i^t(1 - pc_{ij}^t) + k_i'^t pc_{ij}^t\right) - v_j^t\left(k_j^t(1 - pc_{ij}^t) + k_j'^t pc_{ij}^t\right) \leqslant \left((\overline{v}_i^t|k_i^t| + \overline{v}_j^t|k_j^t|)(1 - pc_{ij}^t) + (\overline{v}_i^t|k_i'^t| + \overline{v}_j^t|k_j'^t|) pc_{ij}^t\right)(1 - \delta_{ijt}^2) \tag{4d}$$

$$-v_i^t\left(\cos(m_i^{*t})(1 - pc_{ij}^t) - \sin(m_i^{*t}) pc_{ij}^t\right) + v_j^t\left(\cos(m_j^{*t})(1 - pc_{ij}^t) - \sin(m_j^{*t}) pc_{ij}^t\right) \leqslant (\overline{v}_i^t + \overline{v}_j^t)(1 - \delta_{ijt}^3) \tag{4e}$$

$$v_i^t\left(h_i^t(1 - pc_{ij}^t) + h_i'^t pc_{ij}^t\right) - v_j^t\left(h_j^t(1 - pc_{ij}^t) + h_j'^t pc_{ij}^t\right) \leqslant \left((\overline{v}_i^t|h_i^t| + \overline{v}_j^t|h_j^t|)(1 - pc_{ij}^t) + (\overline{v}_i^t|h_i'^t| + \overline{v}_j^t|h_j'^t|) pc_{ij}^t\right)(1 - \delta_{ijt}^3) \tag{4f}$$

$$-v_i^t\left(\cos(m_i^{*t})(1 - pc_{ij}^t) - \sin(m_i^{*t}) pc_{ij}^t\right) + v_j^t\left(\cos(m_j^{*t})(1 - pc_{ij}^t) - \sin(m_j^{*t}) pc_{ij}^t\right) \leqslant (\overline{v}_i^t + \overline{v}_j^t)(1 - \delta_{ijt}^4) \tag{4g}$$

$$-v_i^t\left(k_i^t(1 - pc_{ij}^t) + k_i'^t pc_{ij}^t\right) + v_j^t\left(k_j^t(1 - pc_{ij}^t) + k_j'^t pc_{ij}^t\right) \leqslant \left((\overline{v}_i^t|k_i^t| + \overline{v}_j^t|k_j^t|)(1 - pc_{ij}^t) + (\overline{v}_i^t|k_i'^t| + \overline{v}_j^t|k_j'^t|) pc_{ij}^t\right)(1 - \delta_{ijt}^4) \tag{4h}$$

$$\delta_{ijt}^1 + \delta_{ijt}^2 + \delta_{ijt}^3 + \delta_{ijt}^4 = 1 \tag{4i}$$

$\forall f \in \mathscr{F}, \forall t \in \mathscr{T} : t = \{t_f^d + 1, \ldots, t_f^r\}$

$$\sum_{\ell=1}^{t} d_f^{\ell} - \sum_{\ell=1}^{t} d_f^{*\ell} \leqslant \frac{d_f^{*t}}{div}\gamma_f^t \tag{5a}$$

$$\sum_{\ell=1}^{t} d_f^{\ell} - \sum_{\ell=1}^{t} d_f^{*\ell} - \varepsilon \geqslant (-\frac{d_f^{*t}}{div} - \varepsilon)(1-\gamma_f^t) \tag{5b}$$

$$x_f^t - x_f^{*t} - \Big(\sum_{\ell=1}^{t} d_f^{\ell} - \sum_{\ell=1}^{t} d_f^{*\ell}\Big)\cos(m_f^t) \leqslant (\overline{x}_f^t - x_f^{*t} + \frac{d_f^{*t}}{div})(1-\gamma_f^t) \tag{5c}$$

$$x_f^t - x_f^{*t} - \Big(\sum_{\ell=1}^{t} d_f^{\ell} - \sum_{\ell=1}^{t} d_f^{*\ell}\Big)\cos(m_f^{*t}) \geqslant (\underline{x}_f^t - x_f^{*t} - \frac{d_f^{*t}}{div})(1-\gamma_f^t) \tag{5d}$$

$$y_f^t - y_f^{*t} - \Big(\sum_{\ell=1}^{t} d_f^{\ell} - \sum_{\ell=1}^{t} d_f^{*\ell}\Big)\sin(m_f^{*t}) \leqslant (\overline{y}_f^t - y_f^{*t} + \frac{d_f^{*t}}{div})(1-\gamma_f^t) \tag{5e}$$

$$y_f^t - y_f^{*t} - \Big(\sum_{\ell=1}^{t} d_f^{\ell} - \sum_{\ell=1}^{t} d_f^{*\ell}\Big)\sin(m_f^{*t}) \geqslant (\underline{y}_f^t - y_f^{*t} - \frac{d_f^{*t}}{div})(1-\gamma_f^t) \tag{5f}$$

$$x_f^t - x_f^{*t-1} - \Big(\sum_{\ell=1}^{t} d_f^{\ell} - \sum_{\ell=1}^{t-1} d_f^{*\ell}\Big)\cos(m_f^{*t-1}) \leqslant (\overline{x}_f^t - x_f^{*t-1} + d_f^{*t})\gamma_f^t \tag{5g}$$

$$x_f^t - x_f^{*t-1} - \Big(\sum_{\ell=1}^{t} d_f^{\ell} - \sum_{\ell=1}^{t-1} d_f^{*\ell}\Big)\cos(m_f^{*t-1}) \geqslant (\underline{x}_f^t - x_f^{*t-1} - d_f^{*t})\gamma_f^t \tag{5h}$$

$$y_f^t - y_f^{*t-1} - \Big(\sum_{\ell=1}^{t} d_f^{\ell} - \sum_{\ell=1}^{t-1} d_f^{*\ell}\Big)\sin(m_f^{*t-1}) \leqslant (\overline{y}_f^t - y_f^{*t-1} + d_f^{*t})\gamma_f^t \tag{5i}$$

$$y_f^t - y_f^{*t-1} - \Big(\sum_{\ell=1}^{t} d_f^{\ell} - \sum_{\ell=1}^{t-1} d_f^{*\ell}\Big)\sin(m_f^{*t-1}) \geqslant (\underline{y}_f^t - y_f^{*t-1} - d_f^{*t})\gamma_f^t \tag{5j}$$

$$\sum_{\ell=1}^{t} d_f^{\ell} - \sum_{\ell=1}^{t} d_f^{*\ell} \leqslant \beta_f^t \tag{6a}$$

$$\sum_{\ell=1}^{t} d_f^{*\ell} - \sum_{\ell=1}^{t} d_f^{\ell} \leqslant \beta_f^t \tag{6b}$$

$\forall f \in \mathscr{F}, \forall t \in \mathscr{T} : t = \{t_f^d + 1, \ldots, t_f^r - 1\}$

$$\underline{d}_f^t = d_f^{*t} - \frac{d_f^{*t}}{div} \leqslant d_f^t \leqslant d_f^{*t} + \frac{d_f^{*t}}{div} = \overline{d}_f^t \tag{7a}$$

$$\underline{x}_f^t = x_f^{*t} - \frac{d_f^{*t}}{div}\cos(m_f^{*t-1}) \leqslant x_f^t \leqslant x_f^{*t} + \frac{d_f^{*t}}{div}\cos(m_f^{*t}) = \overline{x}_f^t \tag{7b}$$

$$\underline{y}_f^t = y_f^{*t} - \frac{d_f^{*t}}{div}\sin(m_f^{*t-1}) \leqslant y_f^t \leqslant y_f^{*t} + \frac{d_f^{*t}}{div}\sin(m_f^{*t}) = \overline{y}_f^t \tag{7c}$$

$\forall f \in \mathscr{F}, \forall t \in \mathscr{T} : t = \{t_f^d, \ldots, t_f^r\}$

$$x_f^t, y_f^t, a_f^t \in \mathbb{R} \tag{8a}$$

$$v_f^t, a_f^{t+}, a_f^{t-}, d_f^t, \beta_f^t \in \mathbb{R}^+ \tag{8b}$$

$\forall i, j \in \mathscr{F} : i < j \wedge p_{ij} = 0, \forall t \in \mathscr{T} : t = \{\max\{t_i^d, t_j^d\} + 1, \ldots, \min\{t_i^r, t_j^r\} - 1\}$

$$\delta_{ijt}^1, \delta_{ijt}^2, \delta_{ijt}^3, \delta_{ijt}^4 \in \{0,1\} \tag{8c}$$

$\forall f \in \mathscr{F}, \forall t \in \mathscr{T} : t = \{t_f^d, \ldots, t_f^r\}$:

$$\gamma_f^t \in \{0,1\} \tag{8d}$$

The objective function (1) gives the optimization criterion for the model. It has two terms, one for minimizing the sum of the absolute values of the accelerations and the other one for forcing aircraft to return to the initial configuration, where the values of the w_1 and w_2 emphasizes one term over the other. If the second term of the objective function is contemplated, it must be accompanied by constraints (6). Constraints (2) avoid the velocity and the acceleration to be bigger or smaller than the upper or lower bound, respectively. Constraints (3) update the covered distance by an aircraft after the changes in its configuration in time period $t \in \mathscr{T}$. Constraints (4) detect and solve the conflicts in the airspace. The next terms are nonlinear ones:

$$h_i^t = \frac{(x_i^t - x_j^t)s + (y_i^t - y_j^t)\sqrt{(x_i^t - x_j^t)^2 + (y_i^t - y_j^t)^2 - s^2}}{(x_i^t - x_j^t)\sqrt{(x_i^t - x_j^t)^2 + (y_i^t - y_j^t)^2 - s^2} - (y_i^t - y_j^t)s}\cos(m_i^{*t}) - \sin(m_i^{*t}) \tag{9a}$$

$$h_j^t = \frac{(x_i^t - x_j^t)s + (y_i^t - y_j^t)\sqrt{(x_i^t - x_j^t)^2 + (y_i^t - y_j^t)^2 - s^2}}{(x_i^t - x_j^t)\sqrt{(x_i^t - x_j^t)^2 + (y_i^t - y_j^t)^2 - s^2} - (y_i^t - y_j^t)s}\cos(m_j^{*t}) - \sin(m_j^{*t}) \tag{9b}$$

$$k_i^t = \frac{-(x_i^t - x_j^t)s + (y_i^t - y_j^t)\sqrt{(x_i^t - x_j^t)^2 + (y_i^t - y_j^t)^2 - s^2}}{(x_i^t - x_j^t)\sqrt{(x_i^t - x_j^t)^2 + (y_i^t - y_j^t)^2 - s^2} + (y_i^t - y_j^t)s}\cos(m_i^{*t}) - \sin(m_i^{*t}) \tag{9c}$$

$$k_j^t = \frac{-(x_i^t - x_j^t)s + (y_i^t - y_j^t)\sqrt{(x_i^t - x_j^t)^2 + (y_i^t - y_j^t)^2 - s^2}}{(x_i^t - x_j^t)\sqrt{(x_i^t - x_j^t)^2 + (y_i^t - y_j^t)^2 - s^2} + (y_i^t - y_j^t)s}\cos(m_j^{*t}) - \sin(m_j^{*t}) \tag{9d}$$

Constraints (5) update the positions in each time period $t \in \mathscr{T}$, according to the previous changes in velocity made until the current time period. Constraints (6) transform the second term of the objective function in a linear function, since it minimizes an absolute value between the covered distances in the initial flight plan and the covered distances in the resolution. Constraints (7) force some variables to be into fixed bounds, in order to calculate the big M in the position projection constraints. Finally, constraints (8) give the variables' type. See [2] and the full version of the paper [5] for a detailed explanation.

5. ALGORITHMIC APPROACH

For solving iteratively the linearized model, the algorithmic approach described in Almiñana et al. (2007) [6] in a different context inspires the work presented in this paper. It is based on a iterative optimization by starting with the initial configuration and updating the input parameters where the derivatives are centered until a stop criterion is satisfied.

First, the nonlinear constraints have to be linearized by using Taylor polynomials, so the new mathematical expression for each inequality ($n = 1, \ldots, 4$) can be expressed as follows,

$$\Big\{C_n + \big(\frac{\partial C_n}{\partial(v_{i \text{ and } j}^t, x_{i \text{ and } j}^t, y_{i \text{ and } j}^t)}\big)\Big\}_{|(\hat{v}_{i \text{ and } j}^t, \hat{x}_{i \text{ and } j}^t, \hat{y}_{i \text{ and } j}^t)} \begin{pmatrix} v_{i \text{ and } j} - \hat{v}_{i \text{ and } j}^t \\ x_{i \text{ and } j} - \hat{x}_{i \text{ and } j}^t \\ y_{i \text{ and } j} - \hat{y}_{i \text{ and } j}^t \end{pmatrix} \leq M(1 - \delta_{ijt}^n).$$

The algorithm for the resolution is presented below:

Step 1. Computing the values of the nonlinear constraints $\forall i, j \in \mathscr{F} : i < j$ and $\forall f \in \mathscr{F}$, such that $v_i^t, v_j^t, x_i^t, x_j^t, y_i^t, y_j^t$ are replaced with $\hat{v}_i^t, \hat{v}_j^t, \hat{x}_i^t, \hat{x}_j^t, \hat{y}_i^t, \hat{y}_j^t$, respectively. In the first iteration, $\hat{v}_i^t = v_i^{*t}, \hat{v}_j^t = v_j^{*t}, \hat{x}_i^t = x_i^{*t}, \hat{x}_j^t = x_j^{*t}, \hat{y}_i^t = y_i^{*t}, \hat{y}_j^t = y_j^{*t}, \hat{d}_f^t$.

Step 2. Solving the mixed zero-one model, where the nonlinear constraints are substituted by its linear approximation. Let d_f^t be the optimal values of the respective variables.

Step 3. Optimality test. If the following condition is satisfied then stop, the quasi-optimal solution has been obtained. Otherwise, go to Step 4.

$$\sum_{f \in \mathscr{F}} \sum_{t \in \mathscr{T}} (d_f^t - \hat{d}_f^t)^2 \leq \varepsilon$$

where ε is a positive tolerance.

Step 4. Updating the covered distance, the acceleration, the velocity and the positions and go to Step 1.

See [2] and [5] for a detailed explanation.

Now, the linearized model can be solved by using the optimization engine of choice for mixed 0–1 linear models. Unfortunately, the resolution time for large-scale instances could be very big due to the exponential complexity of the Branch and Bound (B&B) schemes in MILP.

In view of the situation, one alternative consists of implementing a metaheuristic scheme able to get a good solution in an affordable time. The characteristics of this problem make the optimality to be aside, being the feasibility of the solution the first goal. The scheme of choice has been published by Lazić et al. (2010) [7], so called "Variable Neighbourhood Decomposition Search" (VNDS).

6. COMPUTATIONAL RESULTS

For comparison purposes of the VNDS solution quality, Table 1 shows the objective function values by plain use of the engine CPLEX [8] for optimizing the VCTP model according to the scheme presented in section 5 and, independently, by using the VNDS scheme for a large testbed, where the headings are as follows:

- Case: Case configuration: C-AAA denotes number of aircraft (AAA) in conflict.
- z_{ip}: Optimal solution by using the exact algorithm.
- z_{VNDS}: Best solution obtained by using VNDS.
- GAP_{VNDS}: $\frac{z_{VNDS}-z_{ip}}{z_{VNDS}}$ %
- t_{ip}: Time (secs.) to obtain the z_{ip} value.
- t_{VNDS}: Best time (secs.) to obtain the best solution by using VNDS.

Table 1 shows the small GAP between the optimal solution obtained by the above exact algorithm and the best solution obtained by the VNDS scheme. The resolution times are notably shorter by using the metaheuristic scheme instead of the exact one. See [2], [5] and [9] for an extensive computational experience.

7. CONCLUSIONS

A mixed 0–1 nonlinear optimization model has been presented in order to solve the collision avoidance for the ATM problem. Due to four hard nonlinear constraints, the model has been linearized by using iteratively Taylor polynomials approximations since no optimization engine to solve mixed integer nonlinear models could solve the problem, as we know. Solving the model by successive mixed 0–1 linear approximations could require a non affordable time. However, a metaheuristic based on VNS has been implemented, getting good solutions (GAP less than 1%) in short time.

8. ACKNOWLEDGEMENTS

This work is partially supported by i-Math Ingenio Mathematica. This work has been carried out within the Framework of ATLANTIDA project, partially funded by the Spanish CDTI, in which the Rey Juan Carlos University is collaborating with GMV Aerospace and Defence S.A. This research has been also partially supported by the project PLANIN MTM2009-14087-C04-01 from Ministry of Science and Innovation, Spain.

The authors would like to thank N. Mladenović and J. Lazić, whose metaheuristic VNDS scheme has helped to obtain the good computational results that have been reported.

Case	z_{ip}	z_{VNDS}	$\text{GAP}_{\text{VNDS}}(\%)$	t_{ip}	t_{VNDS}
C-025	0.0227	0.0228	0.44	19.95	2.02
C-035	0.0315	0.0318	0.94	54.54	8.73
C-045	0.0457	0.0459	0.44	116.55	18.90

Table 1: Results comparison

9. REFERENCES

[1] J. K. Kuchar and L. C. Yang, "A review of conflict detection and resolution modeling methods," *IEEE Transactions on Intelligent Transportation Systems*, vol. 1, pp. 179–189, 2000.

[2] F. J. Martín-Campo, "The collision avoidance problem: Methods and algorithms," Ph.D. dissertation, Department of Statistics and Operational Research, Rey Juan Carlos University, 2010.

[3] L. Pallottino, E. Feron, and A. Bicchi, "Conflict resolution problems for air traffic management systems solved with mixed integer programming," *IEEE Transactions on Intelligent Transportation Systems*, vol. 3, no. 1, pp. 3–11, 2002.

[4] A. Alonso-Ayuso, L. F. Escudero, and F. J. Martín-Campo, "Collision avoidance in the air traffic management: A mixed integer linear optimization approach," *IEEE Transactions on Intelligent Transportation Systems, DOI: 10.1109/TITS.2010.2061971*, 2010.

[5] A. Alonso-Ayuso, L. F. Escudero, and F. J. Martín-Campo, "A mixed 0-1 nonlinear approach for the collision avoidance in atm: Velocity changes through a time horizon," *To be submitted*, 2011.

[6] M. Alminana, L. F. Escudero, M. Landete, J. F. Monge, A. Rabasa, and J. Sánchez-Soriano, "On a mixed 0-1 separable nonlinear approach for water irrigation scheduling," *IIE Transactions*, vol. 44, no. 4, pp. 398–405, 2008.

[7] J. Lazić, S. Hanafi, N. Mladenović, and D. Urošević, "Variable neighbourhood decomposition search for 0-1 mixed integer programs," *Computers & Operations Research*, vol. 37, pp. 1055–1067, 2010.

[8] IBM ILOG, *CPLEX v12.1. User's Manual for CPLEX*, 2009.

[9] A. Alonso-Ayuso, L. F. Escudero, and F. J. Martín-Campo, "Variable neighbourhood decomposition search for the mixed 0-1 nonlinear collision avoidance problem," *To be submitted*, 2011.

Low Energy Scheduling with Power Heterogeneous Multiprocessor Systems

Richard Dobson * Kathleen Steinhöfel *

* King's College London,
Department of Informatics
{richard.dobson, kathleen.steinhofel}@kcl.ac.uk

ABSTRACT

In this paper we consider low energy scheduling for power heterogeneous multiprocessor systems. This is a fast developing area that is of great importance and is currently being researched by both industry and academia. This problem is of great importance because real life multiprocessor computer systems are often heterogeneous at run time. We have developed an algorithm which transforms any multiprocessor system into a Virtual Single Processor (VSP). Using our VSP platform, existing techniques can be explored for low energy scheduling for heterogeneous multiprocessor scheduling. In this study we focus on applying algorithms which minimise ∑Flow + Energy in conjunction with our VSP approach. ∑Flow + Energy have been shown to be very useful in real life situations.

Keywords: Virtual Single Processor, Dynamic Speed Scaling, Energy, Heterogeneous Multiprocessor Systems, Low Energy Scheduling

1. INTRODUCTION

Energy consumption is an important consideration when designing computer systems; especially when for use as a mobile device such as a smart phone, which needs to provide both high performance and good battery life. One of the largest drains of energy in a computer system is the processor(s). In most modern processors energy consumption and processing speed are intrinsically linked, this is normally through the relationship $Power = Speed^a$ where a is a constant which differs between processors but is typically between 2 and 3. A good way of reducing the amount of energy a processor uses is to lower the operational speed, for this we use Dynamic Speed Scaling (DSS).

DSS allows the operating frequency of a processor to be modified at runtime. This means that we do not have to run the processor at maximum speed and waste energy when demand is low. DSS is a tool which enables the speed of the processor(s) to be lowered but we need to combine this with an objective function which controls how much the processors should be slowed down by. This is by no means a trivial task as we need to ensure that energy is not needlessly wasted processing jobs too quickly; but we also need to ensure that jobs are completed by at time such that they are not compromising the operation of the computer system or the user experience.

Multiprocessor computer systems have become increasingly common in the past few years with the most common of these being homogeneous multiprocessor systems; where all processors are equal. The general consensus between academics and industry is that the way forward is heterogeneous multiprocessor systems; where processors are not all equal. Heterogeneous multiprocessor systems are often favourable over homogeneous options due to their flexibility. They can have many low powered processors for low priority tasks or if the demand is low and a collection of high powered processors to deal with high priority jobs or to relive pressure if there is a large amount of jobs. This structure has the potential to provide much greater performance for the user and consume considerably less energy than an equivalent homogeneous system. We also often find that a homogeneous multiprocessor system which implements DSS is heterogeneous at run time.

We address the problem of trying to combine DSS with a heterogeneous multiprocessor system to provide a high level of performance and energy efficiency.

1.1. Related Work

Yao et. al. [9] presented a solution which assumed all jobs have a hard deadline, the processor is then ran at a speed such that all jobs are processed before their deadlines. Job deadlines do not always solicit a feasible schedule unless we allow the maximum speed of the processor to be infinite or we restrict the set of incoming jobs.

Some years later Albers and Fujiwara [1] presented an alternative solution which does not rely on deadlines but attempts to balance the quality of service against power consumption. The authors attempt to minimise $\sum Flow + Energy$, this balances the two conflicting values of $Energy$ against $\sum Flow$ where the $Flow$ of each job is the time between it's release and completion. This means that there is always a feasible solution without the need to have infinite speed processors or restricted job sets. The best algorithm for this problem has been developed by Andrew et al [2] and obtains a competitive ratio of $(2+\varepsilon)$. Andrew et al also show that there exists some trade off functions for which no algorithm can be better than (2)Competitive.

Lam et al extended existing knowledge of single processor F+E algorithms to the homogeneous multiprocessor situation in their paper [7]. The authors looked at the online problem and suggested an algorithm Classified Round Robin (CRR) where jobs are distributed fairly evenly between all processors using a weighted round robin; individual processors are then left to manage their own speed using a single processor F+E minimisation algorithm. Less than a year later Lam et al released [8] which presents a slightly improved version of the algorithm.

Most recently Gupta et al [5] developed an algorithm for heterogeneous multiprocessor systems. They approached the problem (which was formalised by Bower et al [4]) in a similar way to Lam et al has approached the homogeneous multiprocessor problem. Instead of distributing jobs as evenly as possible they aim to distribute jobs to the processor which will provide the least increase in ∑weighted flow. The algorithm is outlined below.

1. Job Selection (which job should be ran on each processor): Highest Density First
2. Speed Scaling (what speed should each processor run at): The speed is set so the power is the fractional weight of the

Research partially supported by EPSRC Grant No. EP/G501483/1 and Nokia Ltd.

unfinished jobs

3. Assignment (which processor should each job be assigned to): A new job is assigned to the processor that results in the least increase in the projected future weighted flow, assuming the adopted speed scaling and job selection policies, and ignoring the possibility of jobs arriving in the future

In the paper the authors prove this algorithm to be "scalable for scheduling jobs on a heterogeneous multiprocessor with arbitrary power functions to minimize the objective function of weighted flow plus energy".

2. LOW ENERGY SCHEDULING FOR HETEROGENEOUS MULTIPROCESSOR SYSTEMS

In [5] the authors present a solution for power heterogeneous multiprocessor systems using weighted Flow + Energy as their objective function. This solution has been shown to be theoretically sound for the model which they use. If we consider a real life multiprocessor computer system we often find that there are constraints which are applied to the processors. For example many multi-core processors require the cores to always run at the same speed. The Gupta et. al approach is not compatible with this situation as the processor speed is linked to the number of jobs.

The Gupta et al algorithm also requires a large amount of runtime computation. Each time a job needs to be assigned to a processor it must find out which processor will provide the smallest increase in the weighted flow. This requires a small amount of processing for just one job but over time this could potentially add up to a large amount of computation. We present a solution that has the ability to overcome both of these issues.

Our solution suggests that we should form a 'Virtual Single Processor' (VSP). The VSP is essentially a collection of processors which have been combined together in an efficient way to form what appears to be a single processor. We then present the VSP (as a single processor) to a DSS algorithm which controls the speed of the overall VSP and specifies which job should be processed first. The VSP in turn translates the overall VSP speed into speeds for each processor which when combined equal the VSP speed.

3. THE VIRTUAL SINGLE PROCESSOR

We begin by defining the term 'system speed', that is the combined processing power of all processors. We wish to use the least amount of energy possible for any system speed. To achieve this we control the processor speed at system level and allow the processors to request a new job when they have completed their current job.

We consider an example of a 4 processor system (P0, P1, P2, P3) where each processor has a finite set of speeds and a simple power function in the form of $P = S^a$, the attributes of which are outlined below.

- P0 (0, 200,300,400, 500) a=2.3
- P1 (0, 600, 700,800,900) a=2.35
- P2 (0, 100, 300, 500, 700, 900) a=2.5
- P3 (0, 1200) a=2.2

We define a combination to be a set of processing speeds (one for each processor); each combination has a system speed which is the sum of the individual speeds. For example {200,600,100,1200} is a valid combination with a system speed of 2100. This means it can process 2100 cycles worth of work per second. In the simple

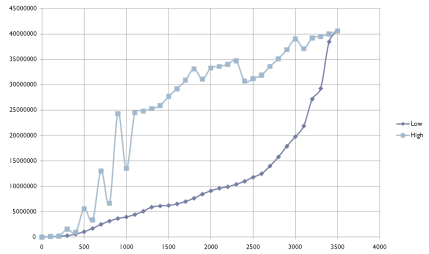


Figure 1: A graph showing the relationship between system speed and power consumption for the best case and worst case processor combinations.

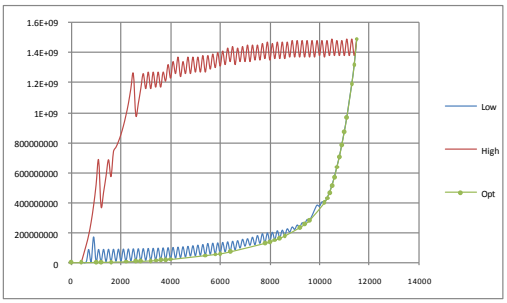


Figure 2: A graph showing the relationship between system speed and power consumption including energy spikes and optimal system speeds marked by dots

system outlined above there are 300 unique combinations many of these provide the same system speed. For example there are 14 different combinations which make up the overall system speed of 1400 alone.

If we consider the worst and best ways of achieving the system speed of 1400 (with regards to energy) we find that the worst case uses 487% of the energy consumed by the most efficient combination. If we look at Figure 1 we can see the difference between the most and least energy is largest in mid range speeds with the graph converging at either end of the system speed range. This simple example highlights how crucial it is to make the correct processor speed selections and this is where the VSP method stems from.

The VSP method pre-computes the optimal processor combination such that no other combination of processors provides the same system speed but uses less energy. To find out which combinations are best we first compute all processor combinations, these are then ordered by system speed. We then compare all of the combinations with the same system speed and discard all apart from the combination with the lowest energy usage. Now we have the list of optimal combinations for each system speed although this is not always good enough.

Using a technique from [3] we can 'simulate' any speed between 0 and S_{max} by alternating between two speeds in different ratios. For example if we wanted to simulate the processor speed of 4 with a single processor which can only operate at either speed 3 or speed 5 we could run the processor for half the time at 3 and half the time at 5. This would average out at speed 4. We can use this method to ensure that we are always using the optimal amount of energy at all times. Figure 2 shows how we can use this technique to lower the overall energy consumption of the system for some speeds.

The VSP essentially provides a level of abstraction between the single processor algorithm and the multiprocessor system. This abstraction allows us to hide the complexity of the multiprocessor system behind the VSP front. We can hide complex requirements by computing a sub VSP where all processor combinations adhere

to the requirements of the system. For example if two cores must always operate at identical speed then we only accept combinations where this is the case. We then feed this sub VSP into the overall VSP as if it were a processor.

By computing the VSP we can also remove the burden of calculating which processor is best for each job as in [5]. This is made possible as the VSP hides the fact that more processors exist and only assigns jobs when a processor has a speed greater than 0 and no job. This means that there is no need for the algorithm to perform costly calculations at run time.

3.1. Formal Description

We first define U to be a set of processing units where each processor U_x has two properties U_xS which is the set of valid processing speeds (which may or may not contain simulated speeds) and $U_xP(s)$ which is a function which given an input of a valid speed s returns e the energy used per second by this speed.

To transform individual processors into a VSP we first combine the processors to form a set of all possible combinations $C = U_0S \times U_1S.... \times U_{n-1}S$ where n is the total number of processing units. We define each combination C_i to have 2 functions C_iE which is the energy consumption of all processors used in the combination and C_iS which is the overall speed of all processors in the combination. We then remove inefficient combinations to leave a streamlined collection of efficient processor combinations.

We define C' to be a new set which contains only the combinations which adhere to the following rules.

$$\begin{array}{l} \forall a C_a \in C' \text{iff} \\ \forall b C_aE \leq C_bE \quad \text{where} \quad C_aS = C_bS \\ \text{and} \\ \neg\exists C_c \in C' \quad \text{where} \quad C_cS = C_aS \end{array} \tag{1}$$

To form the VSP (V_0) we need to extract the available speeds and the power function from C'.

$$\begin{array}{l} V_0S = \{C'_0S, ..., C'_nS\} \\ \forall V_0P(V_0S_a) = C'_aE \end{array} \tag{2}$$

There is one final step we must perform to ensure optimal VSP performance; we must remove speeds which are not efficient.

$$\forall_{0<x<n} V_0P(V_0S_x) < \left(\begin{array}{c} \left(\left(\frac{V_0S_x - V_0S_{x-1}}{V_0S_{x+1} - V_0S_{x-1}} \right) \times V_0P(V_0S_{x+1}) \right) \\ + \\ \left(\left(1 - \left(\frac{V_0S_x - V_0S_{x-1}}{V_0S_{x+1} - V_0S_{x-1}} \right) \right) \times V_0P(V_0S_{x-1}) \right) \end{array} \right) \tag{3}$$

4. USING THE VIRTUAL SINGLE PROCESSOR

Once we have constructed our VSP three things are needed before it can be used: a job selection policy, a speed scaling policy and to know whether the computer system will allow jobs to migrate between processors or not. The third point is crucial to job selection and processor speed changes so we will discuss these two options separately.

4.1. Migratory

Incoming jobs are sorted according to their ranking as judged by the job selection policy. The job with the highest ranking from all jobs which have not yet finished processing is always assigned to the fastest processor, the second highest ranking with the second fastest processor and so on. Jobs are interrupted and replaced such the first criteria is always true. This makes sure that jobs with high priorities always finish quickly. The speed scaling policy is used in conjunction with the system power function to determine what speed our system should operate at, this is then translated into individual processor speeds by the VSP. If a processor is directed to use speed 0 then the job it is currently processing is suspended and returned to the list of incoming jobs.

4.2. Non Migratory

Once again the incoming jobs are sorted according to their ranking as judged by the job selection policy but we also keep a note of 2 things for each processor, how much time is required to finish the current job being processed if the processor speed stays constant P_iT and what the current speed of the processor is P_iS_c. We then calculate which processor will allow the highest priority job to finish first providing processor speeds stay constant and the second highest to finish second and so on. Jobs are then assigned to the 'correct' processors when they become available. When the speed scaling algorithm decides that the system speed should change the VSP converts this into individual processor speed changes; if the speed of a processor should raise then this happens straight away, if the speed of a processor should drop this action is taken after the processor has finished processing the current job. This ensures that no job is trapped on a processor which has a speed 0 as this could result in the job never being finished.

5. VSP SYSTEM ANALYSIS

It is important that we do not see the VSP as a complete scheduling algorithm. It is a platform which allows scheduling algorithms to be applied to or developed for heterogeneous multiprocessor system more easily. In this section we compare the VSP platform combined with the Gupta et al speed scaling algorithm to the Gupta et al approach to show that the VSP is a strong alternative solution.

We first consider a batch of tasks arriving at over time so that we can compare how each approach will deal with these. The Gupta approach will sort the jobs by their density and then calculate which processor will provide the least increase in projected flow for each job, the job is assigned to this processor. Each processor will calculate what speed it should be running at based on the fractional weighted flow of it's work. The VSP approach will sort the jobs by their density and then calculate what the speed of the VSP should be. The VSP will then instruct the processors what speed they should be running at. Jobs are assigned to a processor if it's speed is greater than 0 and it does not already have a job. Jobs with higher priorities will be assigned to faster processors.

The VSP approach has allowed us to remove the majority of the computation from run time; this is possible because we pre-compute the processor configurations. Once we have the processor configurations all we have to do at run time is look up the desired system speed based on the quantity of work and implement the indicated processor speeds. Job distribution is then simplified to putting the highest density job on the fastest processor. This indicates that in real life situations the VSP approach could save a considerable amount of effort at run time over the Gupta et al approach.

5.1. Simulations

To test the overall performance of the VSP system we developed simulations of both approaches and ran a number of tests with a variety of processor configurations and job sets. In this section we will highlight the tests regarding a processor configuration outlined in [6]. The suggested processor configuration has x high powered

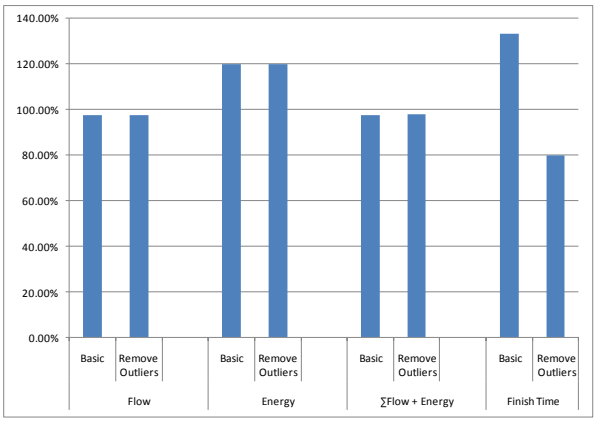


Figure 3: A graph showing the average results from the simulations and the average results with outliers removed

processors, $2x$ medium powered processors and $4x$ high powered processors.

We use the processor configuration from [6] where $x = 1$. $1 \times$ High speed processor: $s = \{0, 1000, 2000\} a = 2.8$, $2 \times$ medium speed processors: $s = \{0, 250, 500, 750, 1000\} a = 2.55$, $4 \times$ low speed processor: $s = \{0, 50, 100, 150, 200, 250\} a = 2.25$. We split the test data into 3 different categories all of which contain jobs with random weights and sizes:

- Immediate; all jobs are all released at time 0
- Uniformly random; jobs are released randomly over time up to time x
- Peaks and troughs; jobs are released in surges which are similar to the action of a computer system up to time x

Covering these categories gives us a good indication of overall system performance.

After running our simulations we found some very interesting results. We report the values below in terms of VSP performance in comparison to the Gupta et. al. algorithm. Upon extracting the data from the simulations we found that there were some jobs which were severely throwing the average finish time for both the VSP and Gupta algorithms. The few jobs which have been removed have both a very low weight and a very large size. After removing these we recalculate the averages and have reported this in Figure 3.

The results prove to be very interesting. They show us that the VSP platform is very promising; this has allowed us to combine the Gupta et. al. algorithm with the VSP platform and achieve very competitive results. Although the energy consumption is larger, the ∑Flow + Energy is lower. This is the aim of the algorithm, to minimise the ∑Flow + Energy so this has been very successful.

6. CONCLUSION

In this paper we present the Virtual Single Processor approach to low energy scheduling for power heterogeneous multiprocessor computer systems. We show that the VSP approach is theoretically sound and can be used with existing technologies soliciting strong results. In future papers we hope to expand this area by looking at variations of the VSP system that are split into sub VSPs some that are designed to handle large / sequential jobs and others for smaller parallel jobs.

7. REFERENCES

[1] Albers, S. and Fujiwara, H.: Energy-Efficient Algorithms for Flow Time Minimization. ACM Transactions on Algorithm, 3(4), 49 (2007)

[2] Andrew, L. and Wierman, A. and Tang, A.: Optimal speed scaling under arbitrary power functions. ACM SIGMETRICS Performance Evaluation Review 37(2), 39–41 (2009)

[3] Bansal, N. and Chan, H.L. and Pruhs, K.: Speed Scaling with an Arbitrary Power Function. Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms, 693–701 (2009)

[4] Bower, F.A. and Sorin, D.J. and Cox, L.P.: The Impact of Dynamically Heterogeneous Multicore Processors on Thread Scheduling. IEEE Micro 28(3), 17–25 (2008)

[5] Gupta, A. and Krishnaswamy, R. and Pruhs, K.: Scalably Scheduling Power-Heterogeneous Processors. International Conference on Green Computing, 165-173, International Conference on Green Computing, 2010

[6] Gupta, A. and Krishnaswamy, R. and Pruhs, K.: Nonclairvoyantly Scheduling Power-Heterogeneous Processors. Proceedings of the 37th International Colloquium of Automata, Languages and Programming, 312–323 (2010)

[7] Lam, Tak-Wah and Lee, Lap-Kei and To, Isaac K. K. and Wong, Prudence W. H.: Competitive Non-migratory Scheduling for Flow Time and Energy. Proceedings of the twentieth annual symposium on Parallelism in algorithms and architectures, 256–264 (2008)

[8] Lam, Tak-Wah and Lee, Lap-Kei and To, Isaac K. K. and Wong, Prudence W. H.: Improved multi-processor scheduling for flow time and energy. Journal of Scheduling, November, Online First (2009)

[9] Yao, F. and Demers, A. and Shenker, S.: A scheduling model for reduced CPU energy. Proceedings of the 36th Annual Symposium on Foundations of Computer Science (FOCS'95) 374–382 (1995)

A linear programming approach for adaptive synchronization of traffic signals

Pablo Coll * Pablo Factorovich * Irene Loiseau *

* Departamento de Computación
Facultad de Ciencias Exactas y Naturales,Universidad de Buenos Aires
{pecoll,pfactoro,irene}@dc.uba.ar

ABSTRACT

As traffic congestion during rush hours is a growing problem for most cities, there is an increasing need for more effective managing traffic signal control and traffic assignment systems. We present here a new adaptive system based on a linear programming model for the signal control problem, having as objective to minimize the total length of the queues of cars waiting at each corner. The model is intended to be fed with traffic information provided by real-time sensors installed at each intersection. In order to compare the performance of our program with that of the current scheduling designed by the transit office of Buenos Aires city, we used a traffic simulation system and real traffic flow data of a pilot area of the city. Preliminary results are very promising.

Keywords: Urban traffic control, Adaptive signal control, Signal timing optimization, Linear programming

1. INTRODUCTION

Traffic signal control are systems for synchronizing the timing of any number of traffic signals in a certain area.Despite the research done in the field of urban traffic control systems since around five decades, there is still an increasing need for more effective managing of traffic control systems. None of the proposals we were able to found in the literature solves the complete optimization problem for big urban areas. As this is a very difficult problem authors handle only part of the problem, or propose hierarchical models that divide the problem in parts that are addressed independently. Some studies focus on small regions or in a single intersection.

Main decisions in signal control strategies at urban areas include determining the duration of the complete cycle and the duration of green lights at each direction of every intersection. Several mathematical and computational approaches have been proposed, most of them based on heuristics. Existing exact models for traffic signal control are very limited in scope but they are useful for providing insight into the problem and examining the performance of heuristics.

Most of the currently implemented traffic control systems may be grouped into three principal categories:

- Fixed time strategies, that are derived off-line by use of codes based on historical data.
- Traffic-responsive strategies make use of real-time measurements to calculate in real time the suitable signal settings.
- Predictive strategies are based on off-line and on-line information.

It is no possible to survey here all the work done in this vast research area, so we will mention only a few selected references. A review on traffic control strategies can be found at [1]. Also Cheng et al.[2], Dotoli et al. [3] and Wey[4] present an overview of available traffic control methods.

Some of the approaches are already implemented in real life while other reflect work still at the research and development stage. Among the commercial systems we can mention TRANSYT [5] which was first developed by Robertson [5] and was substantially improved later. It uses historical information and computes signal control schemes off-line. SCOOT [6, 7] includes a network model that is fed with real data and is run repeatedly to investigate the effect of incremental changes of splits, offsets and cycle times. Changes are implemented if they show to be beneficial. RODHES [8], PRODYN [9] and OPAC [10] developed more rigorous model-based traffic responsive strategies. RODHES and PRODYN solve in real time optimization problems employing dynamic programming and OPAC employs exhaustive enumeration. So the three of them are real-time feasible for only one intersection and they end with a decentralized optimal strategies coordinated heuristically by a superior layer. In TUC [11, 12] a store-and-forward strategy is implemented. The main idea is to simplify the model in order to be able to describe the traffic flow process without using discrete variables. The optimization part of the system requires to solve a quadratic programming problem.

Wey [4] presents an integer linear model for the network wide signal optimization problem and a modification of the network simplex algorithm to solve it. The proposal is tested in a five intersections area and compared with exact solution to the MIP. Lo [13] models the traffic flow conditions using a cell transmission model (CTM) based in hydrodynamic concepts. The resulting model for the dynamic signal-control problem is a mixed-integer linear programming program. A two intersections network is used to demonstrate applicability of the formulation. Lin and Wang [14] propose an enhanced 0-1 formulation based also on the CTM model. Also He et al. [15] propose heuristics based on the linear relaxation of the model for solving CTM based MIP formulations. They tested the approach on examples of one or two intersections.

Barisone et al. [16] propose an elaborated real-time nonlinear optimization model. They report having successfully tested it in a urban area of Genova consisting of 18 links. Dotoli et al. [3] modify this model to take into account the presence of pedestrians, different levels of traffic consgetion, vehicle classification, etc. Their case study is an area of two consecutive intersections with heavy traffic. Cheng et al.[2] presents a parallel algorithm for the problem of finding optimal coordinate signal timing plans for a large area based on game theory. They test their algorithm in a real area of 75 intersections, and they claim that using a thousand CPUŽs they found a signal planning in less than 10 minutes. Aboudulas et al.[17] propose a methodology based on store-and-forward traffic model, mathematical optimization and optimal control for real-time signal control in congested large-scale urban traffic networks.

In [18] authors describe a genetic algorithm applied to the coordination of signals in an urban network based on real-time traffic information. They evaluate they approach on a 12 intersections area, using the CORSIM traffic simulation software, and report that they reduce average delay times in a 15%. There are sev-

eral other heuristics approaches developing algorithms based on genetic algorithms, ant colony optimization, Markov processes or neural networks.

Systems already implemented in real life can be evaluated through the improvement of traffic conditions in cities or areas were they are installed. Other approaches are difficult to compare between them, as they show results in different small real or generated ad-hoc case studies.

Our goal was to develop an automatic system that determines in real-time the cycles of the traffic signals in a region of the city, based on information provided by sensors installed at each up-stream corner. The system is continuously fed with information provided by these sensors. Based on the results of the optimization, the planning is automatically modified.

Optimization is done by means of a linear programming model having as objective to minimize the total length of the queues of cars waiting at each corner. Part of our model was derived from some of the equations of the pioneer Robertson model for the platoon dispersion ([5], see also [4]). This model represents vehicles moving in platoons that flow from one corner to the other till they are out of the area . We adapted the equations describing the number of vehicles waiting at the queues, but equations relating neighbor intersections are not included in our model. We also added constraints that bind the red and green lights at each intersection. Bounds that prevent sudden changes on the length of the cycles and the green lights, are also defined.

Several cycles of the traffic signals are considered at each run of the optimization program.

2. MODEL FORMULATION

In this section we define the variables, parameters and constraints and we describe the LP model in detail. We consider an area that consists of a set $\mathscr{N} = \{1 \dots N\}$ of intersections. At each intersections there is a variable number of $J(n)$ directions from which traffic flow arrives, so there are $J(n)$ signals that have to be regulated.

A cycle of the traffic signal is the sum of the lasting times of the green and "not green" phases. The temporal horizon of the model is a predetermined number of complete cycles of the signals $\mathscr{H} = \{1 \dots H\}$. The optimization program is run with the information of the last H cycles. When new data is received from the sensors the oldest cycle is discarded. Each phase (green and "not green") of the cycle is partitioned in identical sets $\mathscr{I} = \{1 \dots I\}$ of time intervals. The decision variables are the duration of each cycle at each corner and the duration of the green lights at each direction. The model implicitly calculates an offset that represents the beginning of each cycle related to an arbitrary initial time.

We need to distinguish which are the cycles starting in green and which are those starting in red, so we define

$RS = \{(n, j)/\text{cycle at (n,j) starts in red}\}$.

As there are several types of intersections in a city, we need to provide the following definitions. Let $JVR(n, j)$ be the set of directions related to the intersection n, for which its greens lights do not overlap and jointly define the time when light of direction (n, j) is red. The simplest and more typical situation is an intersection with two crossing directions j and k. In this case we have $JVR(n, j) = \{k\}$ and $JVR(n,k) = \{j\}$ and if $(n, j) \in RS$, we will have to force $R^h_{n,j} = V^h_{n,k}$; otherwise, $R^{h+1}_{n,j} = V^h_{n,k}$. Analogously, let $JVV(n, j)$ be the set of directions related to the intersection n in which green light times do not overlap and jointly define the time when light (n, j) is green. Here a typical situation is related to the two opposite directions in a two way street, which are usually in green simultaneously. Let us note that $JVV(n, j)$ can be an empty set for some j, but j have to belong to some $JVR(n,k)$ or $JVV(n,k)$ for another direction k. Finally, let $JRR(n, j)$ be a set of directions related to the intersection n, in which red light times are non overlapped and jointly define the time when light of (n, j) direction is red.

2.1. Variables

Variables related to time are:

- $V^h_{n,j}$: length of the green at intersection $n \in \mathscr{N}$, direction $j \in J(N)$ and cycle $h \in \mathscr{H}$
- $R^h_{n,j}$: length of "not green" at intersection $n \in \mathscr{N}$, direction $j \in J(N)$ and cycle $h \in \mathscr{H}(n, j)$

Then $C^h_n = R^h_{n,j} + V^h_{n,j}$ for any direction $(j,n) \in RS$, is the total length of cycle $h \in \mathscr{H}$ at intersection $n \in \mathscr{N}$. If $(n, j) \in RS$, variable $R^{h+1}_{n,j}$ appears instead of $R^1_{n,j}$ at the model. We define for each (n, j) the set of indexes h, $\mathscr{H}(n, j)$, to represent these situations.

Variables related to number of vehicles are:

- $LV^{h,i}_{n,j}$: queue length during green light at intersection $n \in \mathscr{N}$, direction $j \in J(n)$, cycle $h \in \mathscr{H}$ and interval $i \in \mathscr{I}$
- $LR^{h,i}_{n,j}$: queue length during red ("not green") light at intersection $n \in \mathscr{N}$, direction $j \in J(n)$, cycle $h \in \mathscr{H}(n, j)$ and interval $i \in \mathscr{I}$
- $Q^{h,i}_{n,j}$: outcoming flow of vehicles at intersection $n \in \mathscr{N}$, direction $j \in J(n)$, cycle $h \in \mathscr{H}$ and interval $i \in \mathscr{I}$.

All variables of the model are allowed to take non integer values.

2.2. Coefficients

- As interval lasting $R^{h,i}_{n,j}$ is a result of the optimization, $\widehat{ER^{h,i}_{n,j}}$ is an estimation of the information that should be given by sensors at direction j intersection n, cycle h, interval i during red light. This estimation is based on values obtained from the sensors after executing the planning obtained from the last run of the model.
- $\widehat{EV^{h,i}_{n,j}}$ is the same estimation for green lights.
- $CI_{n,j}$ represents the length of the queue at the beginning of the period to be optimized.
- $S_{n,j}$ is the flow capacity of a street section (in number of cars).
- α is a positive parameter of the objective function.

2.3. The model

The linear programming formulation we propose is the following:

$$\min \sum_{n=1}^{N} \sum_{j=1}^{J(n)} \sum_{h=1}^{H} \sum_{i=1}^{I} LR^{h,i}_{n,j} + LV^{h,i}_{n,j} - \alpha Q^{h,i}_{n,j}$$

subject to the following constraints $\forall n \in \mathscr{N}, j \in J(n)$:

$$LR^{h,1}_{n,j} = LV^{h-1,I}_{n,j} + \frac{\widehat{ER^{h,1}_{n,j}}}{I} R^{h}_{n,j} \qquad \forall h \in \mathscr{H}(n,j), h \geq 2 \qquad (1)$$

$$LV^{h,1}_{n,j} = LR^{h,I}_{n,j} + \frac{\widehat{EV^{h,1}_{n,j}}}{I} V^{h}_{n,j} - Q^{h,1}_{n,j} \quad \forall h \in \mathscr{H} \cap \mathscr{H}(n,j) \qquad (2)$$

$$LR^{h,i}_{n,j} = LR^{h,i-1}_{n,j} + \frac{\widehat{ER^{h,i}_{n,j}}}{I} R^{h}_{n,j} \qquad \forall h \in \mathscr{H}(n,j), i \in \{2 \dots I\} \qquad (3)$$

$$LV^{h,i}_{n,j} = LV^{h,i-1}_{n,j} + \frac{\widehat{EV^{h,i}_{n,j}}}{I} V^{h}_{n,j} - Q^{h,i}_{n,j} \quad \forall h \in \mathscr{H}, i \in \{2 \dots I\} \qquad (4)$$

$$LV^{1,1}_{n,j} = CI_{n,j} + \frac{\widehat{EV^{1,1}_{n,j}}}{I} V^{1}_{n,j} - Q^{1,1}_{n,j} \quad \forall (n,j) \notin RS \qquad (5)$$

$$LR^{1,1}_{n,j} = CI_{n,j} + \frac{\widehat{ER^{1,1}_{n,j}}}{I} R^{1}_{n,j} \qquad \forall (n,j) \in RS \qquad (6)$$

$$Q^{h,i}_{n,j} \leq \frac{S_{n,j}}{I} V^{h}_{n,j} \qquad (7)$$

$$-\delta_V \leq V^{h}_{n,j} - V^{h-1}_{n,j} \leq \delta_V \qquad (8)$$

$$-\delta_R \leq R^{h}_{n,j} - R^{h-1}_{n,j} \leq \delta_R \qquad (9)$$

$$V_{Min} \leq V^{h}_{n,j} \leq V_{Max} \qquad (10)$$

$$R_{Min} \leq R^{h}_{n,j} \leq R_{Max} \qquad (11)$$

$$R^{h}_{n,j} = \sum_{k \in JVR(n,j)} V^{h}_{n,k} \qquad \forall h \in \mathscr{H}, \ (n,j) \in RS \qquad (12)$$

$$R^{h}_{n,j} = \sum_{k \in JVR(n,j)} V^{h-1}_{n,k} \qquad \forall h \in \mathscr{H}(n,j), \ (n,j) \notin RS \qquad (13)$$

$$V^{h}_{n,j} = \sum_{k \in JVV(n,j)} V^{h}_{n,k} \qquad \forall h \in \mathscr{H} \qquad (14)$$

$$R^{h}_{n,j} = \sum_{k \in JRR(n,j)} R^{h}_{n,k} \qquad \forall h \in \mathscr{H}(n,j) \qquad (15)$$

$$V^{h}_{n,j}, \ R^{h}_{n,j}, \ LR^{h,i}_{n,j}, \ LV^{h,i}_{n,j}, \ Q^{h,i}_{n,j} \geq 0 \qquad (16)$$

The objective is to minimize the sum of the queue lengths and to maximize the car flow in the area. α is a parameter of the model included to force the car outgoing flow at each corner, in order to make equation (7) as tight as possible.

Constraints (1) to (6) define the new queue lengths for every interval by adding to the previous length the car arrivals and subtracting the outgoing flow of vehicles. These equations are inspired in part of the Robertson model. However, instead of using information of neighbor intersections, arrivals are computed multiplying estimations of number of arriving cars per second by the length of the interval. Constraints (1) and (2) define the length of the queues for intervals $i = 1$, which is the first time interval for the current light color. Constraints (3) and (4) compute the length of the queue in contiguous intervals corresponding to the same light color. Finally, (5) and (6) set initial values of the queues.

Constraint (7) sets a bound, based on street capacity, for the outgoing flow from direction j at intersection n. Constraints (8) and (9) impose smoothness in the changes of light schedules, which is very important, not only for safety, but also to guarantee the quality of estimators $\widehat{ER^{h,i}_{n,j}}$ and $\widehat{EV^{h,i}_{n,j}}$. (10) and (11) are bounds on light duration and constraints (12) to (15) establish the necessary coordination among traffic lights in different directions at each corner. Depending on each intersection structure, some of the equations (12) to (15) can be redundant. For the sake of clarity, we chose to write the model as above instead of using a more complicated notation that could have avoid redundancy in all cases.

3. EXPERIMENTAL RESULTS

Solution was implemented as a C library that takes care of communication between hub, sensors and signals and calls the LP solver. The hub sends sensors information to the library and receives next planing for every semaphore.

As the sensors have not been installed yet, we tested our model by means of a microscopic traffic simulation software package TSIS-CORSIM [19](Figure 1).

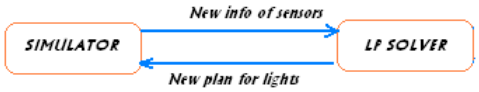


Figure 1: *Adaptive System Scheme*

In order to compare the performance of our program with the current scheduling designed by experts from the transit office of Buenos Aires city, we used real traffic flow data of a pilot area of the city (7 intersections, some one direction streets, some avenues).(Figure 2) We have tested our model in eight different scenarios obtained

Figure 2: *Pilot area*

varying green lights lower bounds and traffic flow rates.

Preliminary results show that our optimization tool outperforms the current system reducing the average queue length in 4,17% at rush hours. Also on tests with 50% and 25% of the current traffic flow, an improvement of 16,49% and 32.72°% respectively, was obtained. Beside this we have tried with the following alternate flow: 150% of rush hour flow in N and E direction and 66% in the S and W. Results of experiments with green lights lower bounds of 30 sec. can be seen at (Figure 3).

Parameters used to perform the tests were: $\alpha = 0.033$, $H = 2$, $I = 5$, $CI_{n,j}$ taking values from simulation using current lights plan and $S_{n,j} = 2$ for every (n,j) except for those corresponding to Vernet Avenue, in which case the value is 4 since the other streets has 2 lanes and Vernet has 4.

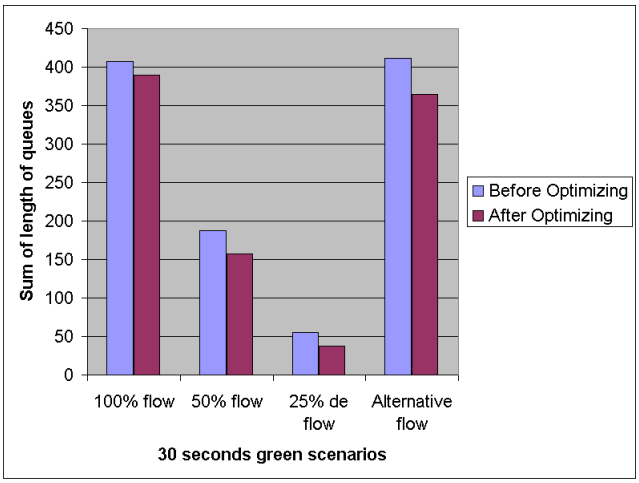


Figure 3: *Preliminary results*

4. CONCLUSIONS

We developed an adaptive system based on a linear programming model for the synchronization of traffic signals at an urban area. The system gets real-time traffic information from sensors installed at each intersection. According to the results of the optimization the system will automatically coordinate traffic lights duration in an area of the city. Preliminary results show that this proposal can be integrated on an efficient tool for traffic congestion management. As the optimization is done by means of an LP program (no integer variables), computational times are affordable in the frame of a real time system. This is the main advantage of our proposal.

5. REFERENCES

[1] M. Papageorgiou, C. Diakaki, V. Dinopoulou, A. Kotsialos, and Y. Wang, "Review of road traffic control strategies," *Proceedings of the IEEE*, vol. 91, no. 12, pp. 2043–2067, 2003.

[2] S.F. Cheng, M.A. Epelman, and R.L. Smith, "Cosign: a parallel algorithm for coordinated traffic signal control," *IEEE Transactions on Intelligent Transportation Systems*, vol. 7, no. 4, pp. 551–564, 2006.

[3] M.Dotoli, M. Fanti, and C. Meloni, "A signal timing plan formulation for urban traffic control," *Control Engeneering Practice*, vol. 14, pp. 190–192, 2006.

[4] W.M. Wey, "Model formulation and solution algorithm of traffic signal control in an urban network," *Computers, Environment and Urban Systems*, vol. 2, pp. 355–377, 2000.

[5] D.I. Robertson, "Transyt method for area traffic control," *Traffic Engeneering and Control*, vol. 10, pp. 276–281, 1969.

[6] P.B. Hunt, D.I. Robertson, R.D. Bretherson, and M.C. Royle, "The scoot on-line traffic signal optimization technique," *Traffic Engeneering and Control*, vol. 23, pp. 190–192, 1982.

[7] D. Bretherton, M. Bodger, and N. Baber, "Scoot-the future," in *Proceedings of the 12th IEE International Conference on Road Transport Information and Control*, London, UK, 2004, pp. 301–306.

[8] P. Mirchandani and L. Head, "Rodhes: a real-time traffic signal control system:architecture, algorithms and analysis," *Transportation Research Part C*, vol. 9, no. 6, pp. 415–432, 2004.

[9] J.LFarges, J. Henry, and J. Tufal, "The prodyn real-time traffic algorithm," in *Proceedings 4th IFAC Symposium on Transportation Systems*, 1983, pp. 307–312.

[10] N.H. Gartner, "Opac: a demand -responsive strategy for traffic signal control system:architecture, algorithms and analysis," *Transportation Research Record*, vol. 906, pp. 75–84, 1983.

[11] C. Diakaki, M. Papageorgiou, and K. Aboudolas, "A multivariate regularor approach to traffic-responsive network-wide signal control," *Contro engeneering Parctice*, vol. 10, pp. 183–195, 2002.

[12] V. Dinopoulou, C. Diakaki, and M. Papageorgiou, "Applicatons of the urban traffic control strategy tuc," *European Journal of Operational Research*, vol. 175, pp. 1652–1665, 2006.

[13] H. Lo, "A cell-based traffic control formulation: strategies and benefits of dynamic timing plans," *Transportation Science*, vol. 35, no. 2, pp. 148–164, 2001.

[14] W.H. Lin and C. Wang, "An enhanced 0-1 mixed-integer lp formulation for traffic signal control," *IEEE Transactions on Intelligent Transportation Systems*, vol. 5, no. 4, pp. 238–245, 2004.

[15] Q. He, W. Lin, H. Liu, and L. Head, "Heuristic algorithms to solve 0-1 mixed integer lp formulations for traffic signal control problems," in *Proceedings 2010 IEEE International Conference on Service Operations and Logistics and Informatics (SOLI 2010)*, Qingdao, China, 2010, pp. 118–124.

[16] A. Barisone, D. Giglio, R. Minciardi, and R. Poggi, "A macroscopic traffic model for real-time optimization of signalized urban areas," in *Proceedings of the 41st IEEE Conference on Desion and Control*, Las Vegas, USA, 2002, pp. 900–903.

[17] K. Aboudolas, M. Papageorgiou, and E. Kosmatopoulos, "Store and forward based methods for the signal control problem in large-scale congested urban road networks," *Transportation Research Part C*, vol. 17, pp. 163–174, 2009.

[18] L. Zang and L. Jia, "Modelling and simulation of traffic signal control for urban network," in *Proceedings of the IEEE Third Internatonal Symposium on Intelligent Information Technoloty Application*, China, 2009, pp. 253–256.

[19] http://mctrans.ce.ufl.edu/featured/tsis.

List of Authors