



**EURO**  
The Association of European  
Operational Research Societies



UNIVERSIDAD  
POLITECNICA  
DE VALENCIA



# ORP<sup>3</sup>

OPERATIONAL RESEARCH  
PERIPATETIC  
POSTGRADUATE  
PROGRAMME

6-10 September 2005, Valencia, Spain.

## Proceedings

Edited by:

Concepción Maroto  
Rubén Ruiz  
Javier Alcaraz  
Eva Vallada  
Fortunato Crespo

ORP3 2005

Operational Research Peripatetic Post-Graduate Programme  
A EURO conference for young OR researchers and practitioners  
6-10 September 2005, Valencia, Spain.

I.S.B.N: 84-689-3077-6

D.L.: V-3130-2005

Address of the Editors: Departamento de Estadística e Investigación Operativa  
Aplicadas y Calidad  
Universidad Politécnica de Valencia  
Camino de Vera s/n  
46021 Valencia, Spain  
tel: +34 96 387 74 90, fax: +34 96 387 74 99  
email: [cmaroto@eio.upv.es](mailto:cmaroto@eio.upv.es)

Published by: ESMAP, S.L.

Cover drawing: Carmen Lloret  
email: [clloret@dib.upv.es](mailto:clloret@dib.upv.es)

---

# Scientific Committee

---

David Alcaide López de Pablo  
Universidad de La Laguna  
Spain

Javier Alcaraz Soria  
Universidad Politécnica de Valencia  
Spain

Ramón Álvarez Valdés Olaguíbel  
Universitat de València  
Spain

Denis Bouyssou  
CNRS - LAMSADE. Université  
Paris-Dauphine  
France

Fortunato Crespo Abril  
Universidad Politécnica de Valencia  
Spain

Laureano Escudero Bueno  
Universidad Miguel Hernández. Centro de  
Investigación Operativa  
Spain

Horst W. Hamacher  
Universität Kaiserslautern  
Germany

Concepción Maroto Álvarez (Chair)  
Universidad Politécnica de Valencia  
Spain

Ethel Mokotoff Miguel  
Universidad de Alcalá  
Spain

Luis Paquete  
Technische Universität Darmstadt  
Germany

Jesús Pastor Ciurana  
Universidad Miguel Hernández. Centro de  
Investigación Operativa  
Spain

Marie Claude Portmann  
École des Mines de Nancy (INPL)  
France

Joaquín Sicilia Rodríguez  
Universidad de La Laguna  
Spain

Thomas Stützle  
Technische Universität Darmstadt  
Germany

Rubén Ruiz García  
Universidad Politécnica de Valencia  
Spain

Enriqueta Vercher González  
Universitat de València  
Spain

---

# Organization Committee

---

Andrés Carrión García

José Jabaloyes Vivas

Fortunato Crespo Abril

Rubén Ruiz García (Chair)

José Miguel Carot Sierra

Eva Vallada Regalado

Gerardo Minella

Elena Vázquez Barrachina

Juan Carlos García Díaz

---

# Referees

---

The editors and scientific committee would like to thank the following referrees:

Carlos Andrés Romano  
Valerie Belton  
José D. Bermúdez Edo  
Andrés Carrión García  
Emilio Carrizosa Priego  
Marco Chiarandini  
Marcos Colebrook Santamaría  
Erik Demeulemeester  
Javier Faulín Fajardo  
Juan Carlos García Díaz  
José Pedro García Sabater  
José Miguel Gutiérrez Expósito  
Martine Labbé  
María Teresa León Mendoza  
Fermín Fco. Mallor Giménez  
Alfredo Marín Pérez

Nieves Martínez Alzamora  
Pedro Mateo Collazos  
Manel Mateu Doll  
Ana Meca Martínez  
Stefan Nickel  
José Parreño Fernández  
Rafael Pastor Moreno  
Joaquín Pérez Navarro  
Marco Pranzo  
Justo Puerto Albandoz  
Andrés Ramos Galán  
Cesar Rego  
Susana San Matías Izquierdo  
Pedro Sánchez Martín  
Baldomero Segura García del Río  
Elena Vázquez Barrachina



---

# Contents

---

Preface . . . . .	ix
1 Heuristic procedures for generating stable project baseline schedules <i>Stijn Van de Vonder, Erik Demeulemeester and Willy Herroelen</i> . . . . .	11
2 Exact Solution Procedures for the Balanced Unidirectional Cyclic Layout Problem <i>Temel Öncan and İ.Kuban Altınel</i> . . . . .	21
3 Multiobjective service restoration in electric distribution networks using a local search based heuristic <i>Vinícius Jacques Garcia and Paulo Morelato França</i> . . . . .	35
4 The Traveling Salesman Problem with Time-Dependent Costs: an exact approach <i>José Albiach, José María Sanchis and David Soler</i> . . . . .	45
5 Dimensioning and designing shifts in a call center <i>Cyril Canon, Jean-Charles Billaut and Jean-Louis Bouquard</i> . . . . .	55
6 A new concept of approximate efficiency in multiobjective mathematical programming <i>César Gutiérrez, Bienvenido Jiménez and Vicente Novo</i> . . . . .	65
7 An Efficient Approach for Solving the Production/Ordering Planning Problem with Time-varying Storage Capacities <i>José Miguel Gutiérrez, Antonio Sedeño-Noda, Marcos Colebrook and Joaquín Sicilia</i> . . . . .	75
8 Optimizing the service capacity by using a speed up simulation <i>Isolina Alberto, Fermín Mallor and Pedro M. Mateo</i> . . . . .	85
9 Exact Algorithms for Procurement Problems under a Total Quantity Discount Structure <i>D.R. Goossens, A.J.T. Maas, F.C.R. Spieksma and J.J. van de Klundert</i> . .	93
10 Designing Reliable Systems with SREMS++ <i>Angel Juan, Javier Faulín, Vicente Bargueño and Anita Goyal</i> . . . . .	115
11 Nonconvex optimization using an adapted linesearch <i>Alberto Olivares, Javier M. Moguerza and Francisco J. Prieto</i> . . . . .	127
12 A multi-criteria and fuzzy logic based approach for the relative assessment of the fire hazards of chemical substances and installations <i>Apostolos N. Paralikas and Argyrios I. Lygeros</i> . . . . .	141

13	Supply Chain Games <i>Federico Perea</i> . . . . .	153
14	Hybrid Supply Chain Modelling - Combining LP-Models and Discrete-Event Simulation <i>Margaretha Preusser, Christian Almeder, Richard F. Hartl, and Markus Klug</i>	163
15	Tolerance-based Branch and Bound Algorithms <i>Marcel Turkensteen, Diptesh Ghosh, Boris Goldengorin and Gerard Sierksma</i>	171
16	Decision support system for Attention Deficit Hyperactivity Disorder diagnostics <i>Iryna Yevseyeva, Kaisa Miettinen and Pekka Räsänen</i> . . . . .	183
17	Optimality conditions in preference-based spanning tree problems <i>Miguel Ángel Domínguez-Ríos, Sergio Alonso, Marcos Colebrook and Antonio Sedeño-Noda</i> . . . . .	195
18	Robust 1-median location problem on a tree <i>Rim KALAI, Mohamed Ali ALOULOU, Philippe VALLIN and Daniel VANDERPOOTEN</i> . . . . .	201
19	Models and Software for Improving the Profitability of Pharmaceutical Research <i>Jiun-Yu Yu and John Gittins</i> . . . . .	213
20	Application of U-Lines principles to the Assembly Line Worker Assignment and Balancing Problem (UALWABP). A model and a solving procedure <i>Cristóbal Miralles, José Pedro García and Carlos Andrés</i> . . . . .	227
21	Modelling and Forecasting Spanish Mortality <i>Ana Debón Aucejo and Francisco Puig Blanco</i> . . . . .	235
22	Planning holidays and working time under annualised hours <i>Amaia Lusa, Albert Corominas and Rafael Pastor</i> . . . . .	243
23	Soft computing-based aggregation methods for human resource management <i>Lourdes Canós and Vicente Liern</i> . . . . .	251
24	Cutting Plane and Column Generation for the Capacitated Arc Routing Problem <i>David Gómez-Cabrero, José Manuel Belenguer and Enrique Benavent</i> . . . . .	259
25	GRASP and Path Relinking for Project Scheduling under Partially Renewable Resources <i>Fulgencia Villa, Ramón Alvarez-Valdes, E. Crespo and J. Manuel Tamarit</i> .	267
26	Evaluation of a hierarchical production planning and scheduling model for a tile company under different coordination mechanisms <i>María del Mar Alemany, Eduardo Vicens, Carlos Andrés and Andrés Boza</i> .	283
27	A Restricted Median Location Model for Stop Location Design in Public Transportation Networks <i>Dwi Retnani Poetranto</i> . . . . .	297
28	A Column Generation Approach to the Capacitated Vehicle Routing Problem with Stochastic Demands <i>Christian H. Christiansen and Jens Lysgaard</i> . . . . .	311
29	Integrating nurse and surgery scheduling <i>Jeroen Beliën and Erik Demeulemeester</i> . . . . .	319
30	A bi-objective coordination setup problem in a two-stage production system <i>Michele Ciavotta, Paolo Detti, Carlo Meloni and Marco Pranzo</i> . . . . .	335

31	Two unifying frameworks in voting theory <i>Estefanía García, José Luis Jimeno and Joaquín Pérez</i>	345
32	Real time management of a metro rail terminus <i>Marta Flamini and Dario Pacciarelli</i>	357
33	Optimization models for the delay management problem in public transportation <i>Géraldine Heilporn, Luigi De Giovanni and Martine Labbé</i>	367
34	A Metaheuristic Approach for the Vertex Coloring Problem <i>Enrico Malaguti, Michele Monaci and Paolo Toth</i>	377
35	Investigating inventory control tactics in two node capacitated supply chains <i>Georgia Skintzi, Gregory Prastacos and George Ioannou</i>	387
36	Ejection Chain Algorithms for the Traveling Salesman Problem <i>D. Gamboa, C. Osterman, C. Rego and F. Glover</i>	403
37	Parallel machine scheduling with resource dependent processing times <i>Raúl Cortés, Jose Pedro García, Rafael Pastor and Carlos Andrés</i>	413
38	A Tabu Search algorithm for two-dimensional non-guillotine cutting problems <i>Francisco Parreño, Ramón Alvarez-Valdes and J. Manuel Tamarit</i>	417
39	Problem of time-consistency in model of Kyoto Protocol realization <i>Maria Dementieva, Pekka Neittaanmäki and Victor Zakharov</i>	429



---

# Preface

---

Every Operations Research Congress is important, whether national, international, general or specific. Sharing knowledge, ideas and experiences with close and distant peers is an inseparable aspect from our work as operation researchers dedicated to finding solutions to decision-making problems that currently challenge enterprises and organisations around the world. ORP3 is also an important and very special congress. It is important because it is born under the auspices of the European society EURO which ensures scientific rigor in both paper selection and the level of participants, since they are not only required to present their own scientific work, but must also discuss the work of a colleague. And it is special because it seriously raises the subject of the introduction of future OR scientists to the transfer of knowledge and to the notion of respect towards the work of fellow researchers. It is therefore an important and special congress because of its contribution to the training of future OR researchers who, in the next decades, could and should in turn contribute to the improvement of our society by increasing the competitiveness of enterprises, public services and organisations while keeping the professional ethics that characterises experts in the field.

This third edition of the ORP3 also includes the lessons of two well known senior researchers from our field through tutorials aimed to help young researchers understand operations research classic and recent techniques. We would like to thank Denis Bouyssou and Thomas Stützle for their valuable contribution to the congress.

The Scientific Committee, together with a large number of assisting referees, have revised each of the numerous papers presented twice. The 40 young researchers whose papers have been selected are from 14 different countries and offer an accurate representation of operations research techniques. Therefore, this volume compiles various contributions, from new solutions to classic problems such as the travelling salesman problem, to the newest applications of game theory to the Kyoto protocol. More specifically, this volume includes papers dealing with optimization problems, scheduling problems, manufacturing systems, vehicle routing, manpower planning, multicriteria decision making, set covering and inventory. Furthermore, several different techniques have been used, and among them: integer programming, dynamic programming, Branch & Bound methods, simulation, multiobjective algorithms, cooperative games and metaheuristics.

As usual in ORP3 congresses, the organization has been left to a group of young OR re-

searchers. The particular characteristics of ORP3 turned the organization of the congress into a real challenge, since accommodation and full board were included for all participants in the 200 Euro registration fee. In order to comply with this requirement and to provide accommodation minutes away from where the congress will be held, the residence hall Galileo Galilei at the Universidad Politécnica of Valencia main campus site has been chosen. The Committee has organized two interesting and hopefully fun visits within the social programme for the congress: A guided tour around historical Valencia and its landmarks and a guided visit to the natural park “Albufera”, an ecologically valuable site and the most important wetland in the Valencian Community. Participants will be able to enjoy a tipically Valencian dinner in a “Barraca” (traditional small farmhouse), encouraging a friendly atmosphere among peers.

And last, on behalf of the Scientific Committee and the Organization Committee, we would like to thank EURO for placing their trust in the Operations Research Group and in the Universidad Politécnica of Valencia for the organization of the third ORP3 congress. Likewise, we would like to thank all the referees for their effort in revising the papers of their particular fields of expertise. We hope that between all of us, young and senior researchers, we will be able to offer a small contribution to the development of Operations Research and to its impact on the development of society within the next decades.

Valencia, June 24th, 2005,

Concepción Maroto  
Chair of the Scientific Committee

Rubén Ruiz  
Chair of the Organization Committee

# Heuristic procedures for generating stable project baseline schedules

Stijn Van de Vonder, Erik Demeulemeester and Willy Herroelen

Center for Operations Management, K.U.Leuven

Naamsestraat 69, B-3000 Leuven (Belgium)

Email: Stijn.VandeVonder@econ.kuleuven.ac.be@econ.kuleuven.be

**Abstract**— Solution robust project scheduling is a growing research field aiming at constructing proactive schedules to cope with multiple disruptions during project execution. When stochastic activity durations are considered, including time buffers between activities is a proven method to improve the stability of a baseline schedule.

This paper introduces multiple algorithms to include time buffers in a given schedule while a predefined project due date remains respected. Multiple efficient heuristic and meta-heuristic procedures are proposed to allocate buffers throughout the schedule. An extensive simulation-based analysis of the performance of all algorithms is given. The impact of the activity duration variance structure on the performance is discussed in detail.

**Keywords**— Project scheduling, uncertainty, stability, buffers

## I. INTRODUCTION

THE vast majority of the research efforts in project scheduling over the past several years have concentrated on the development of exact and heuristic procedures for the generation of a workable *baseline schedule* (*pre-schedule* or *predictive schedule*) assuming complete information and a static and deterministic environment. During execution, however, a project may be subject to considerable uncertainty, which may lead to numerous schedule disruptions. Activities can take shorter or longer than primarily expected, resource requirements or availability may vary, new activities might have to be inserted, etc..

Recent research [12] has demonstrated that when projects have to be executed in the face of uncertainty, proactive-reactive project scheduling procedures are capable of combining schedule stability and makespan performance and the use of an objective function aiming at schedule stability pays off. The objective of this paper is to develop and validate a number of heuristic procedures for generating stable project baseline schedules.

The problem used as our vehicle of analysis can be described as follows. A project network  $G = (N, A)$  is

represented in activity-on-the-node representation with dummy start and end nodes. All non-dummy project activities have stochastic activity durations  $d_j$ , are subject to finish start zero-lag precedence constraints and require an integer per period amount  $r_{jk}$  of one or more renewable resource types  $k$  ( $k = 1, 2, \dots, K$ ) during execution. All resources have a fixed per period availability  $a_k$ . Every non-dummy activity  $j$  has a weight  $w_j$  that denotes the marginal cost of deviating the realized starting time of activity  $j$  during execution from its predicted activity start time in the baseline schedule. The weight of the dummy end activity  $w_n$  denotes the cost of delaying the project completion beyond a predefined deterministic project due date  $\delta_n$ . The objective is to build a stable precedence and resource feasible baseline schedule by minimizing the stability cost function  $\sum w_j(E|s_j - s_j|)$ , defined as the weighted sum of the expected absolute deviations between the predicted starting times  $s_j$  of the activities in the baseline and their realized starting times  $s_j$  during actual schedule execution. Following the classification scheme of Herroelen et al. [7] this problem can be classified as  $m, 1|cpm, d_j, \delta_n|\sum w_j(E|s_j - s_j|)$ . The first field ( $m, 1$ ) refers to the use of an arbitrary number of renewable resource types. The second field specifies finish-start precedence relations (*cpm*), stochastic activity durations ( $d_j$ ) for which the distribution function is assumed known or can be estimated, and a deterministic project due date ( $\delta_n$ ). The last field specifies the non-stability function to be minimized. The scheduling problem for stability has been shown to be ordinary NP-hard [10].

Both simple heuristics and meta-heuristics will be presented in this paper, the objective of which is to generate stable baseline schedules with acceptable makespan performance. The remainder of the paper is organized as follows. Section II introduces the different heuristic procedures. The set-up of the computational experiment is described in Section III. Section IV presents the computational results, while a last section is devoted to overall conclusions.

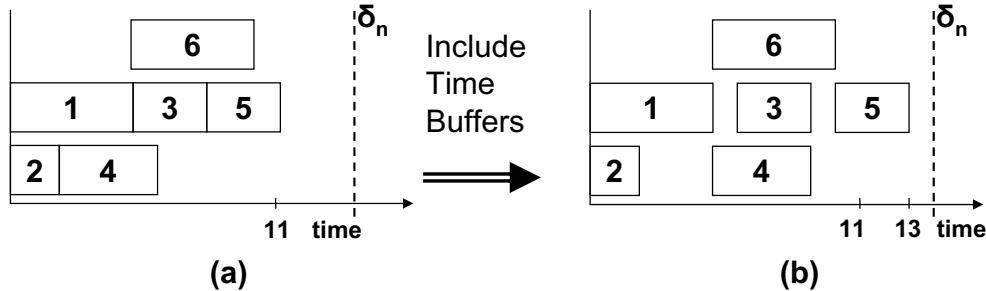


Fig. 1. Inserting time buffers in a baseline schedule

## II. ALGORITHMS

The heuristic algorithms described in this section all consider a deterministic project due date  $\delta_n$  and start from an initial unbuffered schedule in which time buffers are inserted in order to protect against anticipated disruptions. In this section several algorithms for buffer allocations are introduced. Every feasible solution for the deterministic resource-constrained project scheduling problem (RCPSP) using mean activity durations (problem  $m, 1|cpm|C_{\max}$  [7]) can serve as initial unbuffered schedule. The impact of the scheduling procedure used to construct this unbuffered schedule lies outside the scope of this paper. All buffer allocation procedures will be illustrated on a minimum makespan schedule constructed by applying the branch-and-bound algorithm of Demeulemeester & Herroelen [2], [3] for solving the deterministic RCPSP. We set  $\delta_n = \lfloor 1.3 \times C_{\max} \rfloor$ . A recent study [12] investigated the impact of several project due date settings on stability and makespan and a project due date of  $\lfloor 1.3 \times C_{\max} \rfloor$  was found to be adequate for most project schedules. The included time buffers are idle periods (gaps) in the schedule between the planned starting time of an activity and the latest planned finish time of its predecessors. The buffers should act as cushions to prevent propagation of a disruption throughout the schedule.

Figure 1 illustrates the insertion of time buffers in a baseline schedule. Figure 1(a) shows a minimum makespan schedule. Figure 1(b) shows the buffered schedule with time buffers inserted in front of activities 3, 4 and 5.

Two factors are taken into consideration in determining the size of the buffer in front of an activity  $i$ . First, the variability of all the activities that precede activity  $i$  in the schedule (measured by the standard deviation of their duration) is taken into account, because it affects the probability that activity  $i$  can start at its scheduled starting time. Second, the weight of activity  $i$ , and both the weights of its predecessors and successors contain

relevant information, because they reflect how costly it is to disrupt the starting time of activity  $i$  in relation to its predecessors and successors in the schedule.

The *resource flow dependent float factor* (RFDFF) heuristic, proposed in [12], is used as our evaluation benchmark. This heuristic relies completely on the activity weights but does not exploit the available information offered by the activity duration distributions in making its buffering decisions. The operating principles of RFDFF will be recapitulated in Section II.A. The *virtual activity duration extension* (VADE) heuristic presented in Section II.B, relies on the standard deviation of the duration of an activity in order to compute a modified duration to be used in constructing the baseline schedule. The *starting time criticality* (STC) heuristic (described in Section II.C) tries to combine information on activity weights and activity duration variances. Section II.D introduces an improvement phase that can be added to each of the just mentioned heuristics to enhance their performance. Section II.E describes a tabu search meta-heuristic that searches for the best buffer insertion for a given schedule by exploring the neighborhood solutions.

### A. *RFDFF*

The suboptimal resource flow dependent float factor (RFDFF) heuristic that has been developed by Van de Vonder et al. [12] as an extension to the *adapted float factor* (ADFF) heuristic proposed in [9] and [13], will serve as our benchmark. RFDFF starts from an unbuffered schedule (in this paper this is the minimum duration schedule obtained by the branch-and-bound procedure of Demeulemeester & Herroelen [2], [3]) and modifies it by adding safety buffers in front of activities. The hope is that the time buffers serve as a cushion to prohibit the propagation of the disruptions through the schedule.

The starting time of activity  $j$  in the RFDDFF schedule is calculated as  $s_j(S) := s_j(B\&B) + \alpha_j(\text{float}[j])$ , where  $s_j(B\&B)$  denotes the starting time of activity  $j$

in the minimum duration baseline schedule and  $\alpha_j$  denotes the *activity dependent float factor*. The total float,  $\text{float}[j]$ , is the difference between the latest allowable starting time of activity  $j$  given the project due date (i.e. its starting time in the right-justified version of the minimum duration schedule) and its scheduled starting time in the minimum duration schedule.

To calculate the float factors  $\alpha_j$ , we first need to define a resource flow network  $G' = (N, R)$  [1] on the minimum duration schedule. The resource flow network is a network with the same set of nodes ( $N$ ) as the original project network  $G = (N, A)$ , but with arcs connecting two nodes if there is a resource flow between the corresponding activities. It thus identifies how each single item of a resource is passed on between the activities in the schedule. A schedule may allow for different ways of allocating the resources so that the same schedule may be represented by different resource flow networks. We use the single pass algorithm of Artigues & Roubellat [1] to select a feasible resource flow network.

The float factors  $\alpha_j$  are now calculated as  $\alpha_j = \beta_j / (\beta_j + \lambda_j)$ , where  $\beta_j$  is the sum of the weight of activity  $j$  and the weights of all its transitive predecessors in both  $G$  and  $G'$ , while  $\lambda_j$  is the sum of the weights of all transitive successors of activity  $j$  in both networks. The weights of activities that start at time 0 are not included in these summations because it is assumed that these activities can always start at their planned start time and thus do not need any buffering to cope with possible disruptions of their predecessors. The RFDF heuristic consequently inserts longer time buffers in front of activities that would incur a high cost if started earlier or later than originally planned and resource constraints will always remain satisfied in the resulting schedule.

## B. VADE

The *virtual activity duration extension* (VADE) heuristic starts from a completely different point of view. The standard deviations  $\sigma_j$  of the activity durations, assumed known, are used to iteratively compute virtual duration extensions for the non-dummy activities. These virtual activity durations are used to update the activity start times and, by doing so, insert time gaps in the baseline schedule. The updated activity starting times are then used to generate the buffered baseline schedule using the original expected activity durations.

The iterative procedure works as follows:

For  $j = 1, 2, \dots, n - 1$  do  $d_j^* = E(\mathbf{d}_j)$  and  $v_j = 1$ ;  
 Compute  $s_j$ ,  $j = 1, 2, \dots, n$ ;  
 While  $s_n \leq \delta_n$  do

```

Find  $j^* : \frac{v_j^*}{\sigma_{j^*}} = \min_j \left\{ \frac{v_j}{\sigma_j} \right\}$ 
(tie-break:  $\max sw_i = \sum_{i=\text{succ}(j)} w_i$ );
 $v_j^* = v_j^* + 1$ ;
 $d_j^* = d_j^* + 1$ ;
Compute  $s_n$ ;
```

Generate the buffered baseline schedule.

Initially each non-dummy activity duration  $d_j^*$ ,  $j = 1, 2, \dots, n - 1$  is set equal to its expected value  $E(\mathbf{d}_j)$  and all  $v_j = 1$ . The initial activity start times  $s_j$ ,  $j = 1, 2, \dots, n$ , are computed by creating an early start schedule for the resource flow network using the activity durations  $d_j^*$ . As long as the project duration stays within the due date, the activity start times are iteratively updated as follows. Determine the activity  $j^*$  for which  $\frac{v_j^*}{\sigma_{j^*}} = \min_j \left\{ \frac{v_j}{\sigma_j} \right\}$ . Ties are broken by selecting the activity for which the sum of the weights of all its non-dummy successors is the smallest. Set  $v_j^* = v_j^* + 1$  and  $d_j^* = d_j^* + 1$ . If necessary, update the schedule and reiterate.

## C. STC

The *starting time criticality* heuristics exploit information about both the weights of the activities and the variance structure of the activity durations. The basic idea is to start from an unbuffered initial schedule and iteratively create intermediate schedules by adding a one-unit time buffer in front of that activity that needs it the most in the current intermediate schedule, until adding more safety would no longer improve stability. We thus need a measure to quantify for each activity how critical its current starting time is in the current intermediate baseline schedule.

The starting time criticality of an activity  $j$  is defined as  $stc(j) = P(s(j) > s(j)) \times w_j = \gamma_j \times w_j$ , where  $\gamma_j$  denotes the probability that activity  $j$  cannot be started at its scheduled starting time.

The iterative procedure runs as follows. At each iteration step (see pseudocode below) the buffer sizes of the current intermediate schedule are updated. The activities are listed in decreasing order of the  $stc(j)$ . The list is scanned and the size of the buffer to be placed in front of the currently selected activity from the list is augmented by one time period and the starting times of the direct and transitive successors of the activity are updated. If this new schedule has a feasible project completion ( $s_n < \delta_n$ ) and results in a lower estimated stability cost ( $\sum_{j \in N} stc(j)$ ), the schedule serves as the input schedule for the next iteration step. If not, the next activity in the list is considered. Whenever we reach an activity  $j$  for which  $stc(j) = 0$  (all activities  $j$  with  $s_j = 0$  are by definition in this case) and no feasible improvement is

found, a local optimum is obtained and the procedure terminates.

### Iteration step

```

Calculate all  $stc[i]$ 
Sort activities by decreasing  $stc[i]$ 
While no improvement found do
    take next activity  $j$  from list
    if  $stc[j]=0$ : procedure terminates
    else add buffer in front of  $j$ 
        update schedule
        if improvement & feasible do
            store schedule
            goto next iteration step
        else
            remove buffer in front of  $j$ 
            restore schedule

```

The iteration step of the STC heuristic

Regrettably, the probabilities  $\gamma_j$  are not easy to compute. We define  $k(i, j)$  as the event that predecessor  $i$  disturbs the planned starting time of activity  $j$ . The probability that this event occurs can be expressed as  $P(k(i, j)) = P(s_i + d_i + LPL(i, j) > s_j)$  in which  $LPL(i, j)$  is the sum of the durations  $d_h$  of all activities  $h$  on the longest path between activity  $i$  and activity  $j$  in original network  $G$  or the resource flow network  $G'$ .  $\gamma_j$  can then be calculated as  $\gamma_j = P(\bigcup_{(i,j) \in T(A \cup R)} k(i, j))$ , with  $T(A \cup R)$  being defined as the set of all direct and transitive<sup>1</sup> predecessors of  $j$  in the original network and the resource flow network. STC makes two assumptions in approximating  $\gamma_i$ : (a) predecessor  $i$  of activity  $j$  starts at its originally planned starting time when calculating  $k(i, j)$  and (b) only one activity at a time disturbs the starting time of activity  $j$ . Assumption (b) means that we estimate  $P(\bigcup_{(i,j) \in T(A \cup R)} k(i, j))$  by  $\sum_{(i,j) \in T(A \cup R)} P(k(i, j))$ , i.e. we assume that  $P(k(i1, j) \cap k(i2, j)) = 0$  for each  $(i1, j), (i2, j) \in T(A \cup R)$ . Assumption (a) boils down to setting  $s_i = s_i$ . Combining both assumptions yields  $\gamma'_j = \sum_{(i,j) \in T(A \cup R)} P(d_i > s_j - s_i - LPL(i, j))$  such that  $stc(j) = \gamma'_j \times w_j$ . Because  $s_i$ ,  $s_j$  and  $LPL(i, j)$  and the distribution of  $d_i$  are all known, we can now easily calculate all values of  $\gamma'_j$  and  $stc(j)$  for every activity  $j$ .

### D. Improvement heuristic

The improvement heuristic starts from an initial solution. This can be the initial unbuffered schedule or a schedule found by any of the heuristics discussed above.

<sup>1</sup> $TG = (N, TA)$  represents the transitive closure of  $G = (N, A)$  with edge  $(i, j) \in TA$  if there is a directed path from  $i$  to  $j$  in  $A$

The activities are entered in a list in decreasing order of their starting time in the input schedule. The activities are considered in the order dictated by the list. For the currently selected activity from the list, it is determined how many periods the activity can be moved backward and forward in time without affecting the starting time of any other activity in the schedule. From all discrete time instants in this displacement interval, the instant that yields the lowest stability cost during simulation, is chosen as the new starting time of the current activity in the updated schedule. With this updated schedule, we proceed to the next activity in the list. If the next activity in the list is the dummy start activity, we restart the list. If the list is scanned entirely without any improvement, a local optimum has been found and the procedure terminates.

Basically the algorithm is a combination of steepest and fastest descent. For an activity selected from the list, we examine all possible starting times and select the best one (steepest descent). However, we do not examine the entire range of starting times of *all* the activities and select the best, but instead we already update the schedule if a better solution is found for the current activity before proceeding to the next activity in the list. This fastest descent part of the algorithm is included to speed up computations.

### E. Tabu search

Descent approaches may terminate at a local optimum after some iterations when no further improvement can be found in the direct neighborhood of the current solution. Glover [5], [6] developed the principle of tabu search algorithms, which allow to select the mildest ascent solution to continue the search whenever no improvement can be found. A tabu list keeps track of recent solutions that will be forbidden moves in order to avoid cycling.

The tabu search procedure departs from the STC-schedule described in Section II.C. At each iteration step, the neighborhood of the current solution contains at most  $2 \times (n-2)$  solutions. For each non-dummy activity of the project, we have two possible neighborhood solutions. One is obtained by increasing the buffer in front of the activity in the schedule by one time period, if possible (*plus-move*). The other is obtained by decreasing the buffer size of this activity by one unit, if possible (*minus-move*). The buffers in front of all other activities are left unchanged. Two tabu lists are kept, both of length  $n/3$ . The first list stores all recent plus-moves, while the second one stores all recent minus-moves. Before allowing a new plus-move, we have to check whether this activity

is not in the second list. If a buffer size decrease (minus-move) delivers the best solution in the neighborhood, the first tabu list has to be checked. By doing so, we avoid cycling, but we do allow an activity to be consecutively selected more than once if the considered moves have the same direction. The aspiration criterion defines that a move that would yield a new best solution will be accepted even if it would normally be prohibited by the tabu list. Because the tabu search described here only adds or subtracts one unit of time buffer at a time, large shifts of the starting time of an activity compared to its initial starting time will only occur if all intermediate positions yield acceptable solutions. This might obstruct the procedure to move an activity into its actual best positioning for all other activity starting times considered fixed. To remove this inconvenience, one iteration step of the improvement heuristic of Section II.D will be allowed after a fixed number of iterations (100). The overall best found solution is stored throughout the whole procedure. The tabu search stops after a fixed number of iterations.

### III. EXPERIMENTAL SET-UP

All proposed algorithms have been coded in Microsoft Visual C++ 6.0. The procedures were tested on two data sets. A first set consist of the 480 networks of the J30 instance set of PSPLIB [8]. A second set consists of the eighty 30-activity networks constructed in [11] by using the RanGen project scheduling network instances generator developed by Demeulemeester et al. [4] using two settings (0.5 and 0.75) for the order strength, resource factor and resource constrainedness. For an extensive study of the impact of the parameter settings on schedule stability we refer to [12].

In order to investigate the impact of activity duration variability, we distinguish between low, high and random duration variability. *High duration variability* means that the real activity durations are all discretized values drawn from a right-skewed beta-distribution with parameters 2 and 5, that is transformed in such a way that the minimum duration equals 0.25 times the expected duration, the mean duration equals the expected duration and the maximum duration equals 2.875 times the expected duration. *Low duration variability* means that the realized activity durations are also discretizations of values drawn from a beta-distribution with parameters 2 and 5, but with the mean equal to the expected activity duration and with minimum and maximum values equal to 0.75 times and 1.625 times the expected activity duration, respectively. Last, *random duration variability* stands for the case where no overall uncertainty level exists for the project and the variabilities are activity dependent. We

randomly select for every activity whether the activity has *high*, *low* or *medium duration variability*. The distribution functions for high and low variability are as explained above, while *medium duration variability* is an intermediate case where the realized activity durations are drawn from a beta-distribution with parameters 2 and 5, but with minimum and maximum values equal to 0.5 times and 2.25 times the expected activity duration, respectively. Figure 2 shows the distribution functions from which the realized durations are drawn for an activity with an expected 3-period duration.

The stability cost  $\sum w_j (E|s_j - s_{j*}|)$  is evaluated by drawing the  $w_j$  for each non-dummy activity  $j \in \{1, 2, \dots, n-1\}$  from a discrete triangular distribution with  $P(w_j = q) = (21 - 2q)\%$  for  $q \in \{1, 2, \dots, 10\}$ . This distribution results in a higher probability for low weights and in an average weight  $w_{avg} = 3.85$ . The weight  $w_n$  of the dummy end activity denotes the marginal cost of not making the projected project completion and will be fixed at  $\lfloor 10 \times w_{avg} \rfloor = 38$ . For an extensive evaluation of the impact of the activity weight of the dummy end activity, we refer to [11] and [12].

Extensive simulation will be used to evaluate all procedures on stability and computational efficiency. For every network instance, two sets of 100 executions (referred to further on as the *training set* and the *test set*) are simulated by drawing different actual activity durations from the described distribution functions. The test set of executions is run to avoid overfitting as will be explained below.

Using the simulated activity durations, the *realized schedule* is constructed by applying the following reactive procedure. First, an activity list is deduced from the baseline schedule by ordering the activities in increasing order of their starting time. Ties are broken by decreasing order of activity weight, then by increasing activity number. Afterwards, the realized schedule is constructed by applying a parallel schedule generation scheme (SGS) on this activity list using the actual activity durations. To maintain stability, an activity is never allowed to start earlier than its scheduled starting time. Thus, at each decision point we scan the ordered activity list to select the set of eligible unscheduled activities that have a baseline starting time that is not larger than the current decision time. It should be observed that the resource allocation (i.e. the flow network) of the resulting realized schedule and the baseline schedule may differ. For a comparison of different reactive procedures we refer the reader to [11].

The algorithms proposed in Sections II.D and II.E select the best neighborhood solution drawing disruptions from the stochastic activity duration density functions.

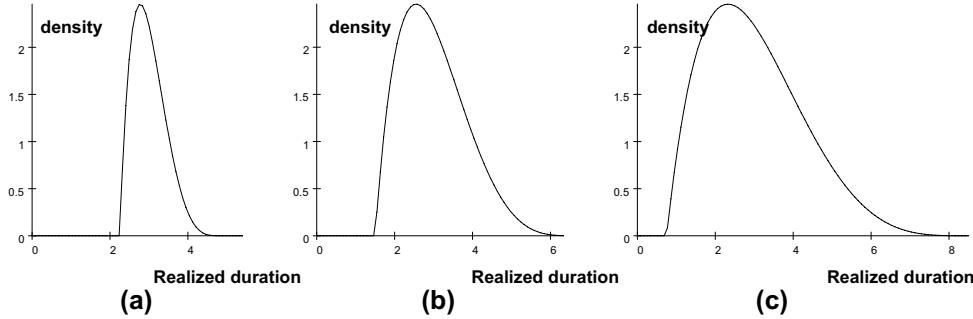


Fig. 2. Distribution functions for low (a), medium (b) and high (c) duration variability if  $E(d_i) = 3$

Including information about these simulated disruptions might make the buffer allocation decision process subject to overfitting. The best schedule for the simulated disruptions will not necessarily be the best schedule for the actual disruptions during project execution, even if we assume that they are drawn from the same density functions. As mentioned above, we try to avoid overfitting by examining the results on both the training and test set of execution scenarios.

#### IV. COMPUTATIONAL RESULTS

All computational results have been obtained on a Pentium IV 2.4 GHz personal computer. A total of ten scheduling procedures are evaluated. The minimum duration schedule obtained by the branch-and-bound procedure of Demeulemeester & Herroelen [2], [3] serves as the benchmark. Algorithms 2-4 are the RFDFF, VADE and STC heuristics introduced in Sections II.A-II.C. Algorithms 5-8 are all instances of the improvement heuristic of Section II.D with the solutions of algorithms 1-4 as initial solution. The tabu search procedure of Section II.E is executed with 100 (Algorithm 9) and 300 (Algorithm 10) iteration steps, respectively. Algorithms 1-4 will be referred to as *simple heuristics*, while algorithms 5-10 will be called the *improvement heuristics*.

For every algorithm we calculate the average stability cost (Stab) over all networks and executions on both a training set and a test set of simulated disruptions. %Best denotes the percentage of network instances for which a certain algorithm yields the minimum stability cost among the algorithms within its class. Again, this measure will be calculated for training and test set disruptions to examine the degree of overfitting. We feel that comparing simple heuristics with improvement algorithms would give few additional insights. Finally, also the average computational times (in seconds) are given for each algorithm. Note that the computational time of algorithms 2-10 include the computational time

of algorithm 1, because the RCPSP always needs to be solved. Also, the reported computational times of the improvement heuristics 5-10 will comprise the time to calculate their initial solution.

In the remainder of this section, results will be discussed for the three activity duration variability cases introduced in the previous section.

##### A. High duration variability

TABLE I  
RESULTS ON PSPLIB DATA SET WITH HIGH DURATION VARIABILITY

Algorithm	Training Set		Test Set		Time
	Stab	%Best	Stab	%Best	
1. RCPSP	400.97	0	401.27	0	0.07
2. RFDFF	137.14	6	137.69	6	0.08
3. VADE	144.96	2	146.94	3	0.17
4. STC	117.67	92	118.45	91	0.10
5. RCPSP D	108.28	3	116.80	9	1.84
6. RFDFF D	99.89	22	107.86	25	1.15
7. VADE D	100.62	11	108.98	13	1.29
8. STC D	100.23	10	107.88	22	1.08
9. Tabu 100	99.22	40	107.86	25	7.50
10. Tabu 300	98.86	63	107.67	25	22.16
11. Best	97.80	100	104.87	100	

Tables I and II show the average results on the PSPLIB and RanGen data set when all activities are subject to high activity duration variability.

1) *Simple heuristics*: Tables I and II reveal that STC gives by far the best results among the simple heuristics. Although VADE and RFDFF have a much smaller stability cost than the minimum duration schedule, they only outperform STC on a few network instances.

On average, VADE shows worse results than RFDFF on the PSPLIB set, but surprisingly generates better results on the RanGen problem set. This is due to the differences in the experimental designs used to construct both data sets. For all the networks in the RanGen

TABLE II  
RESULTS ON RANGEN DATA SET WITH HIGH DURATION  
VARIABILITY

Algorithm	Training Set		Test Set		Time
	Stab	%Best	Stab	%Best	
1. RCPSP	558.82	0	556.00	0	0.01
2. RFDFD	155.93	1	154.69	0	0.01
3. VADE	143.54	4	144.93	3	0.10
4. STC	123.81	95	123.49	98	0.03
5. RCPSP D	128.29	1	135.25	4	2.48
6. RFDFD D	114.30	14	121.49	19	1.12
7. VADE D	112.89	8	120.63	24	0.98
8. STC D	112.50	6	119.45	26	0.83
9. Tabu 100	111.01	45	119.30	18	6.89
10. Tabu 300	110.68	71	119.05	19	20.59
11. Best	109.66	100	116.43	100	

data set a rather high resource factor and resource constrainedness (0.5 or 0.75) are assumed, resulting in resource intensive networks with many forbidden sets. On the other hand, the factorial design for PSPLIB also includes networks that are less resource constrained.

When we only select the 60 resource intensive networks with resource factor exceeding 0.5 and resource strength equal to 0.2 from the PSPLIB data set, we observe that the average stability cost of the VADE heuristic equals 174.38, which is significantly better than the stability costs of RFDFD (189.96). We might thus conclude that RFDFD encounters problems when dealing with resource intensive networks. In these networks many resource conflicts need to be resolved, which will lead to more extra precedence relations in the resource flow network and eventually to a larger  $C_{\max}$  for the RCPSP and thus a larger  $\delta_n$ . RFDFD typically allocates a larger portion of the total safety to the dummy end activity than the other heuristics. For resource intensive networks, the total safety included (recall that  $\delta_n = \lfloor 1.3 \times C_{\max} \rfloor$ ) might be too high such that RFDFD overprotects the project completion and has to pay a high extra stability cost for this unnecessary extra protection due to poorly buffered intermediate activities. This explains the comparative stability advantage of VADE on resource intensive data sets, such as the 80 networks of the RanGen data set and the 60 selected networks of PSPLIB.

The required computational effort is highly correlated with the number of time-consuming<sup>2</sup> evaluations needed in the procedure, which is low for all simple heuristics. Because VADE selects the best among a range of solutions, its computational time is the most demanding

<sup>2</sup>Remember that each evaluation consists of 100 simulated executions of the project that have to be scheduled by using the parallel SGS.

among the simple heuristics. For the same reason, VADE might be slightly affected by overfitting. Indeed, if there is an increase in average stability cost in the test set over the training set results, it is more pronounced for VADE than for the other simple heuristics. However, this overfitting is clearly not critical.

2) *Improvement heuristics:* Comparing algorithms 5-10 on the training set reveals that all stable project schedules (2-4) provide good starting solutions for the improvement heuristic of section II.D. Only the minimum duration schedule leads to a substantial stability cost and computational effort. As somewhat expected, tabu search obtains the best results of all heuristics.

Row 11 represents the average over all networks of the best solutions found by any of the 10 algorithms. In 63% of the PSPLIB networks and 71 % of the RanGen networks, this best solution has been obtained by Tabu 300. The deviation between the average stability cost of Tabu 300 and the stability cost of row 11 is around 1% for both data sets. The  $2 \times (n-2)$  evaluations required at each of the 300 (100) iteration steps of the tabu search procedure account for the increase in computation time.

When comparing the performance of the improvement algorithms on the test set, we observe an increase in the average stability costs, while the mutual differences in stability cost between the algorithms are largely reduced. Although Tabu 300 remains slightly better than the improvements heuristics 6-8 on Stab, the small differences make it harder to justify the high computational burden of Tabu 300. On the RanGen data set, %Best is even better for VADE D and STC D than for our tabu search. We also note that the performance deviation of Tabu 300 from the best result found over all 10 algorithms has increased from 1% to approximately 2.5%. We may conclude that the tabu search is subject to overfitting.

### B. Low duration variability

TABLE III  
RESULTS ON PSPLIB DATA SET WITH LOW DURATION  
VARIABILITY

Algorithm	Training Set		Test Set		Time
	Stab	%Best	Stab	%Best	
1. RCPSP	123.17	0	122.64	0	0.08
2. RFDFD	6.11	0	6.01	0	0.08
3. VADE	1.69	17	1.84	11	0.14
4. STC	1.08	84	1.05	89	0.10
5. RCPSP D	6.14	0	7.03	0	1.35
6. RFDFD D	1.25	8	1.71	8	0.66
7. VADE D	0.75	10	1.29	8	0.54
8. STC D	0.67	28	1.04	35	0.46
9. Tabu 100	0.58	72	1.18	20	4.43
10. Tabu 300	0.56	85	1.18	23	13.02
11. Best	0.54	100	0.84	100	

TABLE IV  
RESULTS ON RANGEN DATA SET WITH LOW DURATION VARIABILITY

Algorithm	Training Set		Test Set		Time
	Stab	%Best	Stab	%Best	
1. RCPSP	172.91	0	171.45	0	0.01
2. RFDFD	7.09	0	6.88	0	0.01
3. VADE	1.55	19	1.64	10	0.07
4. STC	1.09	81	1.04	90	0.03
5. RCPSP D	22.00	0	22.73	0	1.46
6. RFDFD D	2.33	0	2.68	0	0.52
7. VADE D	0.82	16	1.28	16	0.39
8. STC D	0.80	24	1.09	35	0.32
9. Tabu 100	0.70	61	1.24	18	4.25
10. Tabu 300	0.68	83	1.24	21	12.71
11. Best	0.67	100	0.91	100	

Recent research [11] showed that when activity duration variability is rather low, proactive scheduling becomes increasingly attractive because the disadvantage in makespan performance compared to a minimum duration schedule becomes very small, while the improvement on stability remains large. By consequence, the low variability case of this section might be considered particularly interesting. Remark that increasing  $\delta_n$  has been shown [11] to have a similar impact on solution robustness. The amount of uncertainty in a project should always be regarded in accordance with the tightness of the project due date.

1) *Simple heuristics*: Tables III and IV give an overview of the obtained results. Compared to the results of Section IV.A.1 we observe that that RFDFD performs relatively worse on the training set and is even dominated (%Best = 0) on all networks by the other simple heuristics. RFDFD does not use any information about activity duration variability and will thus construct exactly the same schedule whatever the amount of uncertainty in the environment. While this procedure provides reasonable results in the case of high variability, its stability cost in the case of low variability is not competitive with the stability cost values obtained by other simple heuristics. RFDFD typically overprotects the project completion, causing poorly buffered intermediate activities and unnecessary stability losses during execution.

STC ranks once again best among the simple heuristics, closely followed by VADE. Taking into account that  $w_{avg} = 3.85$ , their low average stability costs ( $< 2$ ) indicate that almost every activity of the project will be executed as planned by applying these buffer allocation procedures without any loss in makespan performance compared to the RCPSP solution.

The results obtained on the test set are very similar. We only note a substantial smaller %Best performance for VADE. Because VADE evaluates multiple solutions and selects the best one, it does not come as a surprise that this heuristic is somewhat subject to overfitting, while all other simple heuristics do not use any evaluation for buffer allocation and are by definition unaffected by overfitting.

2) *Improvement algorithms*: Improvement algorithms 6-10 all generate extremely satisfying results. Tabu 300 again

ranks best among all heuristics on the training set. On the test set, we remark that STC D outperforms Tabu 300 on as well Stab as %Best. Even the simple heuristic STC without improvement phase generates a lower average stability cost than Tabu 300 and for the RanGen data set even the most stable schedule overall. The improvement phase in STC D has almost no impact. The more intensive local search done by the tabu search algorithm only overfits the baseline schedule on the simulated disruptions in the training set. Tabu search is certainly not recommended in the low duration variability case.

### C. Random duration variability

TABLE V  
RESULTS ON PSPLIB DATA SET WITH RANDOM DURATION VARIABILITY

Algorithm	Training Set		Test Set		Time
	Stab	%Best	Stab	%Best	
1. RCPSP	267.19	0	267.87	0	0.07
2. RFDFD	58.00	1	57.76	2	0.08
3. VADE	54.92	4	55.37	4	0.15
4. STC	40.79	94	40.41	94	0.10
5. RCPSP D	36.47	3	40.01	6	1.60
6. RFDFD D	32.11	19	35.25	21	0.96
7. VADE D	32.00	13	35.36	19	1.01
8. STC D	31.76	13	34.60	25	0.83
9. Tabu 100	31.13	45	34.72	21	6.24
10. Tabu 300	30.97	64	34.64	28	18.37
11. Best	30.36	100	33.05		

TABLE VI  
RESULTS ON RANGEN DATA SET WITH RANDOM DURATION VARIABILITY

Algorithm	Training Set		Test Set		Time
	Stab	%Best	Stab	%Best	
1. RCPSP	379.61	0	374.97	0	0.01
2. RFDFD	65.46	0	64.74	0	0.01
3. VADE	48.82	6	48.22	9	0.08
4. STC	40.26	94	39.01	91	0.03
5. RCPSP D	48.83	0	51.04	4	2.37
6. RFDFD D	36.78	8	39.08	10	0.91
7. VADE D	34.18	24	36.87	21	0.74
8. STC D	33.78	11	35.97	31	0.66
9. Tabu 100	33.23	40	35.91	28	0.57
10. Tabu 300	32.95	68	35.88	25	16.98
11. Best	32.50	100	34.61	100	

In this case, we assume that the project manager is able to estimate the amount of uncertainty present in each individual activity. By taking this information into account when allocating buffers VADE and STC construct baseline schedules that should be better adapted to the project under consideration.

Overall, the obtained results (Tables V and VI) are very similar to the ones obtained for the high duration variability case of section IV.A. First, STC is again by far the best

performing simple heuristic on the PSPLIB and RanGen data sets. Second, the tabu search procedure obtains the best overall results, although the differences between training set and test set indicate that they are partially affected by overfitting.

The RFDFF schedule that does not use the available extra information, performs on average slightly worse than VADE on PSPLIB, while its disadvantage becomes even more explicit on the resource intensive RanGen networks.

We might conclude that STC (D) manages best to deal with random duration variability. Apart from the tabu search procedure, we see few reasons to use other heuristics than STC in the random variability case.

## V. CONCLUSIONS AND FURTHER RESEARCH

Proactive project scheduling is concerned with building stable baseline schedules that are able to absorb most of the anticipated disruptions during project execution. In this paper, we presented various heuristic algorithms for inserting time buffers in a project schedule.

An extensive simulation-based experiment on PSPLIB and RanGen instances revealed that the STC heuristic in general ranks best among the simple heuristics that do not rely on an improvement phase. STC uses information on activity weights and activity duration variability for the buffer allocation process.

Improvement heuristics will typically yield better solutions, but may be subject to overfitting. To avoid this, results were examined on both a training and a test set, where the test set was constructed by drawing different actual activity durations from the distribution function. When the project environment comprises low activity duration uncertainty, any improvement on the STC baseline schedule will very likely turn out to be overfitting during project execution. When some or all project activities are subject to considerable uncertainty, local search will improve the stability of the baseline schedule even after the impact of overfitting is removed. The proposed tabu search procedure results in the minimum expected cost of the project, but is rather time consuming. A descent approach with STC (or VADE) as initial scheduling procedure results in an almost equally small expected stability cost and requires much less computational effort to calculate the baseline schedule.

In this paper buffers were heuristically allocated to a given minimum duration schedule while the resource allocation was kept fixed. The development of efficient exact buffer allocation procedures is a topic for further research. The study of the impact of different initial schedules and different resource allocations on the efficiency and effectiveness of the buffer allocation process is also an interesting open research issue. Starting from a heuristic RCPSP solution would make it possible to extend the results to the J60, J90 and J120 PSPLIB data sets. Ultimately, we aim at finding an integrated solution robust approach for scheduling, resource allocation and buffering.

## ACKNOWLEDGEMENTS

This research has been supported by Project OT/03/14 of the Research Fund K.U.Leuven.

## REFERENCES

- [1] C. Artigues and F. Roubellat, "A polynomial activity insertion algorithm in a multi-resource schedule with cumulative constraints and multiple modes," *European Journal of Operational Research*, vol. 127, pp. 294–316, 2000.
- [2] E. Demeulemeester and W. Herroelen, "A branch-and-bound procedure for the multiple resource-constrained project scheduling problem," *Management Science*, vol. 38, pp. 1803–1818, 1992.
- [3] ———, "New benchmark results for the resource-constrained project scheduling problem," *Management Science*, vol. 43, pp. 1485–1492, 1997.
- [4] E. Demeulemeester, M. Vanhoucke, and W. Herroelen, "RanGen: A random network generator for activity-on-the-node networks," *Journal of Scheduling*, vol. 6, pp. 17–38, 2003.
- [5] F. Glover, "Tabu search, Part I," *INFORMS, Journal of Computing*, vol. 1, pp. 190–206, 1989.
- [6] ———, "Tabu search, Part II," *INFORMS, Journal of Computing*, vol. 2, pp. 4–32, 1990.
- [7] W. Herroelen, B. De Reyck, and E. Demeulemeester, "On the paper "Resource-constrained project scheduling: notation, classification, models and methods" by Brucker et al." *European Journal of Operational Research*, vol. 128, no. 3, pp. 221–230, 2000.
- [8] R. Kolisch and A. Sprecher, "PSPLIB - a project scheduling library," *European Journal of Operational Research*, vol. 96, pp. 205–216, 1997.
- [9] R. Leus, "The generation of stable project plans," Ph.D. dissertation, Department of applied economics, Katholieke Universiteit Leuven, Belgium, 2003.
- [10] R. Leus and W. Herroelen, "The complexity of machine scheduling for stability with a single disrupted job," *Operations Research Letters*, vol. 33, pp. 151–156, 2005.
- [11] S. Van de Vonder, E. Demeulemeester, and W. Herroelen, "An investigation of efficient and effective predictive-reactive project scheduling procedures," Department of applied economics, Katholieke Universiteit Leuven, Belgium, Tech. Rep. 0466, 2005.
- [12] S. Van de Vonder, E. Demeulemeester, W. Herroelen, and R. Leus, "The trade-off between stability and makespan in resource-constrained project scheduling," 2004, *International Journal of Production Research*, to appear.
- [13] ———, "The use of buffers in project management: the trade-off between stability and makespan," *International Journal of Production Economics*, vol. 97, pp. 227–240, 2005.



# Exact Solution Procedures for the Balanced Unidirectional Cyclic Layout Problem

Temel Öncan \* and İ.Kuban Altinel †

\* Galatasaray University/Department of Industrial Engineering  
Ortaköy, İstanbul, 34357, TÜRKİYE

Email: ytoncan@gsu.edu.tr

†Boğaziçi University/Department of Industrial Engineering  
Bebek, İstanbul, 34342, TÜRKİYE

Email: altinel@boun.edu.tr

**Abstract**—In this paper we consider the balanced unidirectional cyclic layout problem (BUCLP) arising in the determination of workstation locations around a closed loop conveyor system, in the allocation of cutting tools on the sites around a turret, in the positioning of stations around a unidirectional single loop AGV path. BUCLP is known to be NP-hard. One important property of this problem is the balanced material flow assumption where the material flow is conserved at every workstation. We first develop a branch-and-bound procedure by using the special material flow property of the problem. Then, we propose a dynamic programming algorithm, which provides optimum solutions for instances with up to 20 workstations due to memory limitations. The branch and bound procedure can solve problems with up to 50 workstations.

**Keywords**—Balanced unidirectional cyclic layout, flexible manufacturing systems, branch and bound, dynamic programming.

## I. INTRODUCTION

CONSIDER a manufacturing cell which consists of a circular material handling system and  $n$  workstations assigned to  $n$  predetermined candidate locations. Circular material handling systems connect all workstations by a circuit passing through each workstation exactly once. The system is assumed to move the materials unidirectionally (e.g. clockwise or counter-clockwise) around the circuit. Typical examples of this type material handling systems are loop conveyors (Figure 1), robot arms rotating unidirectionally (Figure 2), and unidirectional single loop automated guided vehicles (Figure 3). As it can be seen one of the workstations serves as the load/unload (LUL) area. A common operational policy for circular material handling systems is to require all parts enter and leave the manufacturing cell at the LUL area. Each part is routed through workstations

following the sequence specified in its process plan. When a part is processed at one of the workstations, material handling system moves it unidirectionally to the next workstation pointed in the process plan. If the workstation is occupied, the part is awaited in a buffer until it becomes available. The objective is to determine the assignment of workstations to candidate locations which minimizes total transportation cost of the parts in the manufacturing cell within a unit time. This is a layout problem where a layout is an assignment of workstations to locations. According to Afentakis [2], Unidirectional Cyclic Layouts (UCLs) are mostly preferred because of their relative low initial investment costs, high material handling flexibility and their ability of being easily accommodated to future introduction of new parts and process changes.

As underlined by Kouvelis, Chiang and Kiran [13] the optimal design of its physical layout is crucial for the performance of an FMS. The work by Afentakis [2] is the very first attempt for the design of a UCL in this respect. He gives a binary integer linear programming formulation to determine locations of the workstations which minimize material transport in a UCL. We refer this problem as the Unidirectional Cyclic Layout Problem (UCLP) in the sequel. Kouvelis and Kim [15] have shown that the UCLP is NP-hard. A detailed discussion on the complexity of the UCLP is also provided by Tansel and Bilen [22].

As a generalization of the UCLP we refer to the single loop facilities layout problem (SLFL) where both the unidirectional and bidirectional flow cases are considered [4]. The SLFP is much more complicated than UCLP since it consists of two subproblems: the determination of the optimal sequence of workstations along a single loop based on the part flow matrix and the design of optimal flow path configuration subject to this inter-

workstations sequence.

UCLP has special forms. In one of them the material flow is conserved at each workstation: Total inflow is equal to total outflow at a workstation. This version of the UCLP is known as the *balanced unidirectional cyclic layout problem* (BUCLP). In this work we concentrate on the balanced case which is particularly relevant in automated manufacturing where no manual interruption is allowed to remove/insert parts from/to the workstations. Another special form of the UCLP is the *equidistant unidirectional cyclic layout problem* (EUCLP), where the locations around the unidirectional cyclic material handling system are assumed to be equally distant to each others (Figure 4). The third special type is known as the *balanced equidistant unidirectional cyclic layout problem* (BEUCLP) and combines these two. Equidistant UCLP (EUCLP) and balanced UCLP (BUCLP) have been addressed by Bozer and Rim [6] and Kiran, Ünal and Karabati [12] respectively.

Bennell, Potts and Whitehead [5] considered one extension of the UCLP, namely the min-max loop layout problem (MMLLP). The MMLLP is to find the ordering of workstations around the UCL such that the maximum number of circuits required for any product is minimized. Notice that only the objective functions of the UCLP and MMLLP are different. The authors proposed iterated descent, tabu search and simulated annealing methods for this problem.

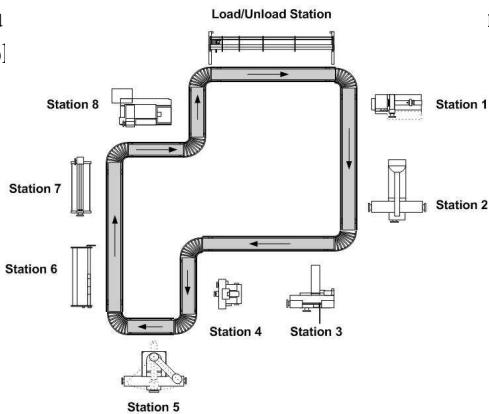


Fig. 1. Closed loop conveyor system with 9 stations

In this work, we propose two exact solution approaches for the BUCLP: branch and bound scheme and dynamic programming algorithm. New upper and lower bounding methods are proposed and tested within the branch and bound scheme. Using the special matrix properties of the problem we propose new dominance rules that are used within both the branch and bound and dynamic programming algorithms. To the best of our knowledge, there is no other work proposing exact algorithms for the BUCLP. Only, Lee, Huang and Chiang [16], Kouvelis and Kim [15] and Kiran and Karabati

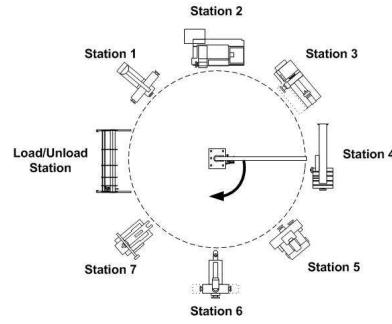


Fig. 2. Unidirectional robot arm serving 8 stations

[11] have general UCLP

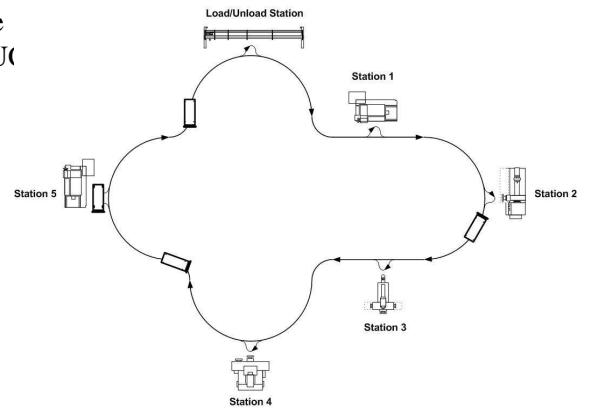


Fig. 3. AGV system with 6 stations and 4 vehicles

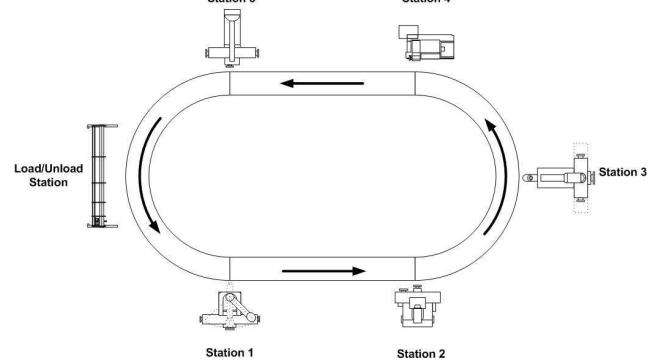


Fig. 4. Equidistant closed loop conveyor system with 6 workstations

The remainder of this paper is organized as follows. In the next section we provide a formal description of the BUCLP and explain two known formulations. The new branch and bound algorithm is presented in Section III. In Section IV we give our dynamic programming approach for the BUCLP. Section V includes our computational results. Finally, the conclusions are given in Section VI.

## II. BALANCED UNIDIRECTIONAL CYCLIC LAYOUT PROBLEM

Let  $N = \{1, \dots, n\}$  be the set of workstations to be located at candidate locations connected by a

unidirectional circular material handling system where one of the workstations represents the LUL area, and  $F$  be the  $n \times n$  part flow matrix whose  $(i, j)^{th}$  entry  $f_{ij} \geq 0$  denotes the average number of jobs to be moved from workstation  $i$  to workstation  $j$  over a given length of time. Clearly  $f_{ii} = 0$ . A discussion of how matrix  $F$  is determined from process plans can be found in [22]. Let  $C_i = \sum_{j,j \neq i} f_{ji}$  denote the sum of jobs moved other workstations to workstation  $i$  (i.e. total inflow of workstation  $i$ ) and,  $R_i = \sum_{j,j \neq i} f_{ij}$  denote the sum of jobs moving from workstation  $i$  to other workstations (i.e. total outflow of workstation  $i$ ). For the BUCLP, we assume that the material flow is conserved namely,  $R_i = C_i$ , holds for all workstations  $i = 1, \dots, n$ .

Unidirectional cyclic material handling systems together with the  $n$  candidate locations specified around it can be modelled by a circuit with  $n$  vertices. Notice that  $n$  is the number of candidate locations (also workstations) including the LUL area. The vertices are numbered 1 through  $n$  in increasing order in clockwise direction, which is assumed without loss of generality as the direction of movement is illustrated in Figure 5.

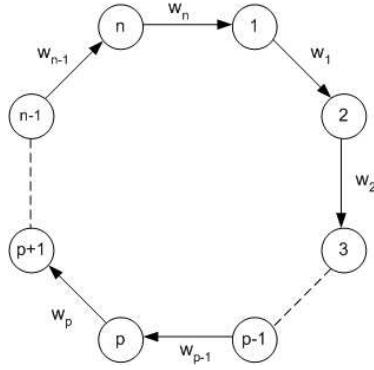


Fig. 5. A unidirectional cyclic material handling system.

The weight  $w_t$  is the length of arc  $(t, t + 1)$  and determines the distance between locations  $t$  and  $t + 1$ . Notice that location  $n + 1$  denotes location 1 because of circularity. As a consequence,  $d_{ij}$ , the transportation distance from location  $i$  to location  $j$ , becomes

$$d_{ij} = \begin{cases} \sum_{t=i}^{j-1} w_t & \text{if } i < j \\ \lambda - \sum_{t=i}^{j-1} w_t & \text{if } i > j \end{cases}. \quad (1)$$

Here the constant  $\lambda$  is the length of the circuit. Namely,  $\sum_{t=1}^n w_t = \lambda$  and distance  $d_{ij}$  satisfies the following metric properties:

- 1)  $d_{ij} = 0 \quad \forall i, j; i = j$
- 2)  $d_{ik} + d_{kj} \geq d_{ij} \quad \forall i, j, k; i \neq j \neq k$
- 3)  $d_{ij} + d_{ji} = \lambda \quad \forall i, j; i \neq j$
- 4)  $d_{ij} \neq d_{ji}$  unless  $d_{ij} = \lambda/2 \quad \forall i, j; i \neq j$

A layout of  $n$  workstations, which has already been defined as the assignment of workstations  $i = 1, \dots, n$  to candidate locations  $\{1, \dots, n\}$  with one workstation per location, can be denoted as a permutation vector  $\pi = (\pi_1, \dots, \pi_n)$  where  $\pi_i$  is the workstation number assigned to location  $i$ . Then the question is to determine an assignment which minimizes certain appropriate cost function of material transport. For the UCLP two types of objective function have been used in the literature:

- 1) Minimization of the total part transport distance per unit time
- 2) Minimization of the total number of parts that pass through the LUL area per unit time

As shown by Bozer and Rim [6], Kiran and Karabati [11], and Kiran, Ünal, and Karabati [12] under the first objective the UCLP becomes the QAP after letting  $\Pi$  denote the set of all possible layouts, namely all possible permutations of indices  $\{1, \dots, n\}$

$$\min_{\pi \in \Pi} c_1(\pi) = \min_{\pi \in \Pi} \sum_{i=1}^n \sum_{\substack{j=1 \\ j \neq i}}^n f_{\pi_i \pi_j} d_{ij}. \quad (2)$$

To formulate the UCLP with the second objective function Afentakis [2], and Kouvelis and Kim [15] have defined the indicator function

$$I(i, j) = \begin{cases} 1 & \text{if } i > j \\ 0 & \text{otherwise} \end{cases}, \quad (3)$$

which is used to count the number of parts passing through the LUL area. If the location  $i$  is greater than location  $j$ , the parts going from workstation  $\pi_i$ , located at location  $i$ , to workstation  $\pi_j$ , located at location  $j$ , passes through the LUL area; and this results in the following QAP formulation of the UCLP:

$$\min_{\pi \in \Pi} c_2(\pi) = \min_{\pi \in \Pi} \sum_{i=1}^n \sum_{\substack{j=1 \\ j \neq i}}^n f_{\pi_i \pi_j} I(i, j). \quad (4)$$

In fact,

$$c_1(\pi) = \lambda c_2(\pi) \quad (5)$$

for *UCLs with single LUL area and balanced part flow*, both formulations become equivalent. This result has been shown by Kouvelis and Kim [15]. Then, any one of the workstations can be chosen as the LUL area. As a remark, although it is not particularly stated in any of these works, balanced part flow assumption is necessary in the derivation of relation (5).

The relation (5) between  $c_1(\pi)$  and  $c_2(\pi)$  is important because  $c_2(\pi)$  has the following property which helps us to provide computationally efficient upper bounds [3] and new dominance rules.

Consider the initial layout

$$\bar{\pi} = (\pi_1, \dots, \pi_{i-1}, \pi_i, \pi_{i+1}, \dots, \pi_{j-1}, \pi_j, \pi_{j+1}, \dots, \pi_n), \quad (6)$$

and *move forward* workstation  $\pi_i$  located at location  $i$  to location  $j$  (with  $i < j$ ). This results in the new layout

$$\bar{\pi}^f = (\pi_1, \dots, \pi_{i-1}, \pi_{i+1}, \dots, \pi_{j-1}, \pi_j, \pi_i, \pi_{j+1}, \dots, \pi_n), \quad (7)$$

where workstation  $\pi_i$  is now at location  $j$ . However, for the same initial layout (6), when workstation  $\pi_j$  located at location  $j$  is *moved backward* to location  $i$  (with  $i < j$ ) the new layout

$$\bar{\pi}^b = (\pi_1, \dots, \pi_{i-1}, \pi_j, \pi_i, \pi_{i+1}, \dots, \pi_{j-1}, \pi_{j+1}, \dots, \pi_n), \quad (8)$$

is obtained. Workstation  $\pi_j$  is now at location  $i$ . Then, the change in the cost function  $c_2(\pi)$  due to the forward move of workstation  $\pi_i$  to location  $j$  and the backward move of workstation  $\pi_j$  to location  $i$  are respectively

$$\Delta_{c_2}^f = c_2(\bar{\pi}) - c_2(\bar{\pi}^f) = \sum_{k=i+1}^j (f_{\pi_k \pi_i} - f_{\pi_i \pi_k}), \quad (9)$$

and

$$\Delta_{c_2}^b = c_2(\bar{\pi}) - c_2(\bar{\pi}^b) = \sum_{k=i}^{j-1} (f_{\pi_j \pi_k} - f_{\pi_k \pi_j}). \quad (10)$$

This is a direct consequence of the definition of the cost function  $c_2(\pi)$ . We can determine the costs of layouts  $\bar{\pi}$  and  $\bar{\pi}^f$  as,

$$\begin{aligned} c_2(\bar{\pi}) = & \sum_{k=j+1}^n \sum_{l=1}^{i-1} f_{\pi_k \pi_l} + \sum_{k=j+1}^n f_{\pi_k \pi_i} + \\ & \sum_{k=j+1}^n \sum_{l=i+1}^{j-1} f_{\pi_k \pi_l} + \sum_{k=j+1}^n f_{\pi_k \pi_j} + \\ & \sum_{l=1}^{i-1} f_{\pi_j \pi_l} + f_{j,i} + \sum_{l=i+1}^{j-1} f_{\pi_j \pi_l} + \\ & \sum_{k=i+1}^{j-1} \sum_{l=1}^{i-1} f_{\pi_k \pi_l} + \sum_{k=i+1}^{j-1} f_{\pi_k \pi_i} + \\ & \sum_{l=1}^{i-1} f_{\pi_i \pi_l} + \sum_{k=2}^{i-1} \sum_{l=1}^{k-1} f_{\pi_k \pi_l} + \\ & \sum_{k=i+2}^{j-1} \sum_{l=i+1}^{k-1} f_{\pi_k \pi_l} + \sum_{k=j+2}^{n-1} \sum_{l=j+1}^{k-1} f_{\pi_k \pi_l}, \end{aligned}$$

and

$$\begin{aligned} c_2(\bar{\pi}^f) = & \sum_{k=j+1}^n \sum_{l=1}^{i-1} f_{\pi_k \pi_l} + \sum_{k=j+1}^n \sum_{l=i+1}^{j-1} f_{\pi_k \pi_l} + \\ & \sum_{k=j+1}^n f_{\pi_k \pi_j} + \sum_{k=j+1}^n f_{\pi_k \pi_i} + \\ & \sum_{l=1}^{i-1} f_{\pi_i \pi_l} + \sum_{l=i+1}^{j-1} f_{\pi_i \pi_l} + f_{\pi_i \pi_j} + \\ & \sum_{l=1}^{i-1} f_{\pi_j \pi_l} + \sum_{l=i+1}^{j-1} f_{\pi_j \pi_l} + \\ & \sum_{k=i+1}^{j-1} \sum_{l=1}^{i-1} f_{\pi_k \pi_l} + \sum_{k=2}^{i-1} \sum_{l=1}^{k-1} f_{\pi_k \pi_l} + \\ & \sum_{k=i+2}^{j-1} \sum_{l=i+1}^{k-1} f_{\pi_k \pi_l} + \sum_{k=j+2}^{n-1} \sum_{l=j+1}^{k-1} f_{\pi_k \pi_l}. \end{aligned}$$

Then  $\Delta_{c_2}^f$  can be obtained by subtracting  $c_2(\bar{\pi}^f)$  from  $c_2(\bar{\pi})$ , resulting in many cancellations and finally in the simple expression of (9).  $\Delta_{c_2}^b$  can be obtained similarly. Notice that (9) and (10) can be used for the BUCLP with  $c_1(\pi)$  or  $c_2(\pi)$  since both objectives are equivalent. However, this is not possible for the unbalanced UCLP.

In their early work Bozer and Rim [6], later on Kiran, Ünal and Karabati [12] have shown that the cost  $c_1(\pi)$  of a layout  $\pi$  is independent of where the workstations are located around the unidirectional cyclic material handling system when the part flow is balanced. In other words the predetermined locations can be shifted without modifying the value of the objective function as long as the layout, namely the sequence of the workstations around the unidirectional cyclic material handling system, remains the same. In short, when the part flow is balanced, even the locations are nonequidistant, the problem becomes equivalent to the determination of a cyclic permutation of the workstations around the unidirectional circular material handling system, which minimizes total material transport. In fact it has been shown that the BUCLP and EUCLP are equivalent [6], [12].

#### A. Afentakis' BUCLP formulation

In his seminal paper, Afentakis [2] proposed a Binary Integer Linear Programming (BILP) formulation for the BUCLP which he called as the Loop Interconnection Problem (LIP). The author assumed that parts enter and exit the system through a LUL area and complete an integer number of tours around the loop before leaving

the system, which makes the material flow balanced. LIP is given below:

$$\text{LIP: } \min \sum_{(i,j) \in A}^n f_{ij}(1 - y_{ij}) \quad (11)$$

$$\text{s.t. } y_{ij} + y_{ji} = 1 \quad \forall i, j; i \neq j \quad (12)$$

$$y_{ik} + y_{kj} - 1 \leq y_{ij} \quad \forall i, j, k; i \neq j \neq k \quad (13)$$

$$y_{ij} = 0, 1 \quad \forall i, j; i \neq j \quad (14)$$

where

$$y_{ij} = \begin{cases} 1 & \text{if workstation } j \text{ is located after} \\ & \text{workstation } i \text{ in an optimal layout .} \\ 0 & \text{otherwise} \end{cases}$$

As noted by Afentakis [2], the objective function (11) minimizes the total number of times the parts cross the LUL area. Constraints (12) are symmetry constraints and guarantee that either workstation  $i$  is located before workstation  $j$ , or vice versa. Constraints (13) are transitivity constraints and state if workstation  $i$  is located before workstation  $k$  and workstation  $k$  is located before workstation  $j$ , then workstation  $i$  must be located before workstation  $j$ .

### B. Kiran, Ünal and Karabati 's BUCLP formulation

Later, Kiran, Ünal and Karabati [12] have developed an extended formulation for the BUCLP. They define the decision variables  $d_{ij}$  denoting the circular distance from workstation  $i$  to workstation  $j$  and the following auxiliary binary decision variables

$$x_{ij} = \begin{cases} 1 & \text{if workstation } j \text{ immediately follows} \\ & \text{workstation } i \text{ in an optimal layout} \\ 0 & \text{otherwise} \end{cases}$$

The distances can be scaled by dividing the constant circuit length  $\lambda$ . This does not affect the solution of the problem. Thus, it is possible to assume that  $\lambda = 1$  and  $0 \leq d_{ij} \leq 1$  without loss of generality. The cost parameters  $f_{ij}$  state the total part flow from workstation  $i$  to workstation  $j$ . Then their BILP formulation for the

BUCLP is given as

$$\text{BUCLP: } \min \sum_{i=1}^n \sum_{\substack{j=1 \\ i \neq j}}^n f_{ij} d_{ij} \quad (15)$$

$$\text{s.t. } \sum_{\substack{j=1 \\ i \neq j}}^n x_{ij} = 1 \quad \forall i \quad (16)$$

$$\sum_{\substack{i=1 \\ i \neq j}}^n x_{ij} = 1 \quad \forall j \quad (17)$$

$$d_{ij} + d_{ji} = 1 \quad \forall i, j; i \neq j \quad (18)$$

$$d_{kj} \geq d_{ki} + d_{ij} + x_{ij} - 1 \quad \forall i, j, k; i \neq j \neq k \quad (19)$$

$$x_{ij} = 0, 1 \quad \forall i, j; i \neq j \quad (20)$$

$$0 \leq d_{ij} \leq 1 \quad \forall i, j; i \neq j. \quad (21)$$

In this formulation, the objective function (15) is to minimize the total travel distance of all parts entering into the system. Constraints (16) and (17) are exactly the assignment constraints and ensure that each workstation has exactly one predecessor and one successor. Constraints (18) and (19) define the properties of the distance matrix. Also (19) guarantees that  $d_{ij} = d_{ik} + d_{kj}$  if workstation  $j$  immediately follows workstation  $k$  in the flow direction. Kiran, Ünal and Karabati [12] have shown that this is a valid BUCLP formulation. An interesting consequence of this formulation is the interpretation of the distance variables  $d_{ij}$ . The flow is balanced first of all. It is also assumed that parts enter and leave the system by the LUL area. As a result the objective function of the BUCLP becomes equivalent to  $c_2(\pi)$ , and counts the number of times parts pass through the LUL area, since the length of the UCL is 1. Therefore  $d_{ij}$  must behave as the indicator function (3) and be equal to 1 if workstation  $i$  comes after workstation  $j$  in a layout.

### III. THE BRANCH AND BOUND ALGORITHM

At the start of our branch and bound algorithm, we have an empty sequence of workstations and when the algorithm stops we obtain a complete sequence of workstations such that total material transportation cost is minimized. The first (last) position of the sequence corresponds to the first (last) candidate location immediately after (before) the Load/Unload area. At every node of the branching tree, we have two sets of workstations: assigned and unassigned. The set of assigned workstations are kept in a partial sequence, with fixed positions. However, the positions of the set of unassigned workstations are not determined.

When a new node is generated, one workstation from the unassigned set is chosen and included into the assigned set. In other words, every time a new node is generated in the branching tree we assign a workstation from the unassigned set, to the first leftmost available location of the partial sequence. The partial sequence of a node is kept of all of its descendant branching nodes. That is to say, the partial sequence of a node is always inherited from its ancestors. For every node a new partial sequence is obtained by choosing and locating an unassigned workstation to the first leftmost available position in the inherited partial sequence.

Moreover, for each node, we compute a lower and upper bound considering the assigned and unassigned sets of workstations. To determine the lower bound, we first compute the cost of assigned workstations. Then, we consider the costs due to the set of unassigned workstations. The upper bound value is the objective function value of the solution obtained with the heuristic procedure. During the search procedure, we keep the best upper bound value of all generated nodes.

In summary, at the very beginning of the algorithm,  $n - 1$  nodes are generated. For each of them, there exist only two assigned workstations in two leftmost positions of the corresponding partial sequence. Note that since for the balanced case  $R_i = \sum_{j,j \neq i} f_{ij} = \sum_{j,j \neq i} f_{ji} = C_i$

$\forall i = 1, \dots, n$ , our concern is the relative ordering of workstations rather than the assignment of workstations to candidate location. To better explain this, consider the objective function values of sequences  $\pi_1 = (1, 2, \dots, n)$  and  $\pi_2 = (2, \dots, n, 1)$ . Both of these values are equal since  $\sum_{j,j \neq 1} f_{1j} = \sum_{j,j \neq 1} f_{j1}$  holds because of the balanced material flow assumption. Therefore, without loss of generality, we locate workstation 1 in the first leftmost position and  $n - 1$  other workstations, namely workstations  $2, \dots, n$ , in the second leftmost positions of the partial sequences. Next, we run branching and bounding procedures consecutively. The lower and upper bounding procedures are explained below. The nodes whose lower bound values are greater than the current best upper bound values are discarded for further consideration. Then, the node with the lowest lower bound value is chosen for further branching to generate new nodes. These steps continue until we have only one node with equal lower and upper bound values.

To better illustrate these steps, we present with Figure 6 a branch and bound subtree for an instance with four workstations. At the first node, namely at Node 0, we assign workstation  $i$  into the first location. In other words, we have the partial sequence  $\{i\}$  for Node 0. Node 1, Node 2 and Node 3 are in the second level of the

tree and they are branches of Node 0. Workstations  $j$ ,  $k$  and  $l$  are respectively assigned into the second available locations. Their partial sequence are respectively  $\{i, j\}$ ,  $\{i, k\}$  and  $\{i, l\}$ . Node 4, Node 5, Node 6, Node 7, Node 8 and Node 9 are in the third level of the tree and they represent all possible solutions with workstation  $i$  assigned to location 1.

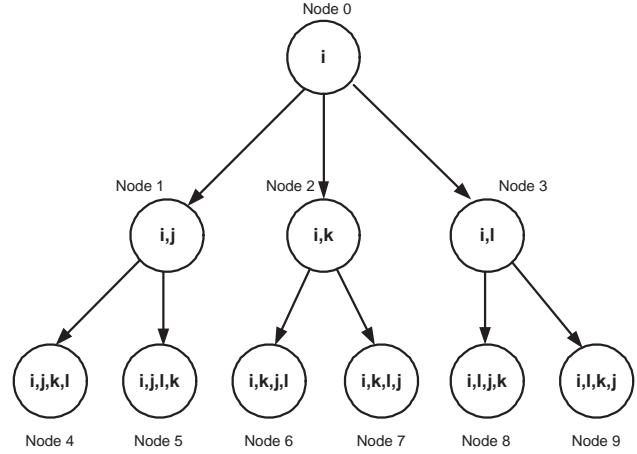


Fig. 6. The enumeration tree of a 4-workstation example

#### A. Preprocessing Steps

The preprocessing steps constitute the initialization phase of our branch and bound algorithm. These procedures are run at the very beginning of the algorithm with the purpose to reduce the size of the material flow matrix, and therefore the size of the problem.

Kouvelis and Kim have proposed the preprocessing steps in their seminal paper [15] on the UCLP. In their work, to reduce the size of the problems, the preprocessing phase consists of searching for the workstations which have only inflows and/or outflows. They have shown that in an optimal solution, a workstation  $i$  that has only inflow (outflow) is always located at the last (first) candidate location in an optimal sequence.

In addition to this rule, we also detect workstations  $i$  with  $f_{ij} = f_{ji}$  for all  $j = 1, \dots, n$  as proposed by Lee, Huang and Chiang [16]. They have noticed the optimal layout is independent of the position of such workstations. As a consequence, these workstations should be detected at the beginning of the algorithm.

#### B. Lower Bound

To compute the lower bound values we need to consider three flow quantities:

- 1) flows between assigned workstations,

- 2) flows between unassigned-assigned workstation pairs, and
- 3) flows between unassigned workstations.

The flows between assigned workstations and the flows between unassigned-assigned workstations pairs can be directly computed since the actual position of assigned workstation is known. In fact, to directly compute these two flow quantities we use the following observation. For a given sequence of workstations, the workstation located in the first position contributes to the objective function value as much as the sum of the inflows from all other workstations. Namely, if workstation  $i$  is located in the first position, the contribution to the total cost due to workstation  $i$  is  $C_i = \sum_{k,k \neq i} f_{ki}$  where  $f_{ki}$  is the fixed flow quantity from workstation  $k$  to workstation  $i$ . Moreover suppose that, workstation  $j$  is located in the second position of that partial sequence. Then, the contribution to the total objective function is  $(\sum_{k,k \neq j} f_{kj}) - f_{ij} = (C_j) - f_{ij}$ . In other words, it is equal to the total amount of flow to workstation  $j$  except the inflow quantity from workstation  $i$  to workstation  $j$ .

As an illustrative example, consider a sequence  $\{i, k, l, j\}$  of four workstations. Our objective is to calculate the objective function  $\sum_i \sum_j f_{ij} d_{ij}$ . Recall that  $d_{ij} = 1$  if workstation  $i$  is located after workstation  $j$  and the LUL area is located at some position on the path from workstation  $i$  to workstation  $j$ . For the given sequence of workstation the objective function value is

$$C_i + (C_k - f_{ik}) + (C_l - f_{il} - f_{kl}) \quad (22)$$

where  $C_i$  is the sum of column  $i$  of the flow matrix, i.e.  $C_i = \sum_{j,j \neq i} f_{ji}$ .

To calculate a bound for the third case, namely the flow corresponding to the unassigned workstations, may be much more complicated than the first two cases. One way to give a bound for the third case is to solve the LP relaxation of the BUCLP formulation by Afentakis [2]. One disadvantage of this method is its dependence on the efficiency of the commercial LP solver used. Another alternative is to perform the lagrangian relaxation of this formulation. By using the similarity of the BUCLP with the LOP and other sequencing problems we decided to follow the lower bounding method proposed by Potts [19] for the Precedence Constrained Single Machine Scheduling Problem. When there is no precedence constraints this problem is to find a sequence of jobs which minimizes the sum of completion times. In other words, it turns out to find a linear ordering of jobs on a single machine.

Before explaining the steps of the lagrangian relaxation scheme lets consider the formulation by Afentakis [2]. First, notice that we can rewrite the transitivity constraints (13) as  $(1 - y_{ki}) + (1 - y_{jk}) - 1 \leq y_{ij}$ . Then by rearranging the terms in the left hand side we obtain  $1 \leq y_{ij} + y_{jk} + y_{ki}$ . Note that together with the symmetry constraints (12) and appropriate objective function we obtain the well-known Linear Ordering Problem formulation [21]. At this stage, we are faced with the problem of deciding which constraint set should be relaxed. Note that when we relax the transitivity constraints (13), the remaining symmetry constraints forms a Totally Unimodular Matrix. On the other hand, from the lagrangian relaxation theory, we know that we will never reach a lower bound better than the Linear Programming relaxation bound of the original formulation (i.e. LIP formulation) [17]. Therefore, we associate lagrangian multipliers  $\alpha_{ij}$  for symmetry constraints (12). Then, we obtain the following relaxed formulation of the BUCLP:

$$\begin{aligned} c_2^{(0)} &= \sum_{i=1}^n \sum_{\substack{j=1 \\ j \neq i}}^n f_{ij} y_{ji} = \\ &\sum_{i=1}^n \sum_{\substack{j=1 \\ j \neq i}}^n f_{ij} y_{ji} + \sum_{i=1}^n \sum_{\substack{j=1 \\ j \neq i}}^n (1 - y_{ij} - y_{ji}) \alpha_{ij} \\ \text{s.t. } y_{ij} + y_{jk} + y_{ki} &\geq 1 \quad \forall i, j, k; i \neq j \neq k \\ y_{ij} &= 0, 1 \quad \forall i, j; i \neq j \end{aligned}$$

where the lagrangian multipliers  $\alpha_{ij}$  are set to

$$\alpha_{ij} = \alpha_{ji} = \frac{\min\{f_{ij}, f_{ji}\}}{2}.$$

The reduced cost matrix  $F^{(0)} = (f_{ij}^{(0)})$  consists of  $f_{ij}^{(0)} = f_{ij} - \alpha_{ij} - \alpha_{ji}$ . Note that either  $f_{ij}^{(0)} = 0$  or  $f_{ji}^{(0)} = 0$ . Then, the transitivity constraints  $y_{ij} + y_{jk} + y_{ki} \geq 1$  for  $i, j, k = 1, \dots, n; i \neq j \neq k$  are consecutively introduced. Whenever a constraint is considered in the context of lagrangian relaxation we update the objective function value,  $c_2$ . Broadly speaking, when the  $(t-1)^{th}$  transitivity constraint is introduced, the lagrangian objective function value will be

$$c_2^{(t-1)} = \sum_{i=1}^n \sum_{\substack{j=1 \\ i \neq j}}^n f_{ij}^{(t-1)} y_{ji} + \sum_{i=1}^n \sum_{\substack{j=1 \\ i \neq j}}^n \alpha_{ij} + \sum_{p=1}^{t-1} \beta^{(p)} \quad (23)$$

Next, at the  $t^{th}$  step, we update the lagrangian function with

$$c_2^{(t)} = c_2^{(t-1)} + \beta^{(t)} (1 - y_{ij} - y_{jk} - y_{ki}) \quad (24)$$

where  $\beta^{(t)} = \min\{f_{ji}, f_{kj}, f_{ik}\}$ . Thus  $\beta^{(t)}$  is set as large as possible while keeping the elements of the reduced

cost matrix  $F^{(t)}$  nonnegative. The value of  $\beta^{(t)}$  gives the increase in the lower bound. In other words, the bigger  $\beta^{(t)}$  is, the larger increase we have in the lagrangian relaxation function value (23). On the other hand, when there is no  $\beta$  multiplier with positive value we reach to a feasible ordering of workstations and the reduced cost matrix can be used to obtain this feasible ordering, as pointed by Potts [19].

When we find a  $\beta^{(t)}$  multiplier with positive value, the elements of the reduced cost matrix  $F^{(t-1)}$  corresponding to  $y_{ij}$ ,  $y_{jk}$  and  $y_{ki}$  are updated and we obtain  $F^{(t)}$ . In other words we perform the following operations:

$$\begin{aligned} f_{ji}^{(t)} &= f_{ji}^{(t-1)} - \beta^{(t)} \\ f_{kj}^{(t)} &= f_{kj}^{(t-1)} - \beta^{(t)} \\ f_{ik}^{(t)} &= f_{ik}^{(t-1)} - \beta^{(t)}. \end{aligned}$$

To improve the lower bound we must consider transitivity constraints (13) as much as possible. However, the decision of which transitivity constraints should be considered first, may greatly affect the efficiency of the lower bounding procedure. One criterion may be to consider them in increasing order of indices  $i < j < k$ . Another criterion is to consider the sequence generated by the upper bounding procedure. Then the transitivity constraints satisfied by this sequence are consecutively considered. With the second criterion, the first few constraints introduced, contribute better to the lower bound than the first few constraints considered with the first criterion. In this work, we prefer to apply the second criterion with the hope to obtain stronger lower bounds.

Now, we can formally present our lower bounding procedure. For any workstation  $j$ ,  $j = 1, \dots, n$ , let  $Z_{\{i\}}^j$  be the amount of part flow into workstation  $j$ , except  $f_{ij}$  where  $\{i\}$  is the workstation located before workstation  $j$ . In other words,  $Z_{\{i\}}^j = C_j - f_{ij}$  where  $C_j$  is the  $i^{th}$  column's sum of the part flow matrix. Consider the case where a sequence of two workstations, namely  $\{i, k\}$ , is located before workstation  $j$ . The amount of part flows into workstation  $j$ , except  $f_{ij}$  and  $f_{kj}$ , is  $Z_{\{i,j\}}^j = C_j - f_{ij} - f_{kj}$ . For a partial sequence of workstations  $\{i, j, k\}$  we have the lower bound  $LB = C_i + Z_{\{i\}}^j + Z_{\{i,k\}}^j + lb(k)$ , where  $lb(k)$  is the bound obtained for unassigned workstations, namely all workstations except  $i$ ,  $j$  and  $k$ , using the lagrangian relaxation. For instance consider the branch and bound subtree of Figure 6. For Node 0 and Node 1 the lower bounds are respectively,  $LB_0 = C_i + lb(i)$  and  $LB_1 = C_i + (C_j - f_{ij}) + lb(i, j)$ . For Node 4, the lower bound

is

$$\begin{aligned} LB_4 &= C_i + (C_j - f_{ij}) + (C_k - f_{ik} - f_{jk}) + lb(i, j, k) \\ &= C_i + (C_j - f_{ij}) + (C_k - f_{ik} - f_{jk}) \\ &\quad + (C_l - f_{il} - f_{jl} - f_{kl}) \\ &= C_i + (C_j - f_{ij}) + (C_k - f_{ik} - f_{jk}). \end{aligned}$$

### C. Upper Bound

To compute the upper bounds, we temporarily complete the partial sequence with the set of unassigned workstations. To assign the unassigned workstations into candidate (available) locations in the rest of the partial sequence, we use the KK3 heuristic proposed by Kouvelis and Kim [15]. As a result we have a complete sequence of workstations. Then, we perform the Move-Reverse (MR) heuristic of Altinel and Öncan [3] to improve the newly inserted set of workstations. Note that, we perform the KK3 and MR without harming the initial partial sequence.

KK3 starts with the part flow matrix  $F$  and determines workstation pair  $(i^*, j^*)$  which gives the largest

$$RC_{ij} = (R_i - C_i) - (R_j - C_j) + f_{ji} - f_{ij} \quad (25)$$

value. Here,  $R_i$  and  $C_i$  are respectively obtained by summing up the entries of the  $i^{th}$  row and  $i^{th}$  column of  $F$ . They denote total outflow and total inflow for workstation  $i$ . As for  $f_{ij}$  and  $f_{ji}$ , they are the number of parts processed in workstation  $i$  per unit time that must be routed to workstation  $j$ , and vice versa. These two workstations, namely  $i^*$  and  $j^*$ , are respectively assigned to the first and the last available locations. Then rows  $i^*$  and  $j^*$ , and columns  $i^*$  and  $j^*$  are deleted from  $F$  and a new  $(i^*, j^*)$  is determined by using (25) on the new part flow matrix  $F$ . These steps are repeated until the part flow matrix consists of a single element. KK3 yields a layout where workstations with higher outflow rate are located at the beginning and workstations with higher inflow rate at the end of a layout (i.e. workstation sequence).

In MR, the *move* heuristic and the *reverse* operation alternate until no further decrease in the objective function is possible.

Given the current assignment of  $n$  workstations to  $n$  available locations (i.e. a layout  $\pi$ ), the *move* heuristic computes the change in the objective value that results because of moving any workstation  $\pi_i$  located at location  $i$  to any of the locations  $j \neq i$ . This is done for every workstation and the improvements are recorded. Then the *move* operation resulting in the largest improvement is realized. The procedure is repeated until no improvement is possible. Moving workstation  $\pi_i$  from location  $i$

to location  $j$  consists of a sequence of location changes. First workstation  $\pi_i$  is moved to location  $j$ . Then, if  $j < i$  workstations  $\pi_j, \pi_{j+1}, \dots, \pi_{i-1}$  at locations  $j, j+1, \dots, i-1$ , are shifted forward to locations  $j+1, j+2, \dots, i$ . If  $i < j$ , backward shift occurs for workstations  $\pi_{i+1}, \pi_{i+2}, \dots, \pi_j$ , from locations  $i+1, i+2, \dots, j$  to locations  $i, i+1, \dots, j-1$ . Recall that the changes in the objective function values due to forward and backward move operations are respectively given by expressions (9) and (10) in Section II.

The *reverse* operation simply reverses the order of workstations of a given layout. In particular when applied to the layout obtained by the *move* heuristic, usually results in a layout with a larger objective function value. However, when the *move* heuristic is implemented on the reversed layout, a new layout which is better than the one of the previous *move* heuristic has been given, can be obtained. In other words, *reverse* somewhat represents an uphill move to escape from a local optimum solution obtained by *move* heuristic. Moreover, it has been shown that the *move* heuristic following a *reverse* operation does not worsen the layout the *move* heuristic preceding the *reverse* operation gives [3].

To better illustrate this, consider Node 1 in Figure 6. In Node 1, the set of assigned workstations  $i$  and  $j$ , are respectively assigned in the first and second available locations in the partial sequence. To compute the upper bound value of Node 1 we need to obtain a complete sequence of all four workstations. In other words, we need to temporarily locate the unassigned workstations  $k$  and  $l$  to the candidate positions, namely the third and fourth positions. We first run KK3 to construct a sequence then MR to improve the layout for the rest of the partial sequence. Suppose that MR outputs  $\{l, k\}$ . Then the feasible solution can be represented with the solution  $\{i, j, l, k\}$  and the upper bound value is  $UB_1 = C_i + (C_j - f_{ij}) + (C_l - f_{il} - f_{jl})$ .

#### D. Dominance Rules

Dominance rules help to identify the set of dominant sequences. Given two sequences  $\pi$  and  $\bar{\pi}$ ,  $\pi$  dominates  $\bar{\pi}$ , means that the objective function value obtained with  $\pi$  is better than the one obtained with  $\bar{\pi}$ . For a feasible instance of the problem, there is at least one sequence which dominates other sequences.

In this section we first present existing dominance rules, which are based on the notion of location interchanges. Then we propose a new dominance rule based on the notion of move operations. An interchange operation is the basic action of the famous *swap* heuristic which is widely used for solving the facility layout

problems [7]. Consider the initial sequence

$$\pi = (1, \dots, i-1, i, i+1, \dots, j-1, j, j+1, \dots, n) \quad (26)$$

and the sequence obtained by interchanging the locations of workstations  $i$  and  $j$

$$\bar{\pi} = (1, \dots, i-1, j, i+1, \dots, j-1, i, j+1, \dots, n). \quad (27)$$

Then the change in the cost function is

$$\begin{aligned} \Delta c_2 = c_2(\bar{\pi}) - c_2(\pi) = \\ \sum_{k=i+1}^{j-1} \{(f_{kj} - f_{jk}) + (f_{ik} - f_{ki})\} + f_{ij} - f_{ji}. \end{aligned} \quad (28)$$

where  $c_2$  is the objective function of the UCLP. Clearly when  $\Delta c_2$  is positive then we have a positive gain from this interchange operation. Kouvelis and Kim [15] uses (28) to develop new dominance rules for identifying local optimal solutions. Based on these rules they have devised three construction heuristics: KK1, KK2 and KK3. KK3, is the best of them. Later, Kiran and Karabati [11] have used formula (28) to propose General Dominance (GD) and Adjacent Dominance (AD) rules which they used within their branch and bound algorithm. The GD rule is applied within the branch and bound algorithm when a new node in the branching tree is generated. Every time a new node is generated, a workstation  $i$  from the unassigned set is located at the first available location of the partial sequence. With the GD rule, we first compute all the interchange costs of workstation  $i$  with the workstations of the partial sequence, and in case of a positive gain, we do not consider this newly generated node for further consideration. One weakness of the GD is its CPU time requirement. Considering this, Kiran and Karabati [11] have come up with another dominance rule: AD which is a simplified version of the GD rule. They consider only adjacent workstations. In the branch and bound algorithm, the AD rule computes the interchange cost of workstation  $i$  and the workstation immediately before workstation  $i$  in the partial sequence. In case this value is positive, the new node is fathomed. The AD rule is also used by Lee, Huang and Chiang [16] within their branch and bound algorithm.

Both GD and AD are interchange based dominance rules. Now we propose a new dominance rule based on the idea of move operations. According to extensive computational experiments, Tansel and Bilen [22] have observed that the *move* heuristic is more efficient than the *swap* heuristic. This is the basic result that the move based dominance rule is expected to be more efficient than GD and AD rules. Moreover, a move operation is more elementary than an interchange operation; and

every interchange operation can be performed by two move operations.

In our branch and bound algorithm, we only apply the move based dominance rules when a new workstation  $i$  is assigned to the last available position of the partial sequence. We efficiently compute the cost of all possible forward and backward move operations of workstation  $i$  through the partial sequence by using formula (9) and (10), respectively. In case a positive gain is detected in the objective function value we do not consider this node for further branching namely, this node is fathomed. For instance consider Node 1. At this node we have assigned workstation  $j$  to the available position at the end of the partial sequence  $\{i\}$  of Node 0, and we have obtained partial sequence  $\{i, j\}$ . However, if the cost of partial sequence  $\{j, i\}$ , which is obtained by moving workstation  $j$  into location  $i$ , is better than the partial sequence  $\{i, j\}$ , then the partial sequence  $\{i, j\}$  must be reordered as  $\{j, i\}$  in the optimal sequence. As a result, we do not consider Node 1 for further branching.

#### IV. THE DYNAMIC PROGRAMMING ALGORITHM FOR THE BUCLP

The dynamic programming approach presented in this section is based on the ideas of the seminal works by Held and Karp [8] and Karp and Held [10] for sequencing problems. Their dynamic programming scheme is extended to the one dimensional space allocation problem, [18], the row layout problem [14] and the single machine scheduling problem [1].

Consider a subset  $S$  of the set of workstations  $N = (1, 2, \dots, n)$  with cardinality  $|S|$  and  $\bar{S} = N \setminus S$ . Let  $Z(S)$  denote the optimal objective value of the BUCLP formulation such that the set of workstations are located in the leftmost position of the sequence (e.g.  $\pi = (\pi_1, \pi_2, \dots, \pi_{|S|})$ ). For the first stage, we set the boundary condition

$$Z(\{i\}) = 0 \quad \forall i \in N \quad (29)$$

Then the recursive relationship

$$Z_i(S_{k+1} = S_k \cup \{i\}) = \min_{i \in S_k} (Z_j(S_k)) + \left\{ \sum_{l \in S_k} f_{il} \right\} \quad (30)$$

is valid for stage  $|S_k| + 1$ , where  $|S_k|$  is the cardinality of the subset  $S_k$  which consists of  $k$  workstations and  $i \in \bar{S}_k$

The optimal solution is reached by backtracking from stage  $n$ . That is to say, the optimum solution is given by  $Z(N)$ . Karp and Held [10] have shown that by

using the recursive relationship (30) we can reach to the optimum in  $O(n2^n)$  operations. As noted by Hubert, Arabie and Meulman [9], the dynamic programming implementations of this type are extremely memory resource dependent. Indeed, we were able to solve BUCLP instances of size up to 20 workstations with our hardware platform of 1GByte memory.

#### V. COMPUTATIONAL RESULTS

In an FMS environment, workstations are interconnected by a material handling system and able to process different part types simultaneously. In our experiments we consider four different FMS environments with respectively 20, 30, 40 and 50 workstations interconnected by a unidirectional cyclic material handling system. We also fix the number of different part types to 50.

Let  $\mathcal{P} = \{1, \dots, P\}$  be the set of part types and  $f_{ij}^p$  be the total number of parts of type  $p$  that flow from station  $i$  to station  $j$  per period of time. Notice  $P=50$  in our test bed. Then,

$$f_{ij} = \sum_{p \in \mathcal{P}} f_{ij}^p \quad i, j = 1, \dots, n; i \neq j \quad (31)$$

determines the part flow per time period from station  $i$  to station  $j$ . When the flow is balanced  $f_{ij}^p$  can be expressed as

$$f_{ij}^p = n_p n_{ij}^p \quad i, j = 1, \dots, n; i \neq j; p \in \mathcal{P} \quad (32)$$

where  $n_p$  denotes the number of parts of type  $p$  to be processed in the system and  $n_{ij}^p$  denotes the number of moves part type  $p$  makes from station  $i$  to station  $j$  per period of time, since no part is lost and no new part is created during the process to cause an unbalance in the flow. Each part type may have a different process plan. The process plan  $S_p$  is the sequence in which part type  $p$  visits the workstations. Notice that in any process plan a part can visit a workstation more than once, and hence a workstation's number can appear more than one time in  $S_p$ , but not consecutively. Hence,  $n_{ij}^p$  specifies the number of times workstations  $i$  and  $j$  appear consecutively ( $i$  immediately before  $j$ ) in the process plan  $S_p$ . We have generated randomly 30 process plans, for each one of 50 part types, for our four FMS environments. For this purpose we first created uniform positive integer numbers between 1 and the number of workstations, (i.e. 20, 30, 40 or 50) to determine the number of workstations to be visited by this part type. This actually gives the size of a linear array representing a process plan, which we fill entry by entry by uniform positive integers between 1 and the number of workstations while preventing the same integer from occurring consecutively more than one time. This makes

$30 \times 50 \times 4 = 600$  process plans at sum; and they are all used in the generation of all instances.

As it can be observed when the part flow is balanced formula (32) applies and the value of  $n_p$  has a direct effect to the magnitude of  $f_{ij}$ . Hence, it is possible to control the range of numbers in the part flow matrix by means of  $n_p$ . We generate three sets of  $n_p$  with  $p = 1, \dots, 50$  respectively from uniform distributions  $U(1,10)$ ,  $U(1,50)$  and  $U(1,100)$  as done by Tansel and Bilen [22]. They correspond to low, medium and high variation part flows. We will mark the instances with letters L, M and H to identify the type of variation they have, in the sequel. We first use formula (32) then formula (31) to obtain balanced part flows between workstation pairs. We generate 10 test problems per variation type.

In each set, instances are grouped according to the type of variation their flows have. Each group is formed by four 10-problem packages respectively with 20, 30, 40 and 50 workstations. For example the instance Bal20-M3 is the third instance of the 10-problem package with 20 workstations and medium variation in part flows. In short, there are 40 test instances for each variation type, which makes a test bed of 120 test instances.

The results of our dynamic programming approach are presented in Table I. We report the CPU time as the performance measure of the algorithm. The first, third and fifth columns of these tables stand for instances. With the second, fourth and sixth columns only reports the CPU times for instances with 20 instances. We could not solved instances with larger size because of our limited memory storage (1 GByte).

The results of our branch and bound algorithm are summarized in Tables II – V. We report both the number of nodes explored and the CPU time as computational performance measures. The first, fourth and seventh columns of these tables stand for instances. These instances are randomly generated as we explained above. The second, fifth and eighth columns give the number of nodes explored until the optimal solution is reached. The third, sixth and ninth columns are the total CPU times in seconds.

Considering the overall CPU time averages of the instances in Tables I and II which are 58.11 and 20.67 secs. respectively, it may seem that the dynamic programming approach is more advantageous than the branch and bound approach. However, this is not the case since the minimum and maximum CPU times are 6.11 and 271.16 secs. for the branch-and-bound algorithm and, 15.09 and 29.93 secs. for the dynamic programming algorithm. Moreover, we can not say that the dynamic programming

TABLE I  
CPU TIMES (SEC.) FOR THE DYNAMIC PROGRAMMING ALGORITHM

Instance	CPU	Instance	CPU	Instance	CPU
Bal20-L1	22.46	Bal20-M1	21.76	Bal20-H1	16.37
Bal20-L2	15.09	Bal20-M2	19.27	Bal20-H2	27.28
Bal20-L3	19.63	Bal20-M3	22.12	Bal20-H3	27.72
Bal20-L4	19.71	Bal20-M4	22.64	Bal20-H4	29.93
Bal20-L5	21.70	Bal20-M5	16.65	Bal20-H5	19.34
Bal20-L6	15.30	Bal20-M6	22.67	Bal20-H6	19.26
Bal20-L7	17.47	Bal20-M7	21.99	Bal20-H7	16.01
Bal20-L8	18.06	Bal20-M8	20.64	Bal20-H8	22.75
Bal20-L9	21.89	Bal20-M9	23.40	Bal20-H9	23.48
Bal20-L10	17.48	Bal20-M10	16.47	Bal20-H10	21.54
Average	18.88	Average	20.76	Average	22.37

approach is superior to the branch and bound algorithm because of its excessive memory storage requirement.

We could only solve instances with up to 50 workstations, it is because the computational effort to find the optimum solution grows exponentially with the number of workstations. It is an expected result since BUCLP is naturally a difficult problem as most of the combinatorial optimization problems. However, the proposed branch and bound algorithm appears quite efficient to solve real-life problems.

As a last remark, we can say that as the variation in part flow matrix increases the number of nodes explored increases. This can be observed better for test sets with 20 workstations in Table II.

Finally, we should point out that, our codes are written using C++ programming language and all the computational tests are realized on a 1.7 GHz Pentium IV PC with a 1 GByte RAM.

TABLE II  
COMPUTATIONAL RESULTS OF THE BRANCH AND BOUND ALGORITHM FOR 20 WORKSTATION INSTANCES

Instance	Nodes Explored	CPU(sec.)	Instance	Nodes Explored	CPU(sec.)	Instance	Nodes Explored	CPU(sec.)
Bal20-L1	19	8.90	Bal20-M1	19	6.50	Bal20-H1	19	7.01
Bal20-L2	19	8.63	Bal20-M2	428	173.18	Bal20-H2	583	263.56
Bal20-L3	19	6.11	Bal20-M3	19	8.43	Bal20-H3	74	38.01
Bal20-L4	19	7.21	Bal20-M4	19	6.47	Bal20-H4	31	16.01
Bal20-L5	19	7.31	Bal20-M5	19	7.25	Bal20-H5	390	155.77
Bal20-L6	19	6.64	Bal20-M6	29	10.90	Bal20-H6	38	16.36
Bal20-L7	561	203.79	Bal20-M7	19	7.39	Bal20-H7	616	271.16
Bal20-L8	19	8.87	Bal20-M8	381	178.56	Bal20-H8	19	8.69
Bal20-L9	19	7.95	Bal20-M9	31	11.62	Bal20-H9	379	165.88
Bal20-L10	183	70.15	Bal20-M10	83	36.13	Bal20-H10	52	18.74
Average	89.6	33.56	Average	104.70	44.64	Average	220.10	96.12

TABLE III  
COMPUTATIONAL RESULTS OF THE BRANCH AND BOUND ALGORITHM FOR 30 WORKSTATION INSTANCES

Instance	Nodes Explored	CPU(sec.)	Instance	Nodes Explored	CPU(sec.)	Instance	Nodes Explored	CPU(sec.)
Bal30-L1	814	238.39	B30-M1	1515	642.69	B30-H1	934	334.64
Bal30-L2	1189	483.94	B30-M2	3703	1781.85	B30-H2	1428	432.83
Bal30-L3	1852	827.05	B30-M3	631	267.78	B30-H3	1803	778.36
Bal30-L4	2421	895.53	B30-M4	2484	1117.49	B30-H4	6472	3161.13
Bal30-L5	2727	969.67	B30-M5	2094	972.69	B30-H5	1093	374.96
Bal30-L6	915	282.92	B30-M6	1274	530.13	B30-H6	931	444.11
Bal30-L7	848	350.12	B30-M7	2260	756.87	B30-H7	1092	546.61
Bal30-L8	2683	1068.65	B30-M8	729	347.19	B30-H8	1946	631.30
Bal30-L9	979	365.02	B30-M9	2503	1020.68	B30-H9	2011	764.62
Bal30-L10	2510	1158.03	B30-M10	726	336.78	B30-H10	1163	454.25
Average	1693.80	663.93	Average	1791.90	777.42	Average	1887.30	792.28

TABLE IV  
COMPUTATIONAL RESULTS OF THE BRANCH AND BOUND ALGORITHM FOR 40 WORKSTATION INSTANCES

Instance	Nodes Explored	CPU(sec.)	Nodes Explored	CPU(sec.)	Nodes Explored	CPU(sec.)
Bal 40-L1	15320	4812.44	B40-M1	14238	3496.03	B40-H1
Bal 40-L2	18573	5296.63	B40-M2	13573	3649.73	B40-H2
Bal 40-L3	16429	6362.18	B40-M3	12840	3190.23	B40-H3
Bal 40-L4	13920	6300.49	B40-M4	16849	5670.41	B40-H4
Bal 40-L5	11392	4997.09	B40-M5	19579	7171.27	B40-H5
Bal 40-L6	18312	7530.94	B40-M6	20924	5248.25	B40-H6
Bal 40-L7	16327	6531.86	B40-M7	18420	5980.59	B40-H7
Bal 40-L8	18344	5107.44	B40-M8	21033	4865.35	B40-H8
Bal 40-L9	14232	6519.36	B40-M9	17364	6245.00	B40-H9
Bal 40-L10	12329	5213.65	B40-M10	15390	4768.33	B40-H10
Average	15517.80	5867.21	Average	17021	5028.52	Average
					18968.80	6958.17

TABLE V  
COMPUTATIONAL RESULTS OF THE BRANCH AND BOUND ALGORITHM FOR 50 WORKSTATION INSTANCES

Instance	Nodes Explored	CPU	Nodes Explored	CPU	Nodes Explored	CPU
Bal 50-L1	29932	12482.28	B50-M1	27429	12326.68	B50-H1
Bal 50-L2	25405	12107.16	B50-M2	28930	9112.58	B50-H2
Bal 50-L3	26430	14960.16	B50-M3	29341	14849.18	B50-H3
Bal 50-L4	25486	10080.09	B50-M4	25936	9703.56	B50-H4
Bal 50-L5	29504	15338.44	B50-M5	27305	7455.85	B50-H5
Bal 50-L6	27942	15299.75	B50-M6	26910	10755.22	B50-H6
Bal 50-L7	29153	13026.67	B50-M7	29027	13933.35	B50-H7
Bal 50-L8	25404	8313.35	B50-M8	28295	15465.80	B50-H8
Bal 50-L9	24195	6784.01	B50-M9	26539	8176.18	B50-H9
Bal 50-L10	28925	13042.66	B50-M10	31042	8283.16	B50-H10
Average	27237.60	12143.46	Average	28075.40	11006.16	Average
					28380.70	11787.72

## VI. CONCLUSIONS

In this work we have developed a branch and bound algorithm for the BUCLP. Our algorithm is specially designed for the balanced case of the UCLP. The special property of the cost matrix, i.e. balanced work flow matrix, gave a considerable advantage in our lower and upper bound computations and branching strategy. We have also proposed a dynamic programming algorithm for the BUCLP. The dynamic programming approach does also use the special work flow matrix property of the problem. However, it requires excessive computer memory and as a result, we were able to solve instances with up to 20 workstations. On the other hand, the branch and bound algorithm does not require as much memory storage as the dynamic programming algorithm. With the branch and bound algorithm we can solve instances with up to 50 workstations. Both of these approaches are special purpose algorithms for the BUCLP which explore the special structure of the problem.

We used the MR heuristic of Altinel and Öncan [3] as improvement phase in computing upper bounds. We can say that it is a remarkably efficient heuristic. Moreover, we also propose computationally efficient dominance rules by using the balanced work flow property of the problem and the basic move operation. These dominance rules are not only employed within the branch and bound algorithm but also within the dynamic programming approach.

The relation between the BUCLP and the Linear Ordering Problem is implied by Afentakis [2], and Potts and Whitehead [20]. There are many other combinatorial optimization problems with similar structures. The permutation flow-shop scheduling problem, single machine scheduling problem and single row layout problem are only a few of them. One research topic is the adaptation of techniques developed for the BUCLP to these relatives.

## ACKNOWLEDGEMENTS

This research has been partly supported by Boğaziçi Research Fund Grant No: 04HA301D. The first author also acknowledges the hospitality and support of the Computational Optimization Research Center at Columbia University during his stay as a visiting scholar.

## REFERENCES

- [1] T. S. Abdul-Razaq and C.N. Potts, 1988, Dynamic programming state-space relaxation for single-machine scheduling, *Journal of the Operational Research Society*, 39 (2), 141-152.
- [2] P. Afentakis, "A Loop Layout Design Problem for Flexible Manufacturing Systems". *International Journal of Flexible Manufacturing Systems*, vol. 1, no. 2, pp. 207-219, 1981.
- [3] İ. K., Altinel and T. Öncan, "The Design of Optimal Unidirectional Cyclic Layout", FBE-IE-06/2004-09, Institute for Graduate Studies in Science and Engineering, Boğaziçi University, Bebek, İstanbul, Türkiye, 2004.
- [4] P. Banerjee and Y. Zhou, 1995, Facilities layout design optimizatin with single loop material flow path configuration, *International Journal of Production Research*, 33 (1), 183-203.
- [5] J. A. Bennell, C.N. Potts and J.D. Whitehead, 2002, Local search algorithms for the min-max loop layout problem, *Journal of the Operational Research Society*, 53 (10), 1109-1117.
- [6] Y. A. Bozer and S.C. Rim, "Exact Solution Procedures for the Circular Layout Problem", Report No 89-33, University of Michigan, USA, 1989.
- [7] R.L. Francis, J.A. White and L.F. McGinnis, Facility Layout and Location: An Analytical Approach, Prentice – Hall, New Jersey, 1991.
- [8] M. Held and R.M. Karp, 1962, A Dynamic Programming Approach to Sequencing Problems, *Journal of the Society for Industrial and Applied Mathematics*, 10, 196-210.
- [9] L. Hubert, P. Arabie and J. Meulman, 2001, Combinatorial Data Analysis, *Optimization by Dynamic Programming*, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA.
- [10] R.M. Karp and M. Held, 1967, Finite-State Processes and Dynamic Programming, *SIAM Journal of Applied Mathematics*, 15, 693-718.
- [11] A.S. Kiran and S. Karabati, 1993, Exact and Approximate Algorithms for the Loop Layout Problem, *Production Planning and Control*, 4(3), 253–259.
- [12] A.S. Kiran, A.T. Ünal and S. Karabati, 1992, A Location Problem on Unicyclic Networks: Balanced Case, *European Journal of Operational Research*, 62, 194–202.
- [13] P. Kouvelis, W-C. Chiang and A.S. Kiran, 1992, A Survey of Layout Issues in the Flexible Manufacturing Systems, *OMEGA International Journal of Management Science*, 20(3), 375-390.
- [14] P. Kouvelis, W-C. Chiang and G. Yu, 1995, Optimal algorithms for row layout problems in automated manufacturing systems, *IIE Transactions*, 27, 99–104.
- [15] P. Kouvelis and M.W. Kim, 1992, Unidirectional Loop Network Layout Problem in Automated Manufacturing Systems, *Operations Research*, 40(3), 533-550.
- [16] S-D. Lee, K-S. Huang and C-P. Chiang, 2001, Configuring Layout in Unidirectional Loop Manufacturing Systems, *International Journal of Production Research*, 39(6), 1183-1201.
- [17] G.L. Nemhauser and L.A. Wolsey, Integer and Combinatorial Optimization, Wiley – Interscience, New York, 1988.
- [18] J. Pickard and M. Queyranne, 1981, On the one-dimensional space allocation problem, *Operations Research*, 29(2), 371–391.
- [19] C.N. Potts, 1985, A Lagrangean Based Branch and Bound Algorithm for Single Machine Sequencing with Precedence Constraints to Minimize Total Weighted Completion Time, *Management Science*, 31, 1300–1311.
- [20] C.N. Potts and J.D. Whitehead, 2001, Workload Balancing and Loop Layout in the Design of a Flexible Manufacturing System, *European Journal of Operational Research*, 129, 326-336.
- [21] G. Reinelt, 1985, *The Linear Ordering Problem: Algorithms and Applications: Research and Exposition in Mathematics*, vol. 8, In: Hoffmann H.H. and Wille R, (eds.), Heldermann Verlag, Berlin.
- [22] C.B. Tansel and C. Bilen, 1998, Move Based Heuristics for the Unidirectional Loop Network Layout Problem, *European Journal of Operational Research*, 108, 36-48.

# Multiobjective service restoration in electric distribution networks using a local search based heuristic

Vinícius Jacques Garcia\* and Paulo Morelato França\*

\* Faculdade de Engenharia Elétrica e de Computação  
 Universidade Estadual de Campinas - UNICAMP  
 Av. Albert Einstein, 400. C.P.: 6101 - 13083-852  
 Campinas, SP, Brazil  
 Email: {jacques,franca}@densis.fee.unicamp.br

**Abstract**—Contingency situations may cause emergency states in distribution systems; these states are defined as the interruption of power supply. The importance of the maintenance of the quality limits in relation frequency and duration of interruption means that such situations should be avoided whenever possible. The main objective of service restoration is to minimize the number of consumers affected by the fault, transferring them to distribution support feeders to restore power while maintaining electrical and operational conditions, such as radial network configuration, equipment and voltage drop limits. This paper presents a new local search based heuristic for the restoration of service which considers the minimization of the load not restored and of the number of switching operations involved. Computational experiments with three network systems have shown the flexibility and effectiveness of the proposed method.

**Keywords**—power systems operation, distribution systems, service restoration, optimization methods.

## I. INTRODUCTION

CONTINGENCY situations can affect power distribution systems and lead to a blackout state, which is caused by the interruption of the power supply for a portion of the network. Even if such a fault is restricted and correctly isolated, neighboring regions will be affected, thus reducing the indices that measure the quality of service and causing financial loss for utility companies. These aspects have led them to concern with all the issues relating to the delivery of reliable power to customers. Technical considerations suggest that reliability performance is closely related to frequency and duration of service interruption. It is precisely

these two aspects can be significantly improved via effective service restoration procedures.

The main objective of the solution of the Service Restoration Problem (SRP) is to minimize the number of customers faced with an interruption of power delivery by transferring them to support feeders via network reconfiguration, while respecting all operational and electrical constraints. One factor to be considered is the reaction time: outage areas should be restored as quickly as possible to avoid a lowering of the indices based on the duration of interruption.

As shown in the survey of [6], important work has addressed the SRP since the late 80s. The guidelines and operational procedures inspired by many of the first contributions were based on purely heuristic approaches [15] or artificial intelligence techniques (expert systems) [12]. Such heuristic approaches were used to solve the SRP for over a decade [18].

Recent papers, however, have addressed the inherently multiobjective nature of the SRP. Lee et al. [10] has incorporated the relevance of a variety of factors in a multiobjective methodology using fuzzy decision making. In the same year, a multiobjective heuristic method was introduced, made use of indices to guide a search towards a solution [14]. These indices made it possible to distinguish systematically between different network switches on the basis of analytically determined criteria. Matos and Melo [13] then introduced the well-known *Simulated Annealing* method for the network reconfiguration and service restoration based on minimizing the number of switching operations and maximizing the load supplied. Later, Augugliaro et al. [3] suggested a method that combines fuzzy sets with genetic algorithms in the consideration of two criteria: maximization of load supplied and minimization of power losses.

Moreover, Ceric and Popovic [5] developed a heuristic approach using mixed integer programming which was based on a single objective function involving a number of independent criteria.

The present paper proposes a new local search method associated with a constructive procedure to solve the SRP for radial networks through a multi-objective heuristic search methodology. The problem is defined in Section II, and the method described in Section III. Finally, computational experiments and conclusions are presented in Sections IV and V, respectively.

## II. PROBLEM DEFINITION

When one observes a real distribution network, it is possible to identify three well defined states [17]: the normal state, the emergency state and the restoration state. In the first, all loads are supplied within current and voltage limits. The emergency state is characterized by the activation of protective devices which leave some areas with no power supply. Restoration is that state characterized by attempting to find support feeders to reestablish the power supply to as many load as possible.

The SRP arises with the occurrence of a fault, switching the system from the normal to an emergency state in an attempt to identify support feeders which will be able to reduce the size of the out-of-service area, while respecting the constraints related to current and voltage limits.

The network reconfiguration proposed by a restoration plan should be minimal, since the emergency state is transitory existing only until the fault is eliminated. The SRP should thus be considered as a multiobjective optimization problem, since it must minimize both the load not supplied and the number of switching operations. The solution for this problem is a trade-off between these two criteria. Other aspects such as power losses and feeder load balance could be included, but are normally better left for consideration after normal operating conditions, whereas other constraints such as the line, power source, and voltage drop limits are included in the calculations to avoid the activation of protective devices.

In the following mathematical formulation based on [5] we define the SRP to minimize the number of switching operations and the load not restored in a scenario which considers limits on current, voltage and substation power, power balance constraints and the maintenance of a radial structure. In order to

avoid negative branch currents, imaginary branches were included (variable  $X'$ ).

$$\begin{aligned} \text{Min } z_1 = & \sum_{k \in F_{cs}} (1 - X_k) + \sum_{k \in F_{cs}} (1 - X'_k) + \\ & + \sum_{k \in F_{os}} X_k + \sum_{k \in F_{os}} X'_k \end{aligned} \quad (1)$$

$$\text{Min } z_2 = \sum_{k \in B} (1 - Z_k) L_k \quad (2)$$

subject to:

$$\sum_{k \in F_q} P_k \leq G_q, \quad \forall q \in S \quad (3)$$

$$P_k - I_{max}^{F_k} X_k \leq 0, \quad P'_k - I_{max}^{F_k} X'_k \leq 0, \quad \forall k \in F \quad (4)$$

$$|V_k^{min}| \leq |V_k| \leq |V_k^{max}|, \quad \forall k \in B \quad (5)$$

$$\sum_{k \in F_i} (P_k + P'_k) + \sum_{k \in T_i} (P_k + P'_k) \leq L_i Z_i, \quad \forall i \in B \quad (6)$$

$$\sum_{k \in T_i} (X_k + X'_k) \leq 1, \quad \forall i \in B \quad (7)$$

$$X_k + X'_k \leq 1, \quad \forall k \in F \quad (8)$$

where on this problem:

- $Z_k$  is an integer variable denoting energizing of load  $k$  (1) or its lack (0);
- $X_k$  is an integer variable denoting use of branch  $k$  (1) or its lack (0);
- $X'_k$  is an integer variable denoting use of imaginary branch  $k$  (1) or its lack (0);
- $P_k$  is a variable denoting the power flow in branch  $k$ ;
- $P'_k$  is a variable denoting the power flow in imaginary branch  $k$ ;
- $B$  is the set of all buses;
- $F$  is the set of all branches;
- $S$  is the set of all source nodes in the network;
- $F_i$  is the set of all branches whose initial node is  $i$ ;
- $T_i$  is the set of all branches whose terminal node is  $i$ ;
- $F_{os}$  is the set of all switches which are normally open;
- $F_{cs}$  is the set of all switches which are normally closed;
- $L_k$  is the load of bus  $k$ ;
- $G_q$  is the available power at source  $q$ ;

- $V_k$  is the voltage at bus  $k$ ;
- $I_{max}^{F_k}$  is the flow capacity at branch  $k$ ;
- $V_k^{min}/V_k^{max}$  is the minimum/maximum acceptable voltage drop at bus  $k$ .

In this multiobjective formulation, equation 1 refers to the minimization of the number of switching operations involved and equation 2 to the minimization of the load not supplied. The corresponding constraints are the equations 3-8. Substation limits are included in equation 3, whereas the equations 4 and 5 refer to the branch and voltage drop limits, respectively. Power balance between supply and demand is included in equation 6 and the radial configuration is guaranteed by equation 7. Finally equation 8 only permits the use of the real or of the imaginary branch.

A feasible solution for the SRP must maintain all network switch status so that the configuration implemented not violate any of these constraints. In fact, it is possible that there will be no restoration plan capable of energizing any of the out-of-service areas. When this happens the network configuration implemented by the activation of protective devices will be maintained until the fault is eliminated.

### III. THE LOCAL SEARCH BASED HEURISTIC METHOD

In this paper a multiobjective heuristic search method is proposed for the exploration of the search space of the SRP, denoted by all post-fault network configurations. This is a neighborhood-based method which systematically generates solutions from the transformation of others. The solutions generated, called neighbor solutions, are derived by a specific solution-generation mechanism.

Multiobjective optimization problems involve the simultaneous minimization (or maximization) of a set of conflicting criteria, while simultaneously satisfying a certain set of constraints [19]. This optimization procedure is based on a dominance relation in which, given two criteria vectors  $z^1$  and  $z^2$ , one says that  $z^1$  dominates  $z^2$  if  $z_j^1 \leq z_j^2$  for all of  $j$  objectives considered and  $z_j^1 < z_j^2$  for at least one  $j$ . When a point is not dominated by any other point, it is considered *non-dominated*; the set of all non-dominated points is known as *Pareto set* or an *efficient solution set*.

Some assumptions must be done about the SRP: (1) the distribution system is radial; (2) the pre-fault system state is known; and (3) the faults have been isolated.

Under these circumstances, the SRP is characterized by the occurrence of loads without a power

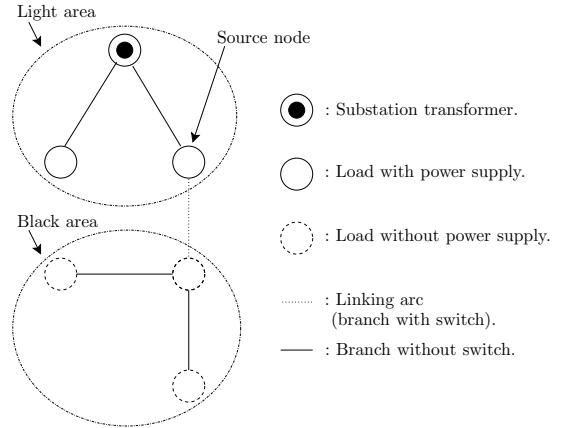


Fig. 1. SRP graph representation for an hypothetical network.

supply, leading to their disconnection from the energized network. In a graph representation [1] there is a *light area*, composed of all the loads where the power supply has been maintained, and a *black area*, including loads without power supply. Therefore, an SRP instance corresponds to a forest graph, with one tree for the light area and at least one other for the black area. The best case solution is to reestablish power supply for all loads in the black area.

Specially for solving the SRP, many other approaches have used evolutionary algorithms like Aoki et al. [2], Hsu and Kuo [9], and Augugliaro et al. [3]. The major concern is related to the maintenance of feasibility, since the problem has several constraints which are not easy to be included into evolutionary operators. Therefore, the proposed method attempts to overcome this obstacle by adopting a local search procedure with almost the same structure as described in Ehrgott and Gandibleux [7]. In addition, the initial non-dominated points are given by a constructive heuristic that considers one objective at a time; however in the local search phase all objectives are improved simultaneously. In order to promote better diversity, a certain number of neighborhoods are generated at the same iteration of the local search, like in Hansen [8].

The proposed method makes use of an other very important concept [20]: *source nodes*. These belong to the light area, with each one having at least one switch to connect it to the black area. These switches, or *linking arcs*, are included in the graph representation. Figure 1 shows the light and black areas, the source nodes and the linkings arc for an hypothetical network obtained after fault isolation.

This method of resolution is based on the appropriate use of source nodes to connect loads to the

---

```
MOLocalSearchBasedHeuristic( $IT_C, IT_L, N_{sol}, G_{op}$ )
1.  $PS = \text{Constructive}(IT_C, G_{op})$ ;
2.  $PS = \text{LocalSearch}(IT_L, N_{sol}, PS)$ ;
3. return( $PS$ )
```

---

Fig. 2. Multiobjective heuristic search procedure.

light area, always respecting the problem constraints. The feasibility of the solutions is maintained for every load connected by using a *backward-forward sweep* power flow method [4]. The algorithm illustrated in Figure 2 shows how the constructive and improvement phases are managed in the search process.

The algorithm considers three parameters, as well as the pre-fault configuration( $G_{op}$ ): the first two parameters correspond to the maximum number of iterations of the constructive phase ( $IT_C$ ) and of the local search phase ( $IT_L$ ), respectively; the third ( $N_{sol}$ ) refers to the number of solutions for which the search will be conducted in parallel in the local search phase.

The first two steps refer to the phases in the solution of the problem, both requiring a specific solution representation. Step 1 involves the creation of initial solutions, whereas the Step 2 involves their improvement. These two steps will be discussed in the following sections, as well as the solution representation adopted. Finally, in Step 3, the Pareto set  $PS$  is returned as the result.

#### A. Solution representation

One of the most common ways to represent a distribution system configuration is through its switch states. A vector containing the status of all switches can be used to denote such a configuration:  $[x_1, x_2, \dots, x_n]$ , where  $x_i = 1$  if the switch  $i$  is closed or  $x_i = 0$  if it is open. This approach has been adopted by two well-known papers, those of Morelato and Monticelli [15] and Hsu and Kuo [9]. One disadvantage of this representation is the difficulty of preserving a feasible configuration when a search space of  $2^n$  combinations is considered. Even approaches such as that proposed by Aoki et al. [2], which try to reduce this number of combinations by adopting heuristic techniques to manipulate the switches, may have trouble in overcoming large outage areas.

In this paper we propose a forest graph [1] to represent the radial distribution system. Such a representation is obtained from consideration of the system switches as graph arcs and of the system buses as graph nodes. However, neither the constructive heuristic nor the local search algorithm consider the

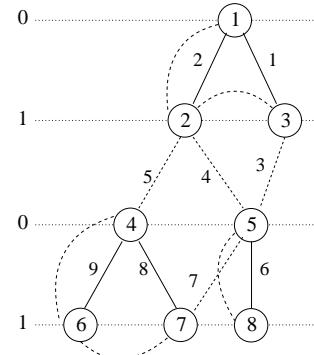


Fig. 3. Sample Network.

TABLE I  
REPRESENTATION FOR THE SAMPLE NETWORK.

	Nodes							
	1	2	3	4	5	6	7	8
$N(i)$	2	3	0	6	8	7	0	0
$P(i)$	0	1	1	0	0	4	4	5
$D(i)$	0	1	1	0	0	1	1	1
$T(i)$	1	1	1	2	3	2	2	3
	Trees							
	1	2	3					
$R(a)$	1	4	5					

search space of the nodes, but rather that of sectors, in which configuration cannot change because there are no switches. Such an approach can reduce the processing time, since the search space of sectors is smaller than that of nodes.

Figure 3 and Table I show a sample forest graph network and its correspondent representation. The system configuration is managed by five lists: List  $N(i)$  providing the next node for node  $i$  when the graph is traversed in preorder [1]; List  $P(i)$  indicating the predecessor node of node  $i$ ; List  $D(i)$  furnishing the depth of node  $i$ ; and List  $T(i)$  determining the tree to which node  $i$  belongs. The fifth list,  $R(a)$ , is needed to indicate the root of the trees, with the number 1 conventionally referring to the light area.

In order to reduce computational time, we have adopted a special structure for storing the sectors of the distribution system and for facilitating the manipulation of these sectors. The algorithms of the method make use of this structure especially when a sector is included in a single tree. Since the representation stores all the nodes of the graph, it is advantageous to maintain a structure that helps add all the nodes of a single sector to a given tree, thus preventing unnecessary search across those arcs which are outside the sector. Such a structure is shown in Figure 4 for the sample network in Figure 3.

The list  $S(i)$  attempts to store the relation between nodes and sectors, in such a way that it is possible to identify the sector in which a node is located at the time  $O(1)$  [11]. Given the sector of a node, the adjacency list of this sector can also be retrieved for this same time  $O(1)$ . With this information in hand, the next step is to determine the node in which the sector tree will be rooted, thus making it possible to traverse the adjacency list of the sector.

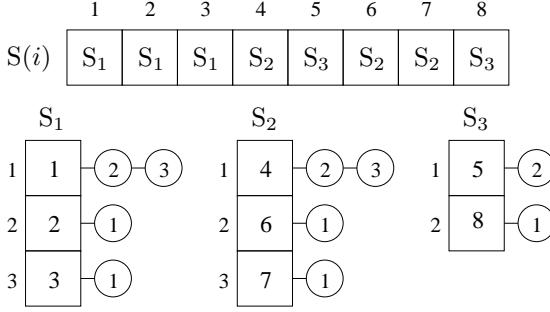


Fig. 4. Data structure for storage of sectors of the distribution system.

Given the representation and the data structure adopted, it is possible to define the algorithms that use them. The next two sections describe the constructive and local search algorithms used by the procedure developed.

### B. Constructive phase

The constructive phase is carried out by a random version of the well-known *Prim* algorithm [1]. Two versions were originally developed, one for minimizing the load not supplied and the other for minimizing the number of switching operations, with the first consisting of a breadth-first search and the latter a depth-first search. These two search algorithms attempt to produce solutions topologically different from each other.

The constructive algorithm is shown in Figure 5. Two parameters are included:  $IT_C$  corresponding to the number of iterations for which the algorithm is repeated; and  $G_{op}$  being the network graph after the system fault has been isolated. The first two steps refer to the initialization of the set  $PS$  (Step 1) and of the iteration count  $it$  (Step 2). The loop between Steps 3 and 23 includes the construction of the source node list (Step 4) and the execution of two other loops: one for minimization of the load not supplied (Steps 6-13) and the other for minimization of the number of switching operations involved (Steps 15-22). Before each one, the solution  $G_{sol}$  is set to the same configuration of  $G_{op}$ , Steps 5 and 14.

### Constructive( $IT_C, G_{op}$ )

```

1.  $PS = \emptyset;$ 
2.  $it = 0;$ 
3. while( $it < IT_C$ ) do
4.    $SNL_1 = SNL_2 = \text{CreateSourceNodes}(G_{op});$ 
5.    $G_{sol} = G_{op};$ 
6.   while( $\text{size}(SNL_1) \neq 0$ ) do
7.      $ADJ = \text{AdjcentNodes}(SNL_1[0], G_{sol});$ 
8.     if( $\text{size}(ADJ) \neq 0$ ) do
9.        $i = \text{LoadBasedRandom}(ADJ, G_{sol});$ 
10.       $\text{AddToEnd}(i, SNL_1);$ 
11.      if  $\text{Include}((SNL_1[0], i), G_{sol})$  then
12.         $\text{Update}(PS, G_{sol});$ 
13.      else  $\text{RemoveFirst}(SNL_1);$ 
14.    $G_{sol} = G_{op};$ 
15.   while( $\text{size}(SNL_2) \neq 0$ ) do
16.      $ADJ = \text{AdjcentNodes}(SNL_2[0], G_{sol});$ 
17.     if( $\text{size}(ADJ) \neq 0$ ) do
18.        $i = \text{SwitchBasedRandom}(ADJ, G_{sol});$ 
19.        $\text{AddToBeginning}(i, SNL_2);$ 
20.       if  $\text{Include}((SNL_2[0], i), G_{sol})$  then
21.          $\text{Update}(PS, G_{sol});$ 
22.       else  $\text{RemoveFirst}(SNL_2);$ 
23.      $it = it + 1;$ 
24. return( $PS$ );

```

Fig. 5. Constructive algorithm developed.

The breadth-first search conducted in the loop between Steps 6 and 13 attempts to minimize the load not supplied. First a list of adjacent nodes of the first node in the  $SNL_1$  list is generated (Step 7). If this list is empty means that the current source node has no successor nodes to connect with it and then this node is removed from the search list  $SNL_1$ . Otherwise, if the list  $ADJ$  is not empty, one node  $i$  is chosen according to a roulette wheel selection function weighted by node loads (Step 9). Next, this node  $i$  is included in end of list  $SNL_1$  (Step 10). At this point the arc to be included in the solution graph  $G_{sol}$  is already defined: between nodes  $SNL_1[0]$  and  $i$ . In Step 11 this inclusion is carried out but it is only completed if the solution  $G_{sol}$  remains feasible after that. The approximated Pareto set  $PS$  is only updated (Step 12) if this inclusion has occurred. This loop will be repeat until the search list  $SNL_1$  is empty. The depth-first search is conducted as the same manner as the breadth-first search (Steps 15-22), except by Steps 18 and 19: the adjacent node  $i$  is chosen on the basis of a roulette wheel selection function weighted by switch status in Step 18; in Step 19 this node  $i$  is included in the beginning of the list. The algorithm finishes when the maximum number of iterations ( $IT_C$ ) has been reached.

The constructive algorithms can be understood better by considering Figure 3. The black area in-

cludes nodes 4-8, while nodes 2 and 3 are the source nodes. Suppose that all arcs are switches and that node of the switches associated with arcs 3, 4 and 5 can support the power load for the entire black area (nodes 4-8), and further that  $L(4) > L(6) > L(7) > L(5) > L(8)$ , where  $L(i)$  corresponds to the load of node  $i$ . Therefore, the only way to restore nodes 4-8 is to connect some of the nodes to the source node 2 and the others to node 3. When executing the constructive algorithm for minimizing the load not supplied, suppose that the source node 2 is the first chosen. Its adjacent nodes are nodes 4 and 5. Since  $L(4) > L(5)$ , node 4 is first included in the light area and its adjacent nodes will only be considered after the evaluation of node 5, as shown in Figure 6(a). Given the power flow limits on the arcs 4 and 5, the final configuration will involve the inclusion of nodes 4, 5 and 6 in the light area, by closing switches 4, 5 and 9. Figures 6(b) e 6(c) shows the inclusion of nodes 5 and 6.

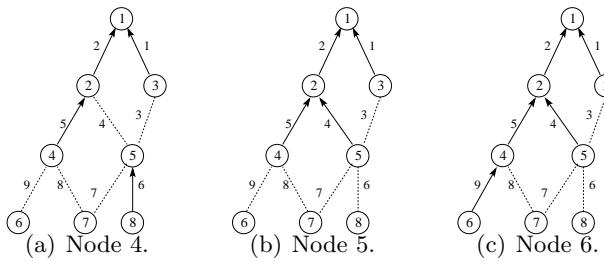


Fig. 6. Inclusion of nodes to the light area following the constructive algorithm for minimizing the load not supplied.

The second algorithm follows the same basic procedure, except that it uses a depth-first search and selects the nodes according to the number of switching operations involved when current status is compared to the pre-fault configuration. Making the same assumptions as in the previous example and further switches 6, 8 and 9 were closed in the pre-fault configuration, the network configurations reached after executing the second constructive algorithm are shown in Figures 7(a)-7(e). The construction begins with the choice of node 3 as the first source node. From that it is possible to connect only node 5 (Figure 7(a)). After, node 8 is chosen due the switch status in arc 6 (Figure 7(b)). Since there are no successors from node 8, node 7 is then considered (Figure 7(c)). Remembering the limits in arcs 3, 4 and 5, no further inclusions are feasible. Therefore, source node 2 is chosen and nodes 4 and 6 are included one at a time as shown in Figures 7(d) and 7(e).

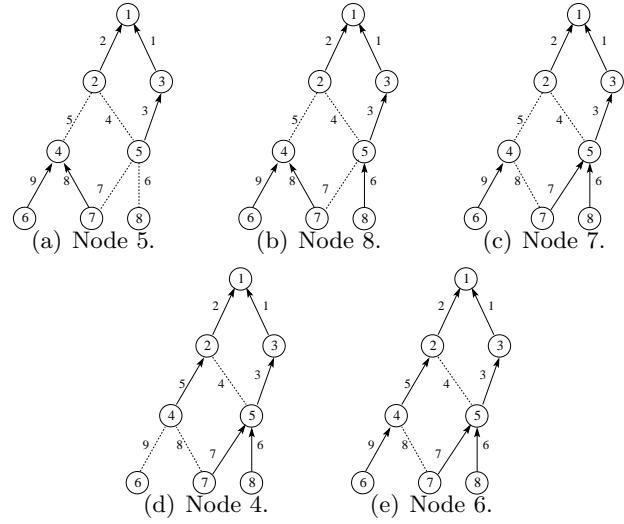


Fig. 7. Inclusion of nodes to the light area following the constructive algorithm for minimizing the number of switching operations involved.

### C. Local search phase

The local search phase tries to improve on the initial solutions by using a multiobjective search procedure, which generates neighbor solutions by changing the source node for each node or sector in the black area. Moreover, after defining the node or sector and the new source node is needed to choose the path between them. In order to find well distributed solutions in the objective space, a subset of them is explored in parallel, employing Pareto dominance as the optimization criterion.

For a better understanding of the procedures involved in this phase, consider Figure 3 and arcs 3, 4, 5 and 7 as the only switches in the network. With this assumptions three sectors are obtained, one corresponding to the light area and the other two to the black area: sector 1 (light area) including nodes 1-3; sector 2 (black area) including nodes 4, 6 and 7; and sector 3 (black area) including nodes 5 and 8. Given a solution with all switches open, the corresponding neighbor solutions are presented in Table II. Each line represents a solution (first column) generated by changing the source node (third column) of a sector (second column) through a path (fourth column) between them.

Figure 8 explains how the multiobjective local search is conducted. The parameters include the maximum number of iterations ( $IT_L$ ), the number of solutions to be explored in parallel ( $N_{sol}$ ), and the approximate Pareto set ( $PS$ ). Two other sets are also used during the procedure : the first containing the non-dominated solutions generated during the

TABLE II  
NEIGHBOR SOLUTIONS FOR THE NETWORK GIVEN IN  
FIGURE 3.

Solution	Sector	Source Node	Path
1	2	2	4-2
2	2	2	7-5-2
3	2	3	7-5-3
4	3	2	5-2
5	3	2	5-7-4-2
6	3	3	5-3

current iteration ( $PS_i$ ); and the second containing the representative solutions chosen for exploration ( $PS_r$ ). The loop between Steps 2 and 15 is repeated for all the iterations ( $IT_L$ ). Steps 3 and 4 attempt to reduce the  $PS$  by using a *clustering* procedure [16], which extracts the  $N_{sol}$  most representative solutions from  $PS$  and stores them in  $PS_r$ , while the remaining solutions stay in  $PS$ . For each solution of the set  $PS_r$  a neighborhood is generated by changing the source node for each black area node or sector (Steps 6-13), also considering all the paths between the given node or sector and the reachable source nodes. Each iteration of the loop of Step 8 includes the initialization of solution  $s$ , the disconnection of the node or sector from the current source node (Step 10) and further its connection to the new source node obtained from the given path  $c_k$  (Step 11). In case of feasibility of solution  $s$ , it is used to update the set of non-dominated solutions of the iteration  $i$  (Steps 12 and 13). After the loop between Steps 6-13, the  $PS_i$  set is used to update the main set of non-dominated solutions  $PS$  (Step 14). This updated  $PS$  set is then used for the next iteration for the selection of further representative solution. Finally the iteration counter is incremented in Step 15 and the  $PS$  set is returned as the result in Step 16.

For example, consider Figure 9(a) and suppose that node 4 is chosen by the local search to be connected to the source node 3 through the path 4-8-7-5-3. Also consider that arcs 3, 4, 5 and 7 are the only switches in the network. First, node 4 is disconnected from source node 2, by opening switch 5, as shown in Figure 9(b). Since node 4 is included in the sector with nodes 6 and 7, all of them should be connected with node 4 to the source node 3. Figure 9(c) shows the network configuration after this connection. Note that nodes 5 and 8 are also included into the light area because they are on the path between node 4 and the source node 3. The same procedure will be conducted to the other sector in the

---

```

LocalSearch( $IT_L, N_{sol}, PS$ )
1.  $it = 0;$ 
2. while( $it < IT_L$ ) do
3.   if( $|PS| > N_{sol}$ ) then
4.      $PS_r = \text{Reduce}(PS, N_{sol});$ 
5.      $PS_i = \emptyset;$ 
6.   for each  $s_i \in PS_r$  do
7.     for each sector  $t_j$  in the black area of  $s_i$  do
8.       for each path  $c_k$  between  $t_j$  and a source node do
9.          $s = s_i;$ 
10.        Disconnect( $t_j, \text{current\_source\_node}, s$ );
11.        Connect( $t_j, c_k, s$ );
12.        if  $s$  is feasible then
13.          Update( $s, PS_i$ );
14.        Update( $PS_i, PS$ );
15.       $it = it + 1;$ 
16.   return( $PS$ );

```

---

Fig. 8. Multiobjective local search procedure.

black area.

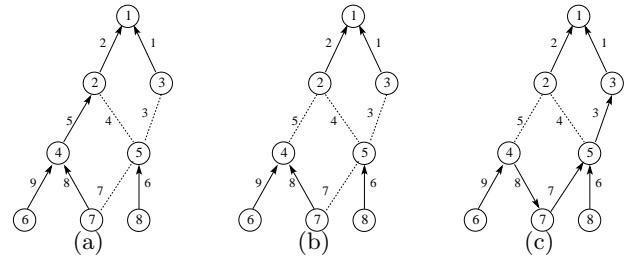


Fig. 9. Example of changing of source node in the local search procedure.

#### IV. COMPUTATIONAL RESULTS

The proposed multiobjective local search based heuristic method was applied to three distribution systems, as shown in Table III. The former was obtained from the paper of Baran and Wu [4]; the second was presented by [3], and the latter corresponds to an actual Brazilian distribution system. The programming language used was C++, on a Pentium 4 PC with 2.8 GHz running Linux 2.6.

In order to evaluate the performance of the proposed method, it was developed a randomly exhaustive enumeration which generates arrangements at random, considering the nodes to connect to the light area as well as their respectively paths up to the source nodes. After several hours of cpu time, it was obtained one front for each system considered.

The proposed method has four parameters: the number of iterations for the constructive phase; the number of iterations for the local search phase; the number of neighborhoods to generate at each iteration of the local search; and the pre-fault configuration. For the first one it was established a value of

TABLE III  
DISTRIBUTION SYSTEMS CONSIDERED.

System	# substations	# nodes	# sectors	# arcs	# switches	total load (kW)	black area load (kW)
1	1	34	1	40	40	3,715	1,490
2	5	90	1	112	113	28,612	3,814
3	1	1,057	42	1,078	63	11,542	3,343

$n^2/2$ , where  $n$  is the number of switches in the black area. For the second and third ones were generated a certain number of combinations in order to evaluate the influence they have in the final result. As the last parameter corresponds to the SRP instance, it was taken one for each system described in Table III.

Table IV shows the results after executing the proposed method in three blocks, one for each system. The first line of each block contains the initial front obtained after executing the constructive phase. The column information is as follows: the first contains the system; the second contains the number of iterations for the local search phase; the third contains the number of neighborhoods generated at each iteration of the local search; the fourth containing the computational time, in seconds, for executing the method; and last containing the final front obtained after executing the proposed heuristic. The solution values are present in pairs  $(x; y)$ , where  $x$  refers to the load not supplied and  $y$  to the number of switching operations involved.

When considering the results for the first two systems, it can be noted that there was not difference between the fronts obtained after executing the method with each combination of parameter values. However, in both cases the contribution of the second phase is clear, since it tries to approximate the front obtained to the optimal Pareto front.

The results for the system 3 presented in Table IV show a clear influence of both parameter values in the quality of the final Pareto front. In general, the proposed heuristic obtained the best results for the large number of iterations and the large number of neighborhoods explored at a time.

Figures 10-12 summarize the results of Table IV when considering two combinations of parameter values for each system: 150 iterations ( $It$ ) and 2 neighborhoods generated at a time ( $N_{ghd}$ ) and 150 iterations and 4 neighborhoods generated at a time. The optimal front obtained by randomly exhaustive enumeration is also included. For all systems the proposed method was able to reach near optimal fronts, including solutions that were not obtained by the randomly exhaustive enumeration, for instance

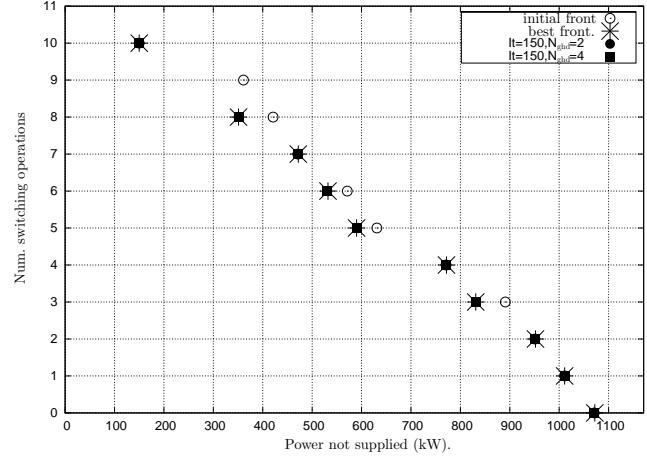


Fig. 10. Fronts obtained for system 1.

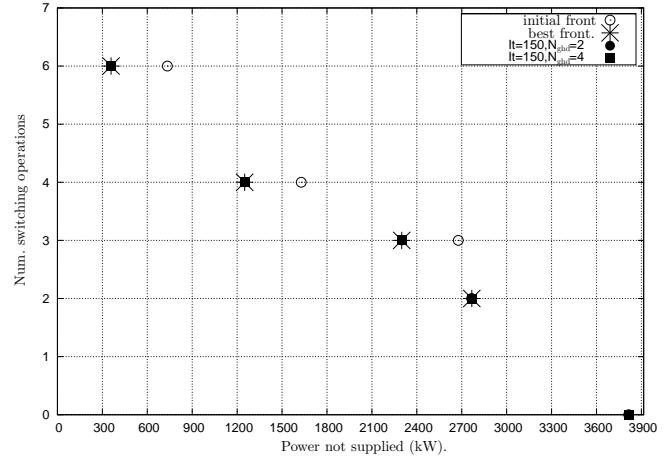


Fig. 11. Fronts obtained for system 2.

solutions  $(1368; 3)$ ,  $(518; 5)$  and  $(0; 6)$  for system 3. After all, for all systems, the non-dominated solutions were relatively well distributed over the front obtained and the computational time required was minimal.

## V. CONCLUSIONS

This paper has presented a multiobjective local search based heuristic method to solve the Service Restoration Problem (SRP) in electrical distribution systems. The mathematical formulation presented considers the minimization of the load not supplied

TABLE IV  
TEST RESULTS.

Initial front			(1070;0) (1010;1) (950;2) (890;3) (770;4) (630;5) (570;6) (470;7) (420;8) (360;9) (150;10)
# iterations	# neigh.	time(s)	Front obtained
System 1	50	2	(1070;0) (1010;1) (950;2) (830;3) (770;4) (590;5) (530;6) (470;7) (350;8) (150;10)
	100	2	(1070;0) (1010;1) (950;2) (830;3) (770;4) (590;5) (530;6) (470;7) (350;8) (150;10)
	150	2	(1070;0) (110;1) (950;2) (830;3) (770;4) (590;5) (530;6) (470;7) (350;8) (150;10)
	50	4	(1070;0) (1010;1) (950;2) (830;3) (770;4) (590;5) (530;6) (470;7) (350;8) (150;10)
	100	4	(1070;0) (1010;1) (950;2) (830;3) (770;4) (590;5) (530;6) (470;7) (350;8) (150;10)
	150	4	(1070;0) (1010;1) (950;2) (830;3) (770;4) (590;5) (530;6) (470;7) (350;8) (150;10)
Initial front			(3814;0) (2765;2) (2676;3) (1627;4) (734;6)
# iterations	# neigh.	time(s)	Front obtained
System 2	50	2	(3814;0) (2765;2) (2299;3) (1250;4) (357;6)
	100	2	(3814;0) (2765;2) (2299;3) (1250;4) (357;6)
	150	2	(3814;0) (2765;2) (2299;3) (1250;4) (357;6)
	50	4	(3814;0) (2765;2) (2299;3) (1250;4) (357;6)
	100	4	(3814;0) (2765;2) (2299;3) (1250;4) (357;6)
	150	4	(3814;0) (2765;2) (2299;3) (1250;4) (357;6)
Initial front			(3343;0) (2825;2) (2222;3) (1704;4) (1431;6) (1327;7) (809;8) (728;9) (0;10)
# iterations	# neigh.	time(s)	Front obtained
System 3	50	2	(3343;0) (2825;2) (2222;3) (1704;4) (1431;5) (1169;6) (1042;7) (543;8) (281;9) (0;10)
	100	2	(3343;0) (2374;2) (2222;3) (1704;4) (1253;5) (735;6) (462;7) (200;8) (0;10)
	150	2	(3343;0) (2374;2) (2222;3) (1704;4) (1253;5) (735;6) (462;7) (200;8) (0;10)
	50	4	(3343;0) (2825;2) (2222;3) (1704;4) (976;5) (587;7) (88;8) (0;10)
	100	4	(3343;0) (2367;1) (1849;3) (1246;4) (728;5) (711;6) (193;7) (0;8)
	150	4	(3343;0) (2344;2) (1368;3) (518;5) (0;6)

and of the number of switching operations, while respecting voltage, current and feeder capacity constraints.

The methodology has proven suitable for the SRP, due to the interaction of constructive and local search algorithms in the selection of source nodes. Especially related to the data structures and neighborhood used, they indeed contribute to maintain the computational time in the minimal standards shown as well as improve the ability for exploring the search space.

Moreover, the method has proved its flexibility in reaching to a variety of possible well distributed solutions throughout the Pareto front, while requiring

minimal computational time.

Further studies should investigate the inclusion of more objective functions and more tests with a large variety of distribution networks.

#### ACKNOWLEDGEMENTS

The authors would like to thank to the Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP) and to the Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) for the financial support provided.

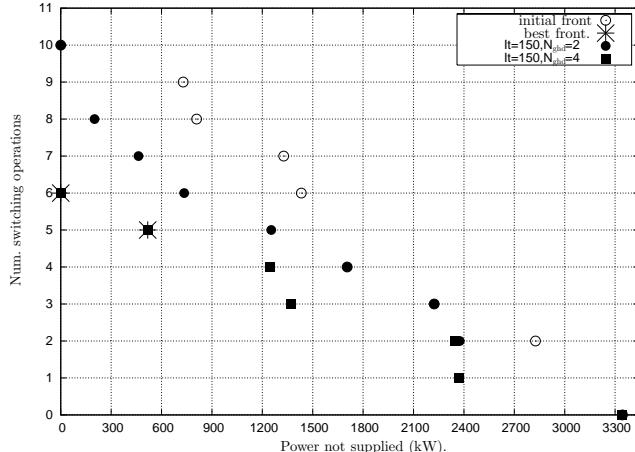


Fig. 12. Fronts obtained for system 3.

## REFERENCES

- [1] R. Ahuja, T. Magnanti, and J. Orlin, *Network Flows: Theory, Algorithms and Applications*. Prentice Hall, Englewood Cliffs, 1993.
- [2] K. Aoki, K. Nara, M. Itoh, T. Satoh, and H. Kuwabara, "A new algorithm for service restoration in distribution systems," *IEEE Transactions on Power Delivery*, vol. 4, no. 3, pp. 1832–1839, 1989.
- [3] A. Augugliaro, L. Dusonchet, and E. R. Sanseverino, "Evolving non-dominated solutions in multiobjective service restoration for automated distribution networks," *Electric Power Systems Research*, vol. 59, pp. 185–195, 2001.
- [4] E. Baran and F. Wu, "Network reconfiguration in distribution systems for loss reduction and load balancing," *IEEE Transactions on Power Delivery*, vol. 4, no. 2, pp. 1401–1407, April 1989.
- [5] R. Ceric and D. Popovic, "Multi-objective distribution network restoration using heuristic approach and mix integer programming method," *Electrical Power and Energy Systems*, vol. 22, pp. 497–505, 2000.
- [6] S. Curcic, C. Ozveren, L. Crowe, and P. Lo, "Electric power distribution network restoration: a survey of papers and a review of the restoration problem," *Electric Power Systems Research*, vol. 35, pp. 73–86, 1996.
- [7] M. Ehrgott and X. Gandibleux, "Approximative solution methods for multiobjective combinatorial optimization," *TOP (Spanish journal of operations research)*, vol. 12, no. 1, pp. 1–90, June 2004.
- [8] M. Hansen, "Metaheuristics for multiple objective combinatorial optimization," Report, Technical University of Denmark, 1998.
- [9] Y. Hsu and H. Kuo, "A heuristic based fuzzy reasoning approach for distribution system service restoration," *IEEE Transactions on Power Delivery*, vol. 9, no. 2, pp. 948–953, 1994.
- [10] S.-J. Lee, S.-I. Lim, and B.-S. Ahn, "Service restoration of primary distribution systems based on fuzzy evaluation of multi-criteria," *IEEE Transactions on Power Systems*, vol. 13, no. 3, pp. 1156–1163, August 1998.
- [11] H. Lewis and C. Papadimitriou, *Elements of the Theory of Computation*. Prentice-Hall International, 1981.
- [12] C. Liu, S. Lee, and S. Venkata, "An expert system operational aid for restoration and loss reduction of distribution systems," *IEEE Transactions on Power Delivery*, vol. 3, no. 2, pp. 619–626, 1988.
- [13] M. Matos and P. Melo, "Multiobjective reconfiguration for loss reduction and service restoration using simulated annealing," in *Proceedings of IEEE Budapest Power Tech'99*. IEEE Service Center, 1999.
- [14] K. Miu, H.-D. Chiand, B. Yuan, and G. Darling, "Fast service restoration for large-scale distribution systems with priority customers and constraints," *IEEE Transactions on Power Delivery*, vol. 13, no. 3, pp. 789–795, 1998.
- [15] A. Morelato and A. Monticelli, "Heuristic search approach to distribution system restoration," *IEEE Transactions on Power Delivery*, vol. 4, no. 4, pp. 2235–2241, October 1989.
- [16] J. N. Morse, "Reducing the size of the nondominated set: Pruning by clustering," *Computers and Operations Research*, vol. 7, no. 1–2, pp. 55–66, 1980.
- [17] L. Murphy and F. Wu, "A comprehensive analysis of distribution automation systems," University of California, Berkeley, Tech. Rep. M90/72, 1990.
- [18] D. Shirmohammadi, "Service restoration in distribution networks via network reconfiguration," *IEEE Transactions on Power Delivery*, vol. 7, no. 2, pp. 952–958, April 1992.
- [19] R. Steuer, *Multiple Criteria Optimization: Theory, Computation and Application*. New York: Wiley, 1986.
- [20] S. Toune, H. Fudo, T. Genji, Y. Fukuyama, and Y. Nakaniishi, "Comparative study of modern heuristic algorithms to service restoration in distribution systems," *IEEE Transactions on Power Delivery*, vol. 17, no. 1, pp. 173–181, January 2002.

# The Traveling Salesman Problem with Time-Dependent Costs: an exact approach

José Albiach\*, †, José María Sanchis\* and David Soler\*

\*Universidad Politécnica de Valencia/Departamento de Matemática Aplicada  
Camino de Vera s/n, 46022 Valencia (Spain)

Emails: jalbiach@mat.upv.es, jmsanchis@mat.upv.es, dsoler@mat.upv.es

† Corresponding author

**Abstract**— In this paper we deal with a problem which generalizes the Traveling Salesman Problem with Time Windows (TSPTW). The generalization consists of the time-dependence of the travel times and costs, for a more accurate fitting of some routing problems inside large cities, where the time or cost of traversing some streets (e.g. main avenues) depend on the moment of the day (for example rush-hours).

In contrast to other existing papers about routing problems with time-dependent costs, we focus on an exact approach to this new problem. To do this we transform it in pseudo-polynomial time into a classical Asymmetric Traveling Salesman Problem for which several exact and heuristic algorithms exist, even for large-scale instances. Computational results are presented on a set of 270 adapted instances from benchmark TSPTW instances.

**Keywords**— Traveling Salesman Problem, time window, time-dependence.

## I. INTRODUCTION

THE Traveling Salesman Problem with Time Windows (TSPTW) is a well-known optimization routing problem that can be defined as follows:

*Given a directed graph  $G = (V, A)$  with nonnegative costs associated with its arcs, such that each vertex  $i$  has associated a time window  $[a_i, b_i]$ , one of the vertices (say  $i_0$ ) is considered as a depot, and traversing arc  $(i, j) \in A$  implies a travel time  $t_{ij} > 0$ , find a minimum cost circuit in  $G$  starting in  $i_0$  at time  $a_{i_0}$  and passing through each vertex exactly once, such that the circuit leaves each vertex in its associated time window and ends in  $i_0$  before  $b_{i_0}$ . Note that it is allowed to arrive at vertex  $i$  before  $a_i$  (waiting time), but in this case, the circuit will leave  $i$  at time  $a_i$ . For simplicity, if a service time is necessary at one vertex,  $i$  this time will be included in the travel times  $t_{ji}$   $j \neq i$ .*

The TSPTW has important applications, especially in sequencing and distribution problems. For this reason many papers have studied this topic in the last decade. See for example [1], [2], [4], [12], [16], [18] and [23].

Like in most routing problems found in the OR literature, in the TSPTW the arc costs or times are considered constant throughout the day. This assumption may result in a weak approximation to real-world conditions, at least in distribution problems inside large cities, where the time or cost of traversing some streets (e.g. main avenues) depend on the moment of the day (for example at rush-hours).

Despite the occurrence of traffic jams in large cities at certain times and in certain areas, routing problems with time-dependent costs have hardly been studied because they are more difficult to model and to solve; however more and more, works on optimization vehicle routing problems are taking into account time-dependent travel costs for a more accurate approximation of the mathematical models to the real problems. The work in [17] includes a detailed review of the literature on time-dependent routing problems, and as recent papers we may cite [15] and [24]. These works are based on heuristic procedures to solve multivehicle routing problems with different kinds of time-dependent travel times.

With respect to single vehicle routing problems including time-dependent travel times, we must cite the works [20] and [21]. These works study the Time-Dependent Traveling Salesman Problem (TDTSP), a problem similar to the one presented here, but with considerable differences which will be commented following the definition of our problem. These authors focused on heuristic procedures for the TDTSP without time windows, tested on instances with up to 55 vertices. In none of these instances the optimality of the solution was proved.

In this paper we present a generalization of the TSPTW in which, in addition to time windows, the cost and the travel time of each arc depend on the moment at which we start traversing it. Because of this, waiting times are allowed in a more general way than in the TSPTW: for total travel cost minimization, if the vehicle arrives at a vertex  $i$  at time  $t_i^1 < b_i$ , it can wait and

leave vertex  $i$  at time  $t_i^2$  with  $\max\{a_i, t_i^1\} \leq t_i^2 \leq b_i$ . Moreover, due also to the time-dependence of costs and travel times, the circuit can start at the depot node  $i_0$  at time  $t_{i_0} > a_{i_0}$ . For example, if time  $a_{i_0}$  belongs to a rush-hour, instead of starting the route at  $a_{i_0}$ , if possible, we can be working inside the warehouse for a short period of time until the traffic be moving quite freely.

The main difference of this work with respect to the above mentioned papers is that we focus on an exact approach to the problem; through a graph transformation we find a way to solve the new problem by transforming it into the classical Asymmetric Traveling Salesman Problem (ATSP), for which several heuristic and exact procedures have been successfully tested, even for large-scale instances with several thousands of vertices (see for example [3], [5], [6] and [19]).

To transform the new problem into an ATSP, we first transform it into another combinatorial optimization problem studied in the OR literature, the Asymmetric Generalized Traveling Salesman Problem (AGTSP). The AGTSP consists of:

*Given a directed graph  $G = (V, A)$  with nonnegative costs associated with its arcs, such that  $V$  is partitioned into  $k$  nonempty subsets  $\{S_i\}_{i=1}^k$ , find a minimum cost circuit passing through exactly one vertex of each subset  $S_i \forall i \in \{1, \dots, k\}$ .*

To solve the AGTSP several polynomial time transformations of this problem into an ATSP have been described in the literature. The most efficient seems to be the transformation defined in [22]. As we will use this transformation, let us describe it briefly:

*From  $G$  construct a new directed graph with the same vertex set but order the vertices of each  $S_i$  consecutively in an arbitrary way  $\{v_1^i, \dots, v_{l(i)}^i\}$   $l(i)$  being the number of vertices in  $S_i$ . For  $j = 1, \dots, l(i) - 1$  define the cost  $c_{j,j+1}^i$  of arc  $(v_j^i, v_{j+1}^i)$  as zero, define  $c_{l(i),1}^i$  as zero and for every  $v_j^i \in S_i$  and every  $w \notin S_i$  set  $c_{v_j^i,w}$  equal to the cost in  $G$  of the arc from  $v_{j+1(\text{mod } l(i))}^i$  to  $w$  plus a fixed positive large quantity  $M$  if  $|S_i| > 1$  and equal to the cost in  $G$  of the arc from  $v_j^i$  to  $w$  plus  $M$  if  $|S_i| = 1$ , any other arc has infinite cost.*

Solving the AGTSP in  $G$  is equivalent to solving the ATSP in the new digraph.

Finally, as our aim is an optimal approach to the new problem, we will use the exact algorithm for the Mixed General Routing Problem (MGRP) described in [9] to solve the resulting ATSP instances. The MGRP basically consists of finding a minimal closed walk on the edges and arcs (links) of a mixed graph  $G$  ( $G$  has simultaneously edges and arcs) which traverses a given subset of “required” links and a given subset of “re-

quired” vertices. This problem contains a large number of important arc and node routing problems as particular cases. For example, if  $G$  is a directed graph, the subset of required arcs is empty and all the vertices are required; in such case we will be dealing with the Graphical Asymmetric Traveling Salesman Problem (GATSP) (see [7]). This last problem, introduced in [13], [14] and [10] in the undirected version, is a generalization of the pure ATSP in which graph  $G$  does not need to be complete (few variables are needed) and then, the solution does not need to be a Hamiltonian cycle (i.e. passing through each vertex exactly once) but a closed walk passing through each vertex at least once. Note that an ATSP instance can be transformed into a GATSP instance by simply adding a large positive number  $L$  to each arc cost in order to assure the occurrence of a Hamiltonian cycle in the optimal solution. Therefore, as the GATSP is a particular case of the MGRP, the exact algorithm in [9] can be used to optimally solve the ATSP, even for large-scale ATSP instances, as we will see next. It is a cutting-plane algorithm based on the polyhedral study of the MGRP presented in [8] and [9] in which branch-and-bound is invoked when violated inequalities are not found.

The rest of the paper is organized as follows. In Section 2 we define the new problem, which we have called the Traveling Salesman Problem with Time Dependent Costs (TSPTDC), and we show the construction of an auxiliary digraph from a TSPTDC instance. In Section 3 we prove that the TSPTDC can be transformed in pseudo-polynomial time into an AGTSP on the auxiliary digraph. The size of the auxiliary digraph is then considerably reduced in order to make this transformation more competitive. In Section 4 we present computational results on the exact resolution on a set of 270 TSPTDC instances obtained by modifying benchmark TSPTW instances (see for example [4], [12] and [18]). The results obtained show that the MGRP algorithm [9] is an efficient tool to solve optimally real TSPTDC instances. Finally, Section 5 presents some conclusions about this work.

## II. DEFINITION OF THE TSPTDC AND AUXILIARY DIGRAPH

We define the Traveling Salesman Problem with Time Dependent Costs (TSPTDC) in the following way:

*Let  $G = (V, A)$  be a simple directed graph,  $V = \{v_i\}_{i=0}^n$  being its set of vertices, where  $v_0$  is the depot vertex, each vertex  $v_i$  has associated a time window  $[a_i, b_i]$  verifying that  $a_i, b_i \in \mathbb{Z}^+ \cup \{0\}$  and  $[a_i, b_i] \subseteq [a_0, b_0] \forall i \in \{1, \dots, n\}$ . Every time window  $[a_i, b_i]$  has associated  $p_i = b_i - a_i + 1$  instants of time  $\{a_i + k - 1\}_{k=1}^{p_i}$ . For simplicity we will denote  $t_i^k =$*

$a_i + k - 1$  and therefore,  $t_i^k \in \mathbb{Z}^+ \cup \{0\}$ . Each instant of time  $t_i^k$  with  $i > 0$  has also associated a waiting time window  $[w_i^k, t_i^k]$ ,  $w_i^k \in (\mathbb{Z}^+ \cup \{0\}) \cap [a_0, t_i^k]$ .

On the other hand, the time and the cost of traversing an arc  $(v_i, v_j) \in A$  depend on the instant of time  $t_i^k$  ( $k \in \{1, \dots, p_i\}$ ) at which we start traversing it. Let denote by  $t_{i,j}^k \in \mathbb{Z}^+$  and  $c_{i,j}^k \geq 0$  the time and the cost respectively of traversing arc  $(v_i, v_j)$  starting at instant  $t_i^k$ . Moreover, a waiting time  $t \in \mathbb{Z}^+$  at vertex  $i$  has associated a cost  $cwt^i(t)$ .

Find a Hamiltonian circuit in  $G$ , starting and ending at  $v_0$  at integer instants of time inside  $[a_0, b_0]$  such that: the sum of the costs of traversing arcs and waiting times be minimum, the circuit leaves each vertex  $v_i \in V$   $i > 0$  inside its associated time window and if the circuit arrives to vertex  $v_i$  with  $i > 0$  in the time window  $[w_i^k, t_i^k]$ , it is allowed an integer waiting time in vertex  $v_i$  and to leave  $v_i$  at time  $t_i^k$ .

Some relevant aspects of this definition are:

- This definition allows the circuit to start after instant  $a_0$ . This is very important to minimize costs; for instance, if  $a_0$  belongs to a rush-hour, if possible, we can work for a short period of time inside the warehouse until the traffic be moving quite freely.

- This definition also allows a waiting time at each vertex  $v_i$  if due to the traffic conditions, it is preferable to wait in order to minimize the cost of the circuit. This waiting time has an associated cost which normally is given by a linear function.

- As usual in vehicle routing problems, we assume that the time of traversing an arc  $(v_i, v_j)$  with  $j > 0$  includes the service time at  $v_j$ .

- From a practical point of view, the fact that the travel times must be integer values does not involve a strong restriction with respect to the continuous case, because we can define an appropriate and as-small-as required unit of time for each instance.

- In contrast to other papers, this definition distinguishes between two magnitudes: the time-dependent travel time and the time-dependent cost (which could be equal), focusing on cost minimization. In the particular case of the TSPTDC in which  $t_{ij}^k = t_{ij}^s = c_{ij}^k = c_{ij}^s \forall k, s \in \{1, \dots, p_i\}$  and  $\forall (v_i, v_j) \in A$  with  $i \neq 0$ ,  $c_{0,j}^k = \infty \forall k > 1$  and  $\forall j > 0$  (the circuit must start at time  $a_0$ ), and  $\forall i > 0$   $[w_i^k, t_i^k] = [a_0, a_i]$  if  $k = 1$  and  $[w_i^k, t_i^k] = \{t_i^k\}$  if  $k > 1$ , we obtain a TSPTW. Thus, the TSPTDC is a NP-hard problem.

In comparison to the problem studied in [20] and [21], some important differences are: in the TDTSP the circuit must start from the depot at exactly instant  $a_0$ ; it can not arrive to any customer  $v_i$  before  $a_i$ ; and the objective function is the difference between the return time and

the starting time of the circuit solution. In addition, the heuristics used work only on TDTSP without time windows. On the other hand, the main difference is that we focus on an optimal approach to the problem. In fact, although the TDTSP is not exactly a particular case of the TSPTDC, it can also be optimally solved with the procedure presented here, except for some small differences in the construction of the auxiliary digraph given next.

Consider then a TSPTDC defined on a directed graph  $G = (V, A)$  with all the corresponding data. We construct a directed auxiliary graph  $G' = (V', A')$  in the following way:

- For each vertex  $v_i$  with  $i \in \{0, \dots, n\}$  and for each instant of time  $t_i^k$  for all  $k \in \{1, \dots, p_i\}$  create a vertex  $v_i^k$ .

- For each pair of vertices  $v_i^k, v_j^l \in V'$  with  $i \neq j$  and such that  $t_i^k + t_{ij}^l \in [w_j^l, t_j^l]$  if  $j \neq 0$  and  $t_i^k + t_{ij}^l = t_j^l$  if  $j = 0$ , add to  $G'$  an arc  $(v_i^k, v_j^l)$  with cost equal to  $c_{i,j}^k + cwt^j(t_j^l - (t_i^k + t_{ij}^l))$ . Note that  $w_j^l \leq t_i^k + t_{ij}^l < t_j^l$  implies a waiting time at vertex  $v_j \in G$  if the circuit takes arc  $(v_i, v_j)$  at time  $t_i^k$  and leaves  $v_j$  at time  $t_j^l$ .

- Divide  $\{1, \dots, p_0\}$  into four subsets  $I_1, I_2, I_3, I_4$  in the following way:

- 1)  $k \in I_1$  if  $v_0^k$  has only leaving arcs in  $G'$ . In this case, replace also name  $v_0^k$  by  $v_{0s}^k$  (starting vertex).

- 2)  $k \in I_2$  if  $v_0^k$  has only entering arcs in  $G'$ . In this case, replace also name  $v_0^k$  by  $v_{0e}^k$  (ending vertex).

- 3)  $k \in I_3$  if  $v_0^k$  has both entering and leaving arcs in  $G'$ . In this case, split  $v_0^k$  into two vertices  $v_{0s}^k$  and  $v_{0e}^k$  such that  $v_{0s}^k$  will be only incident with the leaving arcs from  $v_0^k$  in  $G'$  and  $v_{0e}^k$  will be only incident with the entering arcs to  $v_0^k$  in  $G'$ .

- 4)  $k \in I_4$  if  $v_0^k$  has neither entering arcs nor leaving arcs in  $G'$ . Then, delete  $v_0^k$  from  $G'$  for all  $k \in I_4$ .

- Add to  $G'$  a new vertex  $v_d$ , which will be the depot, with the next arcs, all of them with zero cost:

- For each  $k \in I_1 \cup I_3$ , an arc  $(v_d, v_{0s}^k)$ .

- For each  $k \in I_2 \cup I_3$ , an arc  $(v_{0e}^k, v_d)$ .

A TSPTDC example with  $n = 3$  illustrates the construction of this auxiliary digraph. In order to clearly show the whole transformed directed graph, in this example we will suppose that  $\forall i > 0$   $[w_i^k, t_i^k] = [a_0, a_i]$  if  $k = 1$  and  $[w_i^k, t_i^k] = \{t_i^k\}$  if  $k > 1$ , that is, a waiting time at  $v_i$  is only allowed if we arrive at  $v_i$  before time  $a_i$  and we leave  $v_i$  at time  $a_i$  (like in the TSPTW). We

will also suppose in this example that waiting times have zero cost. The time windows are given in Figure 1.

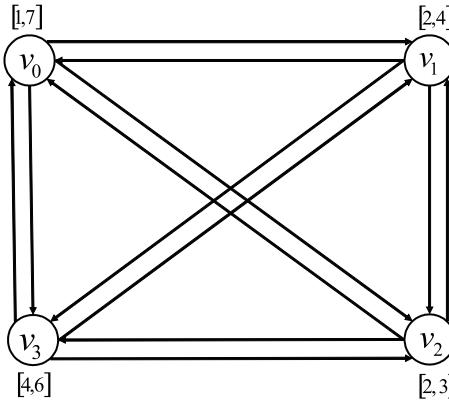


Fig. 1. Graph  $G$

Table I shows the time-dependent costs corresponding to this example. From this table we can easily obtain the travel times, depending on the time instant at which we start traversing the arcs because each  $t_i^k$  has in brackets its corresponding instant of time. For example, the element  $(t_0^1, t_1^2)$  means that if we traverse arc  $(v_0, v_1)$  starting at period of time  $t_0^1$ , which corresponds to instant 1,  $c_{0,1}^1 = 70$  and  $t_{0,1}^1 = 2$  because we arrive at  $v_1$  at  $t_1^2$ , which corresponds to instant 3. The element  $(t_0^1, t_2^1)$  means that if we traverse edge  $(v_0, v_2)$  starting at period of time  $t_0^1$  we arrive at  $v_2$  at most at time  $t_2^1 = a_2$  (then we may have a waiting time here). A dash inside a cell  $(t_i^k, t_j^l)$  means that if we traverse edge  $(v_i, v_j)$  starting at time  $t_i^k$  we will not arrive at  $v_j$  at time  $t_j^l$  if  $l > 1$  or that we will arrive after  $a_j$  if  $l = 1$ . Note that the table does not include the rows and columns with no possible paths.

Figure 2 shows the corresponding auxiliary digraph  $G'$  in which the vertices are denoted by numbers, indicating the order of the time instants in their corresponding time window; the vertices are clustered into subsets  $S_i$  corresponding to original vertices  $v_i$ . The arc costs have been omitted in this figure; they can be easily obtained from Table I and from the construction of  $G'$ .

### III. TRANSFORMATION OF THE TSPTDC INTO AN ATSP

Once the auxiliary digraph  $G'$  has been defined, we present a way to solve the TSPTDC by first transforming it into an AGTSP and then transforming the obtained AGTSP into an ATSP using the transformation in [22] that does not increase the size of the graph.

*Theorem 1:* The TSPTDC can be transformed in pseudo-polynomial time into an AGTSP defined in the auxiliary digraph.

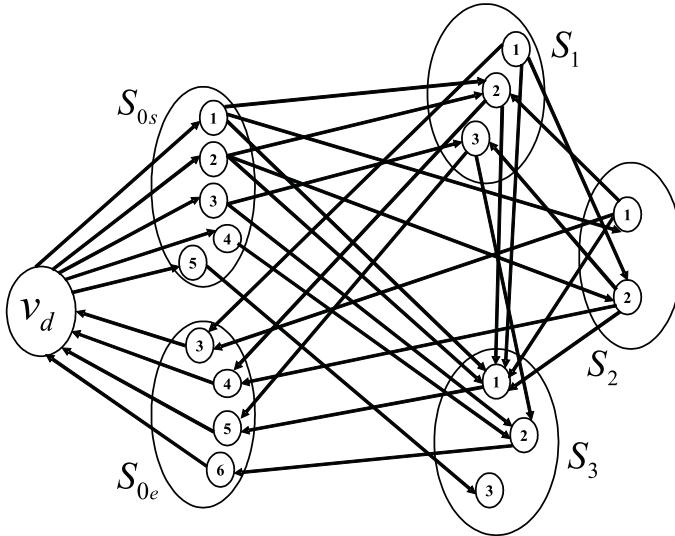
TABLE I  
TIME DEPENDENT COSTS OF GRAPH  $G$ .  $t_i^k(t)$  MEANS THAT INSTANT  $t_i^k$  IS EQUAL TO  $t$ .

	$t_1^2(3)$	$t_1^3(4)$	$t_2^1(2)$	$t_2^2(3)$	$t_3^1(4)$	$t_3^2(5)$	$t_3^3(6)$
$t_0^1(1)$	70	-	55	-	93	-	-
$t_0^2(2)$	55	-	-	39	66	-	-
$t_0^3(3)$	-	53	-	-	-	68	-
$t_0^4(4)$	-	-	-	-	-	40	-
$t_0^5(5)$	-	-	-	-	-	-	40
	$t_0^3(3)$	$t_0^4(4)$	$t_0^5(5)$	$t_2^2(3)$	$t_3^1(4)$	$t_3^2(5)$	
$t_1^1(2)$	43	-	-	40	63	-	
$t_1^2(3)$	-	40	-	-	31	-	
$t_1^3(4)$	-	-	31	-	-	36	
	$t_0^3(3)$	$t_0^4(4)$	$t_1^2(3)$	$t_1^3(4)$	$t_3^1(4)$		
$t_2^1(2)$	35	-	37	-	70		
$t_2^2(3)$	-	36	-	38	36		
	$t_0^5(5)$	$t_0^6(6)$					
$t_3^1(4)$	32	-					
$t_3^2(5)$	-	33					

*Proof:* Let  $G = (V, A)$  be the digraph where a TSPTDC is defined and let  $G' = (V', A')$  be its auxiliary digraph. Consider an AGTSP in  $G'$  corresponding to the partition of  $V'$  into the following subsets:  $S_d = \{v_d\}$ ,  $S_i = \{v_i^k\}_{k=1}^{p_i} \forall i \in \{1, \dots, n\}$ ,  $S_{0s} = \{v_{0s}^k\}_{k \in I_1 \cup I_3}$  and  $S_{0e} = \{v_{0e}^k\}_{k \in I_2 \cup I_3}$ , that is,  $n + 3$  subsets.

By construction of  $G'$  there is a one-to-one correspondence between the set of feasible AGTSP solutions in  $G'$  and the set of feasible TSPTDC solutions in  $G$ . It is enough to identify the circuit AGTSP solution in  $G'$   $T' = \{v_d, v_{0s}^{k_0}, v_{i_1}^{k_1}, v_{i_2}^{k_2}, \dots, v_{i_n}^{k_n}, v_{0e}^{k_{n+1}}, v_d\}$  with the feasible TSPTDC solution  $H$  in  $G$  consisting of the Hamiltonian circuit  $\{v_0, v_{i_1}, v_{i_2}, \dots, v_{i_n}, v_0\}$  starting at  $v_0$  at time  $k_0 \in [a_0, b_0]$ , leaving vertex  $v_{i_r}$  at time  $t_{i_r}^{k_r} = a_{i_r} + k_r - 1 \in [a_{i_r}, b_{i_r}] \forall r \in \{1, \dots, n\}$  and ending at  $v_0$  at time  $a_0 + k_{n+1} - 1 \in [a_0, b_0]$  (note that two feasible TSPTDC solutions in  $G$  with the same Hamiltonian circuit but at least one different leaving instant of time  $t_i^k$  are taken as different solutions). Both  $T'$  and  $H$  have the same cost; so an optimal AGTSP solution in  $G'$  gives rise in an easy way to an optimal TSPTDC solution in  $G$ .

As  $|V'|$  depends on the width of the time windows besides  $|V|$  ( $|V'|$  is  $O((n+1)p^*)$  where  $p^* = \max_{0 \leq i \leq n} (b_i - a_i + 1)$ ), we conclude that this transformation is pseudo-polynomial. ■

Fig. 2. Auxiliary digraph  $G'$ 

In this way, the TSPTDC can be solved from a theoretical point of view. Nevertheless, the size of the auxiliary digraph could be too large to apply known procedures to solve its associated ATSP in some real distribution problems inside large cities: the servicing time windows of the customers could have a relatively small size (for example one or two hours) after preliminary studies and negotiations, but the depot time window should be opened during the entire working day. If there are customers to be serviced early and customers to be serviced at the end of the working day, each one of the sets  $S_{0s}$  and  $S_{0e}$  will contain about  $b_0 - a_0$  vertices. For example, with this condition, in an 8-hour working day with a unit of time equal to 1 minute (this is the smallest time unit normally considered in real vehicle routing problems inside large cities), the depot could give rise to about  $8 \times 60 \times 2 = 960$  vertices in the auxiliary digraph.

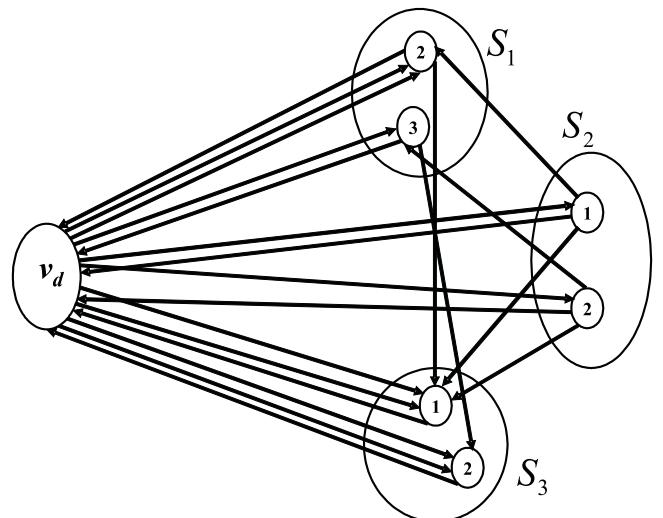
Although nowadays there are exact procedures capable of solving large-scale ATSP instances with thousands of vertices, as we mentioned in the introduction, this transformation does not seem very attractive to solve the TSPTDC because of the size of the depot time window.

We show next that the size of the auxiliary digraph can be considerably reduced thus becoming more competitive. In fact, in the “reduced” auxiliary digraph, the number of vertices generated from the depot will always be one, independently of the size of the depot time window. Thus, in the example given above, we would only have 1 vertex vs the about 960 vertices (a very considerable reduction), and in our example of Figure 1, we would have 1 vertex vs the 10 vertices in Figure 2.

Let then  $G' = (V', A')$  be the auxiliary digraph obtained from the original TSPTDC instance. From  $G'$  we construct a reduced auxiliary digraph  $G'' = (V'', A'')$  in the following way:

- Remove all vertices of  $G'$  corresponding to the subsets  $S_{0s}$  and  $S_{0e}$ .
- Maintain the rest of vertices of  $G'$  including  $v_d$ .
- For every vertex  $v \in G''$  different from  $v_d$  do  $\text{cost}(v_d, v) = \min_k \{\text{cost}(v_{0s}^k, v)\}$ .
- For every arc  $(v, v_{0e}^k)$  with finite cost in  $G'$  do  $\text{cost}(v, v_d) = \text{cost}(v, v_{0e}^k)$ .
- Maintain the arc costs between vertices belonging to different sets  $S_i$  with  $i \in \{1, \dots, n\}$ .
- Remove all vertices  $v_i^k \in G''$  verifying one of the three following conditions, understanding that an arc  $(u, v)$  exists in  $G''$  if it has been assigned before a finite value to  $\text{cost}(u, v)$ :
  - $d^+(v_i^k) = 0$  or  $d^-(v_i^k) = 0$ .
  - $d^+(v_i^k) = d^-(v_i^k) = 1$  corresponding to arcs  $(v_d, v_i^k)$  and  $(v_i^k, v_d)$ .
  - $d^-(v_i^k) \geq 1$ ,  $d^+(v_i^k) = 1$  corresponding to arc  $(v_i^k, v_d)$  and it exists at least one index  $j$  ( $j \neq i$ ) verifying that  $a_i + k - 1 \leq a_j$ .

Figure 3 shows the reduced auxiliary digraph  $G''$  from  $G'$  in Figure 2 corresponding to our example. As we have said, the number of vertices generated from the depot is 1 vs 10 vertices in Figure 2, and vertices  $v_1^1$  and  $v_3^3$  have been removed, so  $G''$  has 7 vertices while  $G'$  has 18 vertices.

Fig. 3. Reduced auxiliary digraph  $G''$ 

We define in  $G''$  an AGTSP in the same terms as the AGTSP defined in  $G'$  (see the proof of Theorem 1), except that the subsets  $S_{0s}$  and  $S_{0e}$  have been removed

and that some  $S_i$  may contain fewer elements than in  $G'$  (the removed vertices).

**Theorem 2:** Solving the AGTSP in  $G'$  is equivalent to solving the AGTSP in  $G''$ .

**Proof:** Given a feasible AGTSP solution in  $G'$ , if we replace its initial sequence  $\{v_d, v_{0s}^k, v\}$  by the sequence (arc)  $\{v_d, v\}$  in  $G''$  and we replace its final sequence  $\{u, v_{0e}^m, v_d\}$  by the sequence (arc)  $\{u, v_d\}$  in  $G''$ , it is evident that we have a feasible AGTSP solution in  $G''$ . Moreover, the costs of  $\{u, v_{0e}^m, v_d\}$  and  $\{u, v_d\}$  are the same and in an optimal solution in  $G'$ , the cost of  $\{v_d, v_{0s}^k, v\}$  must necessarily be  $\min_k \{cost(v_{0s}^k, v)\}$ , which is the cost of  $\{v_d, v\}$  in  $G''$ .

On the other hand, there is no feasible AGTSP solution in  $G'$  containing a vertex  $v_i^k$  satisfying one of the conditions (i), (ii) and (iii) given above: it is evident for condition (i); it is evident for condition (ii) except for the trivial case  $n = 1$  that should not be considered as an AGTSP; and condition (iii) means that subset  $S_j$  will not be visited by the solution.

Thus, an optimal AGTSP solution in  $G'$  gives rise to a feasible AGTSP solution in  $G''$  with the same cost, and with the same reasoning, a feasible AGTSP solution in  $G''$  gives rise to a feasible AGTSP solution in  $G'$  with the same cost, and so, an optimal AGTSP solution in  $G'$  results in an optimal AGTSP solution in  $G''$  and vice versa. ■

Therefore, we can solve a TSPTDC in graph  $G$  by solving an AGTSP in  $G''$ . Following with our example, once we have the reduced auxiliary digraph  $G''$ , according to the transformation of  $G''$  given by Noon and Bean, in Figure 4 we show the ATSP optimal solution corresponding to our example, from which we easily obtain the AGTSP optimal solution to  $G''$  given in Figure 5 and then, the TSPTDC optimal solution in  $G$  given in Figure 6 ( $v_0, v_2, v_1, v_3, v_0$ ) with time sequence (2, 3, 4, 5, 6) and with cost  $39 + 38 + 36 + 33 = 146$ . Note that there are no waiting times in this circuit solution because it leaves each vertex  $v_i$   $i > 1$  after time  $a_i$ ; remember that in this example we have supposed that  $[w_i^k, a_i + k - 1] = \{a_i + k - 1\}$  if  $k > 1$ . But also note that this optimal circuit does not start at time  $a_0$ , which is equal to 1 in this case (there is a waiting time in the warehouse).

#### IV. COMPUTATIONAL EXPERIMENTS

In order to check the efficiency of this transformation, some computational experiments were performed on 270 instances obtained by modifying benchmark TSPTW instances (see for example [4], [12] and [18]) as follows:

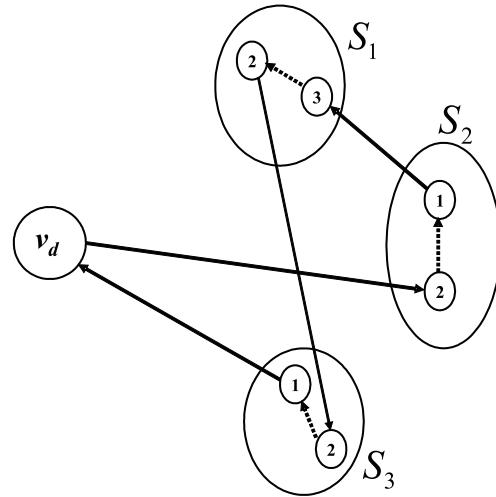


Fig. 4. ATSP optimal solution

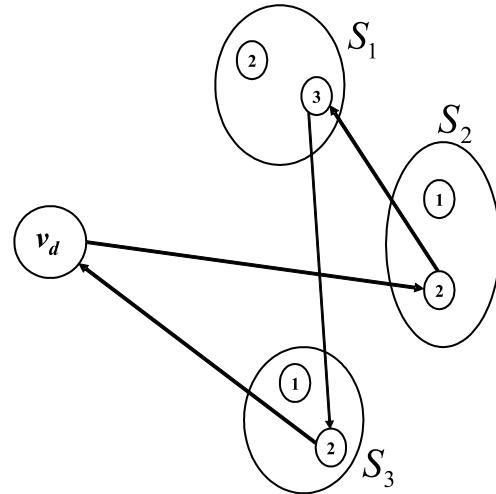


Fig. 5. AGTSP optimal solution in  $G''$

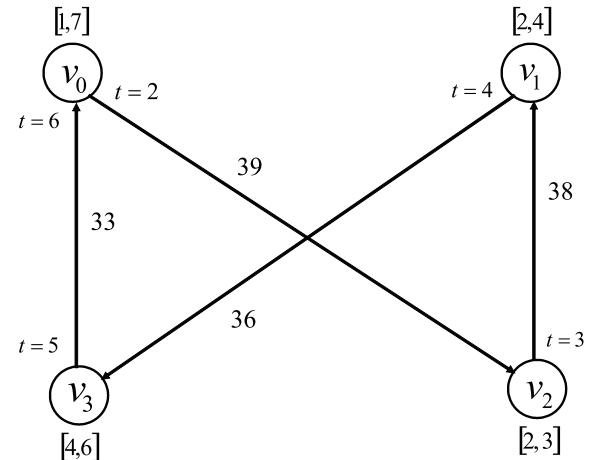


Fig. 6. TSPTDC optimal solution in  $G$

- As it occurs in the TSPTW, in these instances it is allowed to arrive at a vertex  $i$  before  $a_i$  (waiting time with zero cost); but in this case, the circuit will leave  $i$  at time  $a_i$  and no waiting time is allowed inside a time window, except for the depot, that is, in order to minimize the cost, instead of starting the circuit at time  $a_0$  it is allowed to wait for a period of time with zero cost and start at a time  $t_0 > a_0$ . This waiting time may be very important to minimize the cost if  $a_0$  belongs to a rush-hour, as it will occur in all these instances. During this initial waiting time the driver can work inside the warehouse.

- Since in the TSPTDC the travel time of each arc  $(v_i^k, v_j^l)$  is greater than or equal to a unit of time (5, 2 or 1 minute in these instances), and some original TSPTW instances have several vertices with the same or similar tight time windows, these instances may have no TSPTDC solution; this will happen if it is impossible to visit all vertices in their time windows consuming at least one time unit each time an arc is traversed, especially for 5-minute time unit. Therefore, in order to guarantee a priori the existence of a solution in the generated TSPTDC instances, we decided to remove some vertices from each original instance.

- We weighted the time windows corresponding to a working day from 8:00h. till 20:00h. in a department store in all the generated instances, such that for all  $i$ ,  $a_i$  and  $b_i$  be divisible by all the time units considered (5, 2 and 1) and then by 10. Table II shows an example of this weighting of the time windows, with one depot and two customers.

TABLE II  
EXAMPLE OF WEIGHTING OF THE TIME WINDOWS

Original windows		Weighted windows	
$a_i$	$b_i$	$a_i$	$b_i$
0	547	0	720
139	147	180	190
62	83	80	110

- Finally, in all these TSPTDC instances we considered the cost and the travel time of traversing an arc  $(v_i, v_j)$  at time  $t_i^k$  as the integer part of  $p(t_i^k) \cdot |(v_i, v_j)|$ ,  $p$  being a weight function that depends on the time interval to which  $t_i^k$  belongs, and  $|(v_i, v_j)|$  being the Euclidean distance between  $v_i$  and  $v_j$ . We established the time intervals range shown in Table III, considering traffic density in a large city of Spain, i.e. the rush-hours, such as leaving from or going to school or work.

We constructed TSPTDC instances with 10, 20, 30,

TABLE III  
TIME INTERVALS AND ITS WEIGHTS

Time interval	$p(t_i^k)$
[8:00, 9:40[	$p(t_i^k) = 1$
[9:40, 11:40[	$p(t_i^k) = 0.5$
[11:40, 12:40[	$p(t_i^k) = 0.75$
[12:40, 13:30[	$p(t_i^k) = 0.65$
[13:30, 15:20[	$p(t_i^k) = 1$
[15:20, 16:20[	$p(t_i^k) = 0.5$
[16:20, 18:40[	$p(t_i^k) = 0.75$
[18:40, 20:00[	$p(t_i^k) = 1$

40, 50 and 60 vertices taking into account three different maximum widths of the time windows in the original TSPTW instances (20, 40 and 60) and three time units (5, 2 and 1 minute). With all these data we generated sets of five instances for each number of vertices, each maximum width and each time unit, i.e. a total of  $5 \times 6 \times 3 \times 3 = 270$  TSPTDC instances, grouped into 3 sets of 90 instances depending on the time unit.

For each one of these 270 instances we first constructed the auxiliary digraph  $G'$ , then the reduced auxiliary digraph  $G''$ ; in  $G''$  we defined the corresponding AGTSP; using Noon and Bean transformation, we constructed the ATSP instance to be solved, and finally we transformed the ATSP instance into a GATSP instance by adding a large positive number to each arc cost in order to assure the occurrence of a Hamiltonian cycle in the optimal solution provided a Hamiltonian cycle exists (note that from a TSPTDC instance we obtain an ATSP instance whose digraph is far from being a complete digraph).

As the GATSP is a particular case of the MGRP and encouraged by the structure of our ATSP instances (few arcs with respect to the complete digraphs) and preliminary results, we used the exact algorithm for the MGRP given in [9] to solve optimally our ATSP instances. Remember that this algorithm is a cutting-plane procedure based on the polyhedral study on the MGRP presented in [8] and [9] in which the branch-and-bound option of CPLEX [11] is invoked when violated inequalities are not found. The algorithm is coded in C and run on a PC with a 1.8 GHz Pentium IV processor, using CPLEX 8.0 as an LP solver.

In tables IV to VI each row corresponds to a set of similar instances, with the following notation:

- V: number of vertices in the TSPTDC instance including the depot.

- W: maximum width of the time windows in the orig-

inal TSPTW instance from which the TSPTDC instance has been constructed.

- I: number of instances in the set.
- $VG''$ : average number of vertices in  $G''$  rounded to integer.
- ANA: average number of arcs in the GATSP instances obtained from  $G''$  rounded to integer.
- O: number of instances optimally solved in less than three hours.
- AT: average time in seconds to obtain the optimal solution in the solved (in less than three hours) GATSP instances obtained from  $G''$ .
- WT: worst time in seconds to obtain the optimal solution in the solved GATSP instances obtained from  $G''$ .

TABLE IV  
COMPUTATIONAL RESULTS WITH THE 5-MINUTE TIME UNIT

V	W	I	$VG''$	ANA	O	AT	WT
10	20	5	45	320	5	1.06	1.37
10	40	5	76	564	5	1.02	1.10
10	60	5	100	766	5	1.34	1.59
20	20	5	94	1166	5	2.10	3.68
20	40	5	156	1926	5	3.85	6.04
20	60	5	211	2727	5	9.75	21.70
30	20	5	138	2409	5	3.33	4.40
30	40	5	225	4051	5	16.13	27.41
30	60	5	253	4600	5	36.31	86.24
40	20	5	184	4192	5	7.90	13.84
40	40	5	314	7278	5	46.68	67.99
40	60	5	447	10384	5	918.31	2135.61
50	20	5	239	6678	5	18.95	24.39
50	40	5	386	11007	5	180.07	365.97
50	60	5	513	14711	5	468.36	1010.69
60	20	5	297	9805	5	39.58	51.02
60	40	5	476	16052	4	2929.64	5450.47
60	60	5	622	20956	4	1128.59	1438.70

As expected, the running time of this exact and exponential algorithm increases with the number of vertices in the TSPTDC instance, with the width of the time windows and with a smaller time unit, because all of them increase the number of vertices and the number of arcs in the corresponding GATSP instance to be solved by the algorithm.

We point out that instances with 10 vertices are optimally solved in less than one second or within few seconds, even for 1-minute time unit and instances with

TABLE V  
COMPUTATIONAL RESULTS WITH THE 2-MINUTE TIME UNIT

V	W	I	$VG''$	ANA	O	AT	WT
10	20	5	90	668	5	1.16	1.76
10	40	5	165	1240	5	1.40	1.81
10	60	5	224	1687	5	1.99	2.42
20	20	5	184	2295	5	2.71	4.12
20	40	5	342	4206	5	6.01	7.52
20	60	5	498	6370	5	28.05	67.34
30	20	5	269	4699	5	10.18	21.04
30	40	5	493	8775	5	38.77	71.46
30	60	5	574	10325	5	83.56	219.21
40	20	5	360	8129	5	15.03	24.16
40	40	5	690	15792	5	277.78	822.12
40	60	5	1041	24676	5	1997.25	6937.36
50	20	5	474	13120	5	39.81	46.96
50	40	5	850	23973	5	1017.48	2878.21
50	60	5	1200	33893	5	2039.66	5197.22
60	20	5	601	19722	5	112.05	162.36
60	40	5	1053	35066	4	3601.94	6295.39
60	60	5	1454	48290	3	3892.22	5484.04

TABLE VI  
COMPUTATIONAL RESULTS WITH THE 1-MINUTE TIME UNIT

V	W	I	$VG''$	ANA	O	AT	WT
10	20	5	180	1438	5	1.68	3.13
10	40	5	314	2349	5	2.29	2.75
10	60	5	422	3113	5	4.45	5.27
20	20	5	357	4415	5	4.74	8.41
20	40	5	707	8603	5	18.31	30.54
20	60	5	970	12266	5	94.49	146.38
30	20	5	517	8789	5	12.68	16.86
30	40	5	940	16582	5	114.08	228.28
30	60	5	1113	19813	5	182.58	354.76
40	20	5	741	16676	5	45.72	84.09
40	40	5	1316	29704	5	895.57	1670.34
40	60	5	1943	43937	5	3049.60	5497.44
50	20	5	909	24869	5	96.04	135.89
50	40	5	1631	45368	5	1740.35	3283.83
50	60	5	2343	65114	5	5916.77	8936.43
60	20	5	1465	47554	4	210.86	233.22
60	40	5	2021	66419	4	7735.53	10252.78
60	60	5	2810	91552	0	-	-

30 vertices are solved in less than one minute or few minutes. Furthermore, we have been able to optimally solve instances with up to 60 vertices, although with larger running times. From the tables we can see that the instances with 60 vertices produce very large-scale GATSP instances, even with more than 2,000 vertices and 60,000 arcs. Nevertheless, only 12 out the 45 instances with 60 vertices were not solved after 3 hours of running time. Note that for a real delivery route inside a large city with traffic problems, it seems very improbable to serve more than 50 or 60 customers in a working day.

In addition, upon comparing the average number of vertices and the average number of arcs in the ATSP instances (see columns 4 and 5 in the tables), it is evident that their corresponding digraphs are very far from being complete, then, it seems suitable to treat them as GATSP instances.

Therefore, we believe that the exact procedure for the MGRP proposed in [9] is a very good tool to optimally solve at least real TSPTDC instances with several dozens of customers within a reasonable time, even with a time unit equal to 1 minute, which is the smallest time unit normally considered in real vehicle routing problems inside large cities.

## V. CONCLUSIONS

Despite the presence of traffic jams in large cities at certain times and in certain areas, routing problems with time-dependent costs have hardly been studied because they are very difficult to model and to solve. In this paper we have presented a generalization of the well-known TSPTW in which the time and the cost of traversing an arc depend on the period of time at which we start traversing it; in this way more accurate solutions can be obtained for some real vehicle routing problems inside large cities, where the time or cost of traversing some streets depend on the moment of the day. This generalization can be transformed into an AGTSP and then into the classical ATSP for which several heuristic and exact procedures exist, even for large-scale instances with several thousands of vertices.

We have presented a computational experience on optimal resolution on a set of 270 TSPTDC instances adapted from benchmark TSPTDC instances. To obtain the optimal solutions we have applied the exact algorithm for the MGRP in [9] to the instances conveniently modified. Based on our findings, we believe that this exact algorithm is a very good tool to optimally solve real TSPTDC instances with several dozens of customers within a reasonable time -as no more customers are likely to be served in a working day- even with a time unit equal to 1 minute.

We are convinced that as computer power and speed increase, more and more the works on optimization vehicle routing problems will take into account time-dependent costs in order to approach more closely the mathematical models to the real problems. Recent works cited here justify our intuition. In this way, the theoretical results presented in this work can be used in the future as ideas or tools to check the efficiency of specific procedures to solve vehicle routing problems with time-dependent costs.

## ACKNOWLEDGEMENTS

Authors would like to thank Gendreau, Hertz, Laporte and Stan for providing us the set of benchmark TSPTW instances.

J.M. Sanchis wish to thank the Ministerio de Ciencia y Tecnología of Spain (project TIC2003-05982-C05-01) and the Generalitat Valenciana (Ref: GRUPOS03/189) their support.

## REFERENCES

- [1] N. Ascheuer, M. Fischetti, and M. Grötschell, "A polyhedral study of the asymmetric traveling salesman problem with time windows," *Networks*, vol. 36, pp. 69-79, 2000.
- [2] N. Ascheuer, M. Fischetti, and M. Grötschell, "Solving the asymmetric travelling salesman problem with time windows by branch-and-cut," *Mathematical Programming, Ser A*, vol. 90, pp. 475-506, 2001.
- [3] M. Blais and G. Laporte, "Exact solution of the generalized routing problem through graph transformations," *Journal of the Operational Research Society*, vol. 54, pp. 906-910, 2003.
- [4] R.W. Calvo, "A new heuristic for the traveling salesman problem with time windows," *Transportation Science*, vol. 34, pp. 113-124, 2000.
- [5] G. Carpaneto, M. Dell'Amico, and P. Toth, "Exact solution of large-scale, asymmetric traveling salesman problems," *ACM Transactions on Mathematical Software*, vol. 21, pp. 394-409, 1995a.
- [6] G. Carpaneto, M. Dell'Amico, and P. Toth, "Algorithm 750: CDT: A subroutine for the exact solution of large-scale, asymmetric traveling salesman problems," *ACM Transactions on Mathematical Software*, vol 21, pp. 410-415, 1995b.
- [7] S. Chopra and G. Rinaldi, "The graphical asymmetric traveling salesman polyhedron: symmetric inequalities," *SIAM J. Discrete Math.*, vol. 9, pp. 602-624, 1996.
- [8] A. Corberán, A. Romero, and J.M. Sanchis, "The mixed general routing polyhedron," *Mathematical Programming Ser. A*, vol. 96, pp. 103-137, 2003.
- [9] A. Corberán, G. Mejía, and J.M. Sanchis, "New results on the mixed general routing problem," *Operations Research*, vol. 53, pp. 363-376, 2005.
- [10] G. Cornuéjols, J. Fonlupt, and D. Naddef, "The traveling salesman problem on a graph and some related integer polyhedra," *Mathematical Programming*, vol. 33, pp. 1-27, 1985.
- [11] ILOG S.A., ILOG CPLEX 8.0, 2002.
- [12] Y. Dumas, J. Desrosiers, E. Gelinas, and M.M. Solomon, "An optimal algorithm for the traveling salesman problem with time windows," *Operations Research*, vol. 43, pp. 367-371, 1995.

- [13] B. Fleischmann, "A cutting plane procedure for the traveling salesman problem on road networks," *European Journal of Operations Research*, vol. 21, pp. 307-317, 1985.
- [14] B. Fleischmann, "A new class of cutting planes for the symmetric traveling salesman problem," *Mathematical Programming*, vol. 40, pp. 225-246, 1988.
- [15] B. Fleischmann, M. Gietz, and S. Gnutzmann, "Time-varying travel times in vehicle routing," *Transportation Science*, vol. 38, pp. 160-173, 2004.
- [16] F. Focacci, A. Lodi, and M. Milano, "A hybrid exact algorithm for the TSPTW," *INFORMS Journal on Computing*, vol. 14, pp. 403-417, 2002.
- [17] S. Ichoua, M. Gendreau, and J.Y. Potvin, "Vehicle dispatching with time-dependent travel times," *European Journal of Operational Research*, vol. 144, pp. 379-396, 2003.
- [18] M. Gendreau, A. Hertz, G. Laporte, and M. Stan, "A generalized insertion heuristic for the traveling salesman problem with time windows," *Operations Research*, vol. 46, pp. 330-335, 1998.
- [19] S.H. Kwon, H.T. Kim, and M.K. Kang, "Determination of the candidate arc set for the asymmetric traveling salesman problem," *Computers & Operations Research*, vol. 32, pp. 1045-1057, 2005.
- [20] C. Malandraki and M.S. Daskin, "Time dependent vehicle routing problems: Formulations, properties and heuristic algorithms," *Transportation Science*, vol. 26, pp. 185-200, 1992.
- [21] 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.
- [22] C.E. Noon and J.C. Bean, "An efficient transformation of the generalized traveling salesman problem," *INFOR*, vol. 31, pp. 39-44, 1993.
- [23] G. Pesant, M. Gendreau, J.Y. Potvin, and J.M. Rousseau, "An exact constraint logic programming algorithm for the traveling salesman problem with time windows," *Transportation Science*, vol. 32, pp. 12-29, 1998.
- [24] J.Y. Potvin, Y. Xu, and I. Benyahia, "Vehicle routing and scheduling with dynamic travel times," *Computers & Operations Research*, in press, 2006.

# Dimensioning and designing shifts in a call center

Cyril Canon<sup>\*†</sup>, Jean-Charles Billaut<sup>†</sup> and Jean-Louis Bouquard<sup>†</sup>

<sup>\*</sup>Vitalicom, 643 av du Grain d'Or, 41350 Vineuil, France

Email: ccanon@vitalicom.fr

<sup>†</sup>Université François-Rabelais de Tours, Laboratoire d'Informatique, 64 av. Jean Portalis, 37200 Tours, France

Email: {jean.billaut, jean-louis.bouquard}@univ-tours.fr

**Abstract**—In a Call Center, the critical phases of planning are dimensioning and shifts creation. The dimensioning problem consists in the determination of the ideal number of employees needed to face the demand. In this paper, this problem is modeled using a deterministic approach as a multi-purpose machine scheduling problem. A mixed integer linear program, with constraints propagation techniques and a constraint programming approach are proposed to solve this problem. The shift design problem consists in assigning employees to the shifts. A Tabu search algorithm is presented to solve this problem. Computational results are conducted and discussed, both in terms of computation time and of solutions quality.

**Keywords**—Call Center, dimensioning, Shift Design Problem, scheduling, Tabu Search

## INTRODUCTION

CALL Centers are used by organizations as an important channel of communication and transaction with their customers. The most prevalent form of communication is the telephone, even if the proliferation of the Internet allows to use new communication medias such as e-mail, chat, etc. When several types of communications are offered by a company, we call it a *customer contact center*, whereas when only the telephone is used, we call the company a *call center*. These new services have been expanding worldwide in both volume and scope, giving what is called now the *call center industry*. In [15], it is mentioned that “the number of call centers in Europe will grow from 12750 in 1999 to 28289 in 2006” and “Europe's call center market is around \$9 billions”, proving their socioeconomic importance in today's business landscape.

For this type of industry, personnel costs account for around 70% of the cost of running a typical call center. Increasing the service quality can be done, but also with an important personnel cost increase. Thus, there is a compromise to reach between service quality and costs. It is essential to efficiently manage telephone call centers, so that the customer requests are met

without excess staffing. In short, two questions are essential for a call center. The first question is: *How many agents are to be staffed in order to provide the required service quality?*, the related problem is called the *staffing problem* in the following. After answering this question, the number of needed agents is found for every time slot of the planning period. Then, the second question is: *How designing the assignment of shifts to agents?*, also called the *shift design problem* in the following. When solving this problem, a set of legal constraints have to be considered, respecting labor code and collective agreements. In this paper, we answer these two questions.

A lot of research studies have focused on staffing problems (see [4], [7], [18], etc.). From a modeling point of view, a call center can be viewed as a large system, operating in a stochastic environment, that is generally modeled as a  $M/M/N$  system, also called the Erlang-C model. This is “the most prevalent model that supports call center staffing” [4]. In [12], the authors present a state-of-the-art survey on possible models of a call center. They present the well known Erlang-C formulas and model a call center as a Markov Chain. They modify the classical model by adding some constraints like abandonment and retrial, for instance. In [17], the author concentrates on performance analysis and optimization using queueing models and describes mathematical methods and algorithms to relate these decision variables to technical as well as economic performance measures. In [1] the authors present an algorithm which gives the number of agents to hire in order to minimize a cost function. This cost function takes into account the price of an agent and the bonus and malus due to good or bad quality of service. In [10] the authors present a linear program in order to determine the number of agents to hire, considering learning phases and turnover. They present some particularities of the problem and a heuristic algorithm to solve quickly the mixed integer linear program.

In the literature, several approaches have been used to solve the shift design problem, most of them are heuristic algorithms. In [14], the authors propose a Tabu Search algorithm in order to create the shifts according to the agents constraints. In the problem tackled by the authors, breaks and annualized hours constraint are not considered. The annualized hours constraint implies that the number of work hours in a year can be irregularly spread over the weeks, but the annual amount of work hours is fixed. The main constraint in the problem tackled by the authors is to satisfy a workload curve. In [9], the authors propose a Simulated Annealing based method in order to choose the best shifts for the agents. The objective function is to maximize the quality of service. In [16], the author proposes a mixed method: shifts are chosen using integer programming and then are assigned to agents for each week using constraint programming. In the two methods, all the possible shifts are enumerated and the author assigns agents to the shifts. In the problem that we consider, the number of shifts is exponential and such a method cannot be applied. In [2], the authors present a three-step approach to assign shifts to agents considering the annualized hours constraint. The authors respect the Swiss law, which is less restrictive than the French one.

To the best of our knowledge, a deterministic approach has never been proposed to deal with the dimensioning problem. Furthermore, as far as we know, the shift design problem tackled in this paper has never been treated before.

The paper is organized as follows. In Section I the staffing problem is presented, the deterministic model is explained, a mixed integer linear programming formulation and a constraint programming formulation are proposed. In Section II, the shift design problem is presented and solved by a Tabu search algorithm. Computational experiments show the efficiency of the proposed methods.

## I. STAFF DIMENSIONING

We consider an inbound call center, it means a call center that only receives calls. The outgoing calls are not considered, the staff required for these calls is assumed to be fixed by the client that ordered the calls to the call center. The first stage of the process is to determine the temporal requirements, i.e. to determine the number of required employees of each qualification for every period of the planning horizon.

The staff dimensioning problem can be set as follows: the whole horizon is split into  $T$  identical periods of

size  $\tau$ . Generally, each period corresponds to a quarter of hour. For each period  $t$ , we know the number of expected incoming calls. These calls are of several types, depending on the client and on the activity. For instance hot line services, declarations of loss or robbery of credit cards, are different types of activities for which the mean duration of a call may differ. We denote by  $N_k(t)$  and  $P_k(t)$  the number and the duration of expected calls of type  $k$  ( $1 \leq k \leq K$ ) at period  $t$  respectively.

The staffing levels are generally determined from a service level perspective. Indeed, when a client entrusts an activity to a call center, he provides the forecasted calls for each period and the call center has to reach an objective in terms of *quality of service*. The quality of service is defined by a quantitative measure related to the service accessibility: in this paper, the quality of service is equal to the percentage of the average number of incoming calls taken in less than a given time, over the horizon. The quality of service depends on the activity. For instance, one client imposes a mean quality of service greater than or equal to 85% of incoming calls taken in less than 20 seconds during one week. This quality of service is a constraint for the staff dimensioning problem. We denote by  $R_k$  and  $QS_k$  the ratio and the time indicated in the definition of the quality of service for activity  $k$ .

The problem consists in determining the number of required agents at each period minimizing a cost function and respecting the quality of service of each activity.

### A. Problem modeling and notations

We consider one given period  $t$ . We assume each call arrives at a determined time and has to be answered according to the quality of service definition. A call may be not answered or answered after the delay, but the number of such calls is bounded according to the quality of service definition.

We assume that the number of agents in the call center is equal to  $m$ . Each agent  $j$  is considered as a resource denoted by  $M_j$ ,  $1 \leq j \leq m$ . We consider that all the agents constitute a particular workshop, composed by parallel machines (or resources), where all the resources are assumed to be identical. However, since agents do not have the same skills, they are not able to take the same type of calls. Therefore, we associate to each resource a set of tools, where each tool corresponds to the competence which possesses the corresponding agent, i.e. to the type of calls the agent can take. Such a workshop description is known in the scheduling literature as the “parallel multipurpose machines” model, denoted by  $PM\!PM$  [5]. A cost-in-use denoted by  $c_j$  is

associated to each resource  $M_j$ . This cost is a function of the number of tools associated to the resource.

We assume that  $n$  calls will occur during period  $t$ . Each call is considered as an independent job  $i$  with a release date  $r_i$ , a processing time  $p_i$  and a deadline  $\tilde{d}_i$ ,  $1 \leq i \leq n$ .  $N_k$  denotes the number of jobs corresponding to calls of type  $k$  (we omit  $t$  in the notation), we have  $\sum_{k=1}^K N_k = n$ . Finally, a tool is required for processing a job, representing the skill required to take the type of call.

We define the quantity  $AA_k = (\tau - P_k)/N_k$ , to denote the average time spent between two successive calls of type  $k$ . The release times, the processing times and the deadlines of the jobs corresponding to the calls of type  $k$  are computed as follows:

$$r_i^{(k)} = AA_k \times (i - 1), \forall i, 1 \leq i \leq N_k \quad (1)$$

$$p_i^{(k)} = P_k, \forall i, 1 \leq i \leq N_k \quad (2)$$

$$\tilde{d}_i^{(k)} = r_i^{(k)} + p_i^{(k)} + QS_k, \forall i, 1 \leq i \leq N_k \quad (3)$$

with  $QS_k$  the waiting time indicated in the definition of the quality of service for calls of type  $k$ . Note that this waiting time is called the slack in scheduling literature.

All the jobs are then sorted in their release time non decreasing order and renumbered from 1 to  $n$ .

The aim is to assign to the resources and to schedule a part of these jobs in order to minimize the total cost. This problem is solved for each period  $t$  which gives finally the number of agents per period needed to face the whole demand.

**Example:** Lets consider two types of jobs and the data of Table I.

type $k$	$QS_k$	$N_k$	$P_k$
$k = 1$	5	6	10
$k = 2$	5	7	30

TABLE I  
DATA OF THE EXAMPLE

We assume that  $T = 100$ , hence  $AA_1 = \frac{100-10}{6} = 15$  and  $AA_2 = \frac{100-30}{7} = 10$ . Associated to type 1 and to type 2 we obtain 13 jobs as indicated in Table II.

We assume that  $\max_{1 \leq k \leq K} QS_k \leq \min_{1 \leq i \leq n} p_i$ . This hypothesis is realistic in call center contexts, it means that the waiting times allowed by the quality of service definition is always smaller than the duration of the calls.

**Proposition:** Considering the previous hypothesis between duration of calls and waiting time, for any pair of jobs  $(i, i')$  assigned to the same resource, if  $r_i < r_{i'}$

type $j$	$k = 1$					
	1	2	3	4	5	6
$r_j$	0	15	30	45	60	75
$p_j$	10	10	10	10	10	10
$d_j$	15	30	45	60	75	90

type $j$	$k = 2$						
	1	2	3	4	5	6	7
$r_j$	0	10	20	30	40	50	60
$p_j$	30	30	30	30	30	30	30
$\tilde{d}_j$	35	45	55	65	75	85	95

TABLE II  
CORRESPONDING JOBS

then  $i'$  cannot precede  $i$  in a feasible solution.

**Proof:** the starting time of job  $i$  is denoted by  $t_i$ . Suppose that  $r_i < r_{i'}$  and  $i'$  precedes  $i$ . If  $i'$  begins at its earliest start time,  $t_{i'} = r_{i'}$ , and then  $t_{i'} + p_{i'} < \tilde{d}_{i'}$ . Because  $i$  follows  $i'$ , in best  $t_i = t_{i'} + p_{i'}$ , so  $t_i + p_i = t_{i'} + p_{i'} + p_i$ . But  $p_{i'} > QS_k$  if  $k$  is the type of job  $i$ , so  $t_i + p_i > t_{i'} + QS_k + p_i$ . Because  $r_i < r_{i'}$ , we have  $r_i < t_{i'}$ , so  $t_i + p_i > r_i + QS_k + p_i$ , which implies  $t_i + p_i > \tilde{d}_{i'}$ . ■

So, once the assignment of jobs to resources is known, the jobs are ordered by the *SRT* rule (Shortest Ready Time first). The problem is then reduced to an assignment problem. According to the three-field notation [11] of scheduling problems, our problem is denoted by  $PM\bar{P}M|r_i, \tilde{d}_i, QS_{max} < p_{min}|m^w$  where  $m^w$  is the cost function due to machine use,  $QS_{max}$  is the greatest waiting time and  $p_{min}$  the smallest processing time.

The problem can be formulated as a mixed integer linear program (called MILP) and as a constraint programming model (called CP).

### B. Integer linear program

In order to represent the skills constraints, we define boolean data  $b_{i,k} = 1$  if job  $i$  requires skill  $k$  and 0 otherwise ( $1 \leq i \leq n, 1 \leq k \leq K$ ). In the same way,  $v_{j,k}$  is equal to 1 if tool  $k$  is associated to resource  $M_j$  and 0 otherwise. The variables of the model are:  $x_{i,j}$  a boolean variable equal to 1 if job  $i$  is assigned to resource  $M_j$  and 0 otherwise;  $z_j$  equal to 1 if machine  $M_j$  is used and 0 otherwise; and  $t_i$  the starting time of job  $i$ .

The objective function is to Minimize  $Z = \sum_{j=1}^m c_j z_j$ .

The constraints are:

$$\left\{ \begin{array}{ll} \sum_{j=1}^m x_{i,j} = 1 & \forall i \in \{1..n\} \text{ (M1)} \\ x_{i,j} \leq \sum_{k=1}^K b_{i,k} \cdot v_{j,k} & \forall i \in \{1..n\}, \forall j \in \{1..m\} \text{ (M2)} \\ t_{i'} \geq t_i + p_i \times (2x_{i,j} + 2x_{i',j} - 3) & \forall i \in \{1..n\}, \forall i' \in \{i..n\}, \forall j \in \{1..m\} \text{ (M3)} \\ t_i \geq r_i & \forall i \in \{1..n\} \text{ (M4)} \\ t_i + p_i \leq \tilde{d}_i & \forall i \in \{1..n\} \text{ (M5)} \\ z_j \geq x_{i,j} & \forall i \in \{1..n\}, \forall j \in \{1..m\} \text{ (M6)} \end{array} \right.$$

Constraints (M1) insure that each job is assigned to exactly one resource. Constraints (M2) insure that a job is assigned to a resource, only if the required tool is present. The sum in the right hand side of the constraint is a constant, equal to 0 or 1. If the constant equal 0 the variable is not introduced. Constraints (M3) are the disjunctive constraints concerning two jobs assigned to the same resource. In these constraints  $i$  is smaller than  $i'$ , which means that  $i$  precedes  $i'$  if they are on the same resource. Constraints (M4) and (M5) insure the respect of the release dates and the deadlines. Constraints (M6) give the value to variables  $z_j$ . This model contains  $mn$  binary variables,  $(n+m)$  continuous variables and  $O(n^2m)$  constraints.

**Proposition:** Constraints (M3) are sufficient to model the disjunctive constraints.

**Proof:** Let  $i$  and  $i'$  two jobs such that  $i < i'$ .

We denote by  $sl_i$  the slack of job  $i$ , it means  $r_i + p_i + sl_i = \tilde{d}_i$ . This slack time  $sl_i$  is equal to  $QS_k$  if  $i$  is of type  $k$ . We have the hypothesis that  $sl_i \leq p_i$ , for all  $i$ . So  $r_i + p_i + p_i \geq \tilde{d}_i$ , i.e.  $r_i \geq \tilde{d}_i - p_i - p_i$ . Since  $r_{i'} \geq r_i$  and  $t_{i'} \geq r_{i'}$ , we have  $t_{i'} \geq \tilde{d}_i - p_i - p_i$ . Because  $t_i \leq \tilde{d}_i - p_i$ , we have  $t_{i'} \geq t_i - p_i$ . This relation is verified for any assignment of jobs. Since  $t_{i'} \geq t_i - p_i$ , a fortiori, we always have  $t_{i'} \geq t_i - 3p_i$ . Thus, if the jobs are not assigned to the same resource, this constraint is redundant.

Assume that these two jobs are assigned to the same resource  $M_j$ , then  $(2x_{i,j} + 2x_{i',j} - 3) = 1$  and thus  $t_{i'} \geq t_i + p_i$ , which corresponds to the succession of  $i$  and  $i'$ . ■

In order to improve the resolution process of this MILP, we use some constraint propagation techniques. Let consider two jobs  $i$  and  $i'$  with  $i < i'$ , we focus on constraints (M3) as follows:

- if  $\tilde{d}_i \leq r_{i'}$ , the disjunction will necessarily be respected, so the constraint is not introduced.

- if  $r_i + p_i \geq \tilde{d}_{i'} - p_{i'}$  then it is not possible to schedule  $i$  and  $i'$  on the same resource without overlapping. In this case, the corresponding disjunctive constraints (one per resource) are not introduced, but the constraints  $x_{i,j} + x_{i',j} \leq 1$  for all  $j$  are added in order to impose non identical assignments.

Another improvement concerns constraints (M6). An arbitrary order is given for all the resources of the same type, i.e. having the same set of tools, as follows:  $z_j \geq z_{j'}$ , for all  $M_j$  and  $M_{j'}$  of the same type with  $j < j'$ . These constraints break symmetry of the assignment problem.

The solver CPLEX is used as a resolution method for this problem.

### C. Constraint programming model

We denote by  $EJM_i$  the set of resources which can handle job  $i$  according to the tools associated to the resources.

The variables of the model are: for each job  $i$ , its starting time  $T_i$  and its assignment  $R_i$ . For each resource,  $Z_j$  is equal to 1 if machine  $M_j$  is used and 0 otherwise. The variable to minimize is denoted by  $C$ . In order to help the resolution, a lower bound is computed and added to the model. This lower bound is based on the notion of "mandatory part" [13], [3]. Let consider one job  $i$ . This job has to be processed between  $r_i$  and  $\tilde{d}_i$ , and more precisely, whatever is its starting time, the job will be performed in the interval  $[\tilde{d}_i - p_i, r_i + p_i]$ , if  $\tilde{d}_i - p_i < r_i + p_i$ . We denote by  $\mathcal{T}$  the set of dates  $\tilde{d}_i - p_i$  for all jobs  $i$ . We denote by  $\mathcal{S}_t$  the set of jobs that are processed simultaneously at time  $t$ ,  $\forall t \in \mathcal{T}$ . The number  $\max_{t \in \mathcal{T}} |\mathcal{S}_t|$  gives the minimum number of jobs that will be performed simultaneously. We assume that these jobs are assigned to the first cheaper resources, which gives a lower bound denoted by  $LB$ .

The constraints of the model (denoted CP) are the following.

$$\left\{ \begin{array}{ll} C = \sum_{j=1}^m w_j Z_j & \\ C \geq LB & (C1) \\ T_i \in [r_i, \tilde{d}_i - p_i] & \forall i \in \{1..n\} \text{ (C2)} \\ R_i \in EJM_i & \forall i \in \{1..n\} \text{ (C3)} \\ \text{alldifferent}(R_i, \forall i \in \mathcal{S}_t) & \forall t \in \mathcal{T} \text{ (C4)} \\ (R_i \neq R_{i'}) \text{ or } (T_i + p_i \leq T_{i'}) & \forall i \in \{1..n-1\}, \forall i' \in \{i+1..n\} \text{ (C5)} \\ (R_i \neq M_j) \text{ or } (Z_j = 1) & \forall i \in \{1..n\}, \forall j \in \{1..m\} \text{ (C6)} \\ Z_j \geq Z_{j'} & \forall j \in \{1..m\}, \forall j' \in \{j+1..m\} \text{ (C7)} \end{array} \right.$$

Constraint  $(C1)$  is the lower bound constraint. Constraints  $(C2)$  and  $(C3)$  indicate the definition domains of variables  $T_i$  and  $R_i$ . Constraints  $(C4)$  insure that the jobs of a same set  $\mathcal{S}_t$  will not have the same assignment. Constraints  $(C5)$  are the disjunctive constraints that indicate that if  $i$  and  $i'$  are assigned to the same resource and  $i < i'$  then  $i$  precedes  $i'$ . Constraints  $(C6)$  impose to variables  $Z_j$  to take value 1 if resource  $M_j$  is used. If  $R_i = M_j$ , this constraint imposes  $Z_j = 1$ . Constraints  $(C7)$  impose that if a resource  $M_j$  is not used ( $Z_j = 0$ ), then so do the resources of the same class that have a higher cost ( $M_{j'}$  with  $j' > j$ ).

The variables are instantiated as follows: all the variables  $Z_j$  are sorted according to their decreasing cost of use, and set to 0 first ("expensive machines last"); then all the couples of variables  $(R_i, T_i)$ .

The first modification of this model lies in the instantiation method. We denote by  $CP'$  the model  $CP$  with the following instantiation method: all the variables  $Z_j$  are sorted according to their increasing cost of use, and set to 1 first ("cheapest machines first"); then all the couples of variables  $(R_i, T_i)$ .

In the second modification of model  $CP$ , we insert an upper bound constraint on the criterion value. The upper bound is the result of a simple list algorithm: sort the jobs using SRT rule (Shortest Release Time first) and assign the jobs to the First Available Machine, that is able to process it. This algorithm has an  $O(n \log(n))$  time complexity. We denote by  $UB$  the value of the solution returned by this simple heuristic algorithm. The additive constraint is:  $C \leq UB$ ,  $(C8)$ . We denote by  $CP''$ , model  $CP$  (with the initial instantiation method) plus this constraint.

#### D. Computational experiments

Data are generated as described Table III. We consider four different activities, denoted by  $A$ ,  $B$ ,  $C$  and  $D$ . The mean duration of one call of each activity is respectively 60, 120, 45 and 90 seconds. We assume that calls have to be taken in less than 20 seconds. The set of agents is composed by 45 agents having the skills indicated table III. The costs of use are 1000 for one skill, 1300 for two skills and 1500 for three skills.

For each type of call, the number of expected calls is randomly generated between  $N$  and  $2N$  with  $N \in \{5, 10, 15, 20, 25\}$ . It means that for a given value of  $N$ , the total number of jobs  $n$  is comprised between  $4N$  and  $8N$ . 50 instances are generated for each value of  $N$ . The computational time of each instance is bounded by two minutes since the problem has to be solved for each period of the horizon. All the tests have

skill	$A$	$B$	$C$	$D$
max. # agents	5	5	5	5
skill	$AB$	$AC$	$BD$	$ABD$
max. # agents	5	5	5	5
number of skills	1	2	3	
cost of use	1000	1300	1500	

TABLE III  
DATA GENERATION

been realized using CPLEX 8.0 software for testing MILP and Eclipse 5.8\_77 software for testing CP, on a PC Pentium III, 1.4 GHz, 512 Mo.

Results are presented in Table IV. Column #opt. indicates the number of instances for which the optimal solution has been returned in less than two minutes. Column #feas. indicates the number of instances for which a feasible solution – non optimal – has been found, column #unfeas. indicates the number of instances for which no feasible solution has been found. Column CPU(s) indicates the average computation time in seconds. Column #impr. indicates the number of improvements, i.e. the number of instances for which the solution returned by the method is strictly better than  $UB$ .

Method MILP has the advantage to always find a feasible solution to the problem. However, this solution is never optimal – never proved optimal – and is never strictly better than  $UB$ . So, using the heuristic algorithm is always better than using MILP.

Methods  $CP$  and  $CP'$  give complementary results. For  $N = 5$ ,  $CP$  always returns the optimal solution in a very small computation time; for  $N = 10$ ,  $CP$  either finds the optimal or a feasible solution, in less time than  $CP'$  that only gives feasible solutions. For  $N = 15$  the difference is not so evident since  $CP$  finds less feasible solutions than  $CP'$ , but the solutions are of better quality as indicated in column #impr. For  $N = 20$  or  $25$ ,  $CP$  does not find any solution to the problem – except for 10 instances – whereas  $CP'$  always finds a feasible solution. However, all these solutions are worst than  $UB$ . Method  $CP''$ , that cannot be worst than  $UB$  has the advantages of both  $CP$  and  $CP'$ . Adding the upper bound to  $CP$  does not help the solver finding an optimal solution, it only prevents unfeasible solutions. For  $N \leq 15$  adding the upper bound constraint is interesting since the method can improve some solutions. However, for  $N \geq 20$ , using any  $CP$  model is not interesting, since it is always better to use directly the solution of the heuristic algorithm – except for one instance.

MILP						
<i>N</i>	<i>n</i>	#opt.	#feas.	#unfeas.	CPU (s)	#impr.
5	[20,40]	0	50	0	120	0
10	[40,80]	0	50	0	120	0
15	[60,120]	0	50	0	120	0
20	[80,160]	0	50	0	120	0
25	[100,200]	0	50	0	120	0

CP						
<i>N</i>	<i>n</i>	#opt.	#feas.	#unfeas.	CPU (s)	#impr.
5	[20,40]	50	0	0	3.33	21
10	[40,80]	27	23	0	84.33	27
15	[60,120]	0	40	10	120	13
20	[80,160]	0	10	40	120	1
25	[100,200]	0	0	50	120	0

CP'						
<i>N</i>	<i>n</i>	#opt.	#feas.	#unfeas.	CPU (s)	#impr.
5	[20,40]	38	12	0	74.18	17
10	[40,80]	0	50	0	120	0
15	[60,120]	0	50	0	120	0
20	[80,160]	0	50	0	120	0
25	[100,200]	0	50	0	120	0

CP''						
<i>N</i>	<i>n</i>	#opt.	#feas.	#unfeas.	CPU (s)	#impr.
5	[20,40]	50	0	0	3.52	21
10	[40,80]	28	22	0	84.80	27
15	[60,120]	0	50	0	120	16
20	[80,160]	0	50	0	120	1
25	[100,200]	0	50	0	120	0

TABLE IV  
RESULTS

Integer programming and constraint programming results are compared, using MILP and CP' model. Table V indicates in column #M<C' and #M=C' the number of instances for which the result returned by MILP is strictly smaller than – respectively equal to – the one returned by CP'. Column av.M and column av.C' indicate the average objective function value for both methods. A star in these columns indicates the best one.

<i>N</i>	<i>n</i>	#M<C'	#M=C'	av.M	av.C'
5	[20,40]	0	0	8300	*4436
10	[40,80]	15	2	12074	*11760
15	[60,120]	41	0	*16840	21428
20	[80,160]	50	0	*22156	41586
25	[100,200]	50	0	*29784	58294

TABLE V  
COMPARISON BETWEEN MILP AND CP'

For a small value of *N* (*N* ≤ 10), CP' gives better results than MILP, even if the average values of the objective function are similar for *N* = 10. However, for *N* ≥ 15, results returned by MILP are better than

those returned by CP' without ambiguity: in average, the objective value of the solutions returned by MILP is the half of those of CP', for *N* ≥ 20. However, as indicated in Table IV, the heuristic algorithm is always strictly better than MILP.

## II. SOLVING THE SHIFT DESIGN PROBLEM

Once the number of required agents is known for each period, the problem is how to define shifts for each agent, that is solving the shift design problem. The number of employees needed for each period to answer the calls is now a data of the problem.

We assume that all the contracts of the agents are known, it means the minimum and maximum number of work hours per day, the exact number of work hours in the week, the minimum and maximum duration of the break during the day, plus its relative position in the day, the minimum duration of rest between two consecutive days and the number of days-off per week. A solution that respects the legal constraints is called in the following *a legal solution*. Knowing the results of the preceding phase, i.e. the number of agents required per period, we have to determine for each agent *j* and for each day *d* of the week, the start time of work  $T_{j,d}$ , the duration of work  $P_{j,d}$  including the break, the start time of break  $S_{j,d}$  and the duration of break  $L_{j,d}$ . The shift for agent *j* day *d* is the vector  $(T_{j,d}, P_{j,d}, S_{j,d}, L_{j,d})$  and is represented in Figure 1.

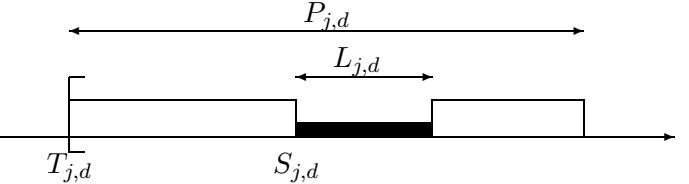


Fig. 1. Shift for agent *j*

The fact that the number of agents scheduled per period and per activity is greater than or equal to the required number is called *a feasible solution*. The objective is to obtain a solution both legal and feasible.

### A. Tabu search algorithm

This problem is solved by using a Tabu Search algorithm [14]. One solution is represented by the four matrices *T*, *P*, *S* and *L*. A solution is evaluated by a function of the total number of missing agents per period.

At the beginning, an initial legal solution is obtained as follows: for each agent *j* and for each day *d*,  $P_{j,d}$ ,  $S_{j,d}$ ,  $L_{j,d}$  are randomly generated in their definition domain. Then, for agent *j*, day *d*, variable  $T_{j,d}$  is assigned to the smallest period belonging to the

definition domain that do not respect the temporal requirement. This simple algorithm allows to generate in  $O(m)$  a legal solution, with no feasibility guaranty.

We define four basic neighbourhood relations changing one of the four features of a shift, and two composite moves:

- for agent  $j$  day  $d$ , increase or decrease
  - $P_{j,d}$
  - or  $L_{j,d}$
  - or  $T_{j,d}$
  - or  $S_{j,d}$ .
- swap the shift for agent  $j$  day  $d$  and the shift for agent  $j$  day  $d'$ ,
- increase  $P_{j,d}$  for agent  $j$  day  $d$  and decrease  $P_{j,d'}$  for agent  $j$  day  $d'$ .

In order to avoid cycles in the neighbourhood, a Tabu mechanism is used: the Tabu list contains the last  $H$  visited solutions, where  $H$  is the length of the Tabu list. Each time a solution is chosen in the neighbourhood, this solution is added to the Tabu list. When the Tabu list is full, the oldest solution is extracted. In fact, the Tabu list contains “*reduced solutions*”: a reduced solution is a matrix with one row for each type of contract (same number of work hours in the week and same min-max number of work hours each day) and one column for each period, representing how many agents having the same contract are assigned to each period. Two solutions are equivalent if the corresponding reduced solutions are identical. The size of the Tabu list has been fixed to 7.

No aspiration criteria is used.

For each solution, all the neighbours are generated. However, all of them are not explored since of all them are not legal solutions. This verification can be checked in polynomial time, assuming that the initial solution was legal.

Among all the neighbour that are not in the Tabu list, the neighbour with the best objective function is chosen for the next exploration. The Tabu algorithm stops when the solution has not been improved during the last 30 moves.

The objective function to minimize is defined as follows. We denote by  $Y$  the total number of hours of the employees, available for working in the considered horizon. We denote by  $W'$  the total number of work hours required for all the periods of the horizon. We define  $\rho = W'/Y$  that represents the percentage of use of the employees.  $\mathcal{C}$  denotes the load curve, i.e. the number of employees needed each time period.  $\mathcal{C}$

is the result of the previous phase.  $\mathcal{C}'$  defined by  $\mathcal{C}'(t) = (1+\rho)\mathcal{C}(t)$  is the augmented load curve, that corresponds to the optimal curve (see Figure 2). We denote by  $\mathcal{P}$  the curve representing the number of scheduled agents per period. This curve can be deduced from one set of matrices  $(T, P, S, L)$ . We define the objective function to minimize by:

$$Z = \alpha \cdot \max(0, \mathcal{C}(t) - \mathcal{P}(t)) + \beta \cdot \sum_{t=1}^T |\mathcal{C}'(t) - \mathcal{P}(t)|$$

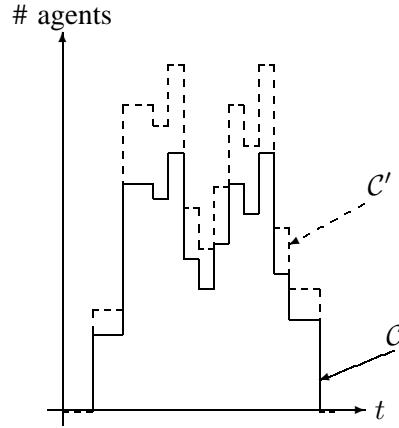


Fig. 2. Load curve definitions

The first term of this expression indicates that the total number of missing employees is weighted by coefficient  $\alpha$ . The aim of the second term is to smooth the overstaffing along the horizon. It is clear that if we obtain  $\mathcal{P} = \mathcal{C}'$ , the objective function value is equal to 0, since all the staff is used, no period is understaffed ( $\mathcal{C}'(t) \geq \mathcal{C}(t), \forall t \in [1, T]$ ) and the overstaffing is regular all along the horizon. Coefficients  $\alpha$  and  $\beta$  have been set to 100 and 1 respectively.

## B. Computational experiments

The number of agents required per period is generated as follows:

- for period  $t$ ,  $1 \leq t \leq 24$ , day  $d$ ,  $1 \leq d \leq 7$ , the number of agents required is uniformly generated between  $t$  and  $t \cdot nb_{max}$ ,
- for period  $t$ ,  $25 \leq t \leq 48$ , day  $d$ ,  $1 \leq d \leq 7$ , the number of agents required is uniformly generated between  $(49-t)$  and  $(49-t) \cdot nb_{max}$ ,
- for period  $t$ ,  $49 \leq t \leq 72$ , day  $d$ ,  $1 \leq d \leq 7$ , the number of agents required is uniformly generated between  $(t-48)$  and  $(t-48) \cdot nb_{max}$ ,
- for period  $t$ ,  $73 \leq t \leq 96$ , day  $d$ ,  $1 \leq d \leq 7$ , the number of agents required is uniformly generated

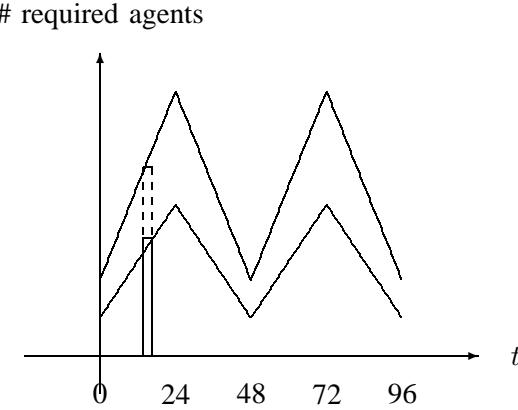


Fig. 3. Random generation of a typical loading curve

between  $(97 - t)$  and  $(97 - t).nb_{max}$ .

Parameter  $nb_{max}$  allows to vary the aspect of the curve by increasing the peaks size. It belongs to  $\{1.5, 2, 2.5, 3, 3.5\}$ . Such a data generation allows to obtain the characteristic “back of camel” curve of call center (see Figure 3). We denote by  $W_t$  the total number of agents required period  $t$  and we set  $W = \sum_{t=1}^T W_t$ . We compute a number of “full time equivalent” agents ( $FTE$ ) as follows. We insert a margin of 10% and we consider in fact  $1.1W$  agents. Since the period is of length  $\tau$  (in minutes) we have to consider  $1.1W/(\frac{60}{\tau})$  working hours, i.e.  $1.1W/4$  if  $\tau$  is a quarter of hour. Since one full time equivalent works 35 hours per week, we have to consider  $1.1W/(35 \times \frac{60}{\tau})$  agents and since each agent has exactly two days-off per week (he/she works only five days per week), it remains  $\frac{7}{5}(1.1W/(35 \times \frac{60}{\tau}))$ . We obtain  $FTE = \frac{1.1W\tau}{1500}$ .

We consider 6 types of contracts, the data are described in Table VI. Column type indicates the type of agent. Column  $\#Ag$  represents the number of agents of each type. Column  $\#Hr$  represents the number of work hours per week per agent. This number is not the same for all the types of agents due to the annualized hours constraints [2], [6], [8]. Column  $\#Hr/d$  represents the interval (minimum and maximum number) of work hours per day. Column  $\delta$  indicates the interval (minimum and maximum) of the gap between the start of the shift and the start of the break. Column  $L$  indicates the duration of the break (in hours).

The number of generated agents is  $7/6FTE$ , since it ensures that the call center contains enough agents for covering the demand. 125 data set are generated per value of  $nb_{max}$ . The results are presented in Table VII. Columns  $ILS$  and  $TA$  indicate the mean objective value of the initial legal solution ( $ILS$ ) and of the Tabu algorithm. A deviation between the initial legal solution

type	#Ag	#Hr	#Hr/d	$\delta$	$L$
1	1/6 FTE	35	[5,9]	[2,3]	1
2	1/6 FTE	33	[5,9]	[2,3]	1
3	1/6 FTE	38	[5,9]	[2,3]	1
4	1/6 FTE	32	[5,9]	[2,3]	1
5	1/6 FTE	37	[5,9]	[2,3]	1
6	2/6 FTE	20	[3,5]	[1,2]	1

TABLE VI  
DATA FOR THE SHIFT DESIGN PROBLEM

and the final solution is computed as follows:

$$\Delta(ILS/TA) = 100 \times \frac{ILS - TA}{ILS}$$

Columns  $\min\Delta$ ,  $\text{avg}\Delta$  and  $\text{max}\Delta$  represent the minimum, average and maximum deviation between  $ILS$  and  $TA$ . In Table VIII, columns  $\text{minCPU}$ ,  $\text{avgCPU}$  and  $\text{maxCPU}$  indicate the minimum, average and maximum CPU running time of the Tabu algorithm in seconds.

$nb_{max}$	$ILS$	$TA$	$\min\Delta$	$\text{avg}\Delta$	$\text{max}\Delta$
1.5	172.27	10.45	41.67%	89.68%	97.82%
2	399.85	56.01	39.67%	84.63%	98.57%
2.5	689.30	152.88	30.82%	76.10%	97.78%
3	1025.75	375.88	18.89%	61.52%	95.74%
3.5	1241.62	506.02	19.35%	58.03%	93.07%

TABLE VII  
COMPARISON OF THE QUALITY OF TA VS ILS

$nb_{max}$	$\text{minCPU}$	$\text{avgCPU}$	$\text{maxCPU}$
1.5	16.09	26.66	46.83
2	34.95	58.80	95.70
2.5	66.31	106.33	187.66
3	98.46	187.18	404.99
3.5	129.91	229.50	316.37

TABLE VIII  
COMPUTATION TIME OF TA

We can notice that the improvement of the initial solution by the Tabu algorithm is really significant. For  $nb_{max} \leq 2.5$ , the peak of load are not too large and the average improvement is more than 75% in average and always more than 30%. For  $nb_{max} \geq 3$ , the peaks are so important that it is needed to find a compromise between assigning employees to the peaks and to the rest of the horizon. For all these instances, it is difficult to know if a solution better than the one returned by TA exists. Note also that a solution with an optimal value of 0 rarely exists.

The running time of TA is reasonable since it is less than 4 minutes on average and it never exceeds 7 minutes, even for instances with  $nb_{max} \geq 3$ .

### CONCLUSION AND FURTHER DIRECTIONS

We have considered in this paper the workforce scheduling problem of an inbound Call Center, decomposed into two successive subproblems: first, the dimensioning problem that consists in the determination of the ideal number of employees needed to face the demand and second, the shift design problem that consists in assigning employees to the shifts.

The first problem has been solved using a mixed integer linear program, some constraint programming models and a heuristic algorithm. The heuristic algorithm is interesting for big instances, and the heuristic algorithm plus one constraint programming model are better for small instances. The second problem is solved using a Tabu search algorithm. The method gives interesting results and is very promising to solve this problem.

The resolution of the first problem may be improved by using the heuristic algorithm, a steepest descent and constraint programming algorithm to check the constraints violations. This method plus another method based on ant colony optimization will be compared to the preceding methods both in terms of quality and running time. Concerning the second problem, it seems that the Tabu search algorithm can be improved by exploring firstly composite moves and secondly basic moves. Furthermore, a lower bound could be usefull to measure more precisely the quality of the solutions returned by the method.

### REFERENCES

- [1] O. Z. Aksin and P. T. Harker, "Capacity sizing in the presence of a common shared ressource : dimensioning an inbound call center," *EJOR*, vol. 147, pp. 464–483, 2003.
- [2] C. S. Azmat and M. Widmer, "A case study of single shift planning and scheduling under annualized hours: A simple three-step approach." *European Journal of Operational Research*, vol. 153, no. 1, pp. 148–175, 2004.
- [3] P. Baptiste, C. L. Pape, and W. Nuijten, *Constraint-based Scheduling: Applying Constraints to Scheduling Problems*. Dordrecht: Kluwer Academic Publishers, 2001.
- [4] S. Borst, A. Mandelbaum, and M. I. Reiman, "Dimensioning large call center," *Operations Research*, vol. 52, no. 1, pp. 17–34, 2004.
- [5] P. Brucker, *Scheduling Algorithms*, 3rd ed. Berlin: Springer-Verlag, 2001.
- [6] C. Canon, J.-L. Bouquard, and J.-C. Billaut, "Staff planning in a call centre industry," in *Conference of the Italian Association of Operations Research (AIRO'03)*, Venice (Italy), Sept. 2003, p. 73.
- [7] B. P. K. Chen and S. G. Henderson, "Two issues in setting call centre staffing levels." *Annals of Operations Research*, vol. 108, pp. 175–192, 2001.
- [8] A. Corominas, A. Lusa, and R. Pastor, "Planning annualised hours with a finite set of weekly working hours and joint holidays," *Annals of Operations Research*, vol. 128, pp. 217–233, 2004.
- [9] A. Fukunaga, E. Hamilton, J. Fama, D. Andre, O. Matan, and I. Nourbakhsh, "Staff scheduling for inbound call centers and customer contact centers," *AI Magazine*, 2002.
- [10] N. Gans and Y.-P. Zhou, "Managing learning and turnover in employee staffing," *Operations Research*, vol. 50, no. 6, pp. 991–1006, 2002.
- [11] R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. H. G. R. Kan, "Optimisation and approximation in deterministic sequencing and scheduling: a survey," *Annals of Discrete Mathematics*, vol. 5, pp. 236–287, 1979.
- [12] G. Koole and A. Mandelbaum, "Queueing models of call centers : an introduction," *Annals of operations research*, vol. 113, pp. 41–59, 2002.
- [13] P. Lopez, J. Erschler, and P. Esquirol, "Ordonnancement de tâches sous contraintes : une approche énergétique (in french)," *RAIRO APII*, vol. 26, no. 6, pp. 453–481, 1992.
- [14] N. Musliu, A. Schaefer, and W. Slany, "Local search for shift design," *European Journal of Operational Research*, vol. 153, no. 1, pp. 51–64, 2003.
- [15] C. C. News. (2002) Internet draft. [Online]. Available: <http://www.callcenternews.com/resources/statistics.shtml>
- [16] A. Partouche, "Planification d'horaires de travail : Méthodologie, modélisation et résolution à l'aide de la programmation linéaire en nombres entiers et de la programmation par contraintes (in french)," Thèse de doctorat, Université Paris-Dauphine, Paris, 1998.
- [17] R. Stolletz, *Performance analysis and optimization of inbound call centers*. Berlin: Lecture Notes in Economics and Mathematical Systems, Springer, 2003.
- [18] W. Whitt, "Dynamic staffing in a telephone call center aiming to immediately answer all calls," *Operations Research Letters*, vol. 24, pp. 205–212, 1999.



# A new concept of approximate efficiency in multiobjective mathematical programming

César Gutiérrez\*, Bienvenido Jiménez† and Vicente Novo‡,§

\*Universidad de Valladolid, Departamento de Matemática Aplicada

E.T.S.I. Informática, Edificio de Tecnologías de la Información y las Telecomunicaciones,  
Campus Miguel Delibes, s/n, 47011 Valladolid, Spain

Email: cesargv@mat.uva.es

†Universidad de Salamanca, Departamento de Economía e Historia Económica

Facultad de Economía y Empresa, Campus Miguel de Unamuno, s/n, 37007 Salamanca, Spain  
Email: bjimen1@encina.pntic.mec.es

‡Universidad Nacional de Educación a Distancia, Departamento de Matemática Aplicada

E.T.S.I. Industriales, c/ Juan del Rosal, 12, Ciudad Universitaria, 28040 Madrid, Spain  
Email: vnovo@ind.uned.es

**Abstract**—This paper deals with approximate ( $\varepsilon$ -efficient) solutions of multiobjective mathematical programs. We introduce a new  $\varepsilon$ -efficiency concept which extends and unifies different approximate solution notions defined in the literature. We obtain necessary and sufficient conditions via linear scalarization, which allow to study this new class of approximate solutions in convex multiobjective mathematical programs. Several classical  $\varepsilon$ -efficiency notions are considered in order to show the concepts introduced and the results obtained.

**Keywords**—Multiobjective mathematical programming, approximate solutions,  $\varepsilon$ -efficiency, scalarization, Weighting Method

## I. INTRODUCTION

**A**PPROXIMATE solutions are an usual kind of solution used in practice to solve optimization problems. There are two reasons for that: firstly, optimization models are simplified representations of real problems. Secondly, these models are solved frequently using iterative algorithms or heuristic methods and these procedures give approximations to the theoretical solution.

The notion of approximate solution in multiobjective mathematical programming is not unique. The first and most popular concept was introduced by Kutateladze [14]. This notion has been used to obtain vector variational principles, approximate Kuhn-Tucker type conditions, approximate duality theorems, resolution methods, etc. (see [3]–[7], [10], [11], [15], [16], [18]–[20], [22], [25], [28], [30]).

However, the  $\varepsilon$ -efficiency set obtained according to the Kutateladze's definition is sometimes too large. So,

several authors are proposed other  $\varepsilon$ -efficiency concepts (see for example [8], [9], [21], [26], [27]). In this work, all these notions are analyzed through a new concept that allows us to study them simultaneously.

We characterize this new  $\varepsilon$ -efficiency notion in convex multiobjective mathematical programs via linear scalarization, i.e., by means of approximate solutions of associated scalar optimization problems. As a consequence of this characterization, we extend the classical Weighting Method to approximate efficiency sets obtained through different  $\varepsilon$ -efficiency notions.

The work is structured as follows: In Section II, the multiobjective mathematical program and the preference relation are fixed. Moreover, we describe some notations used in the sequel. In Section III, we propose a new  $\varepsilon$ -efficiency concept and we prove some properties of this notion when  $\varepsilon$  tends to zero. In Section IV, it is shown that our concept extends and unifies several  $\varepsilon$ -efficiency notions introduced previously in the literature by different authors: Kutateladze [14], Németh [21], Helbig [8] and Tanaka [26]. In Section V, we characterize the  $\varepsilon$ -efficiency set in convex multiobjective mathematical programs through approximate solutions of scalar optimization problems. The scalarization process is based on the Weighting Method. In Section VI, the results attained in the previous section are applied to characterize several types of  $\varepsilon$ -efficient solutions in a convex Pareto context. Finally, in Section VII, conclusions are presented that summarize this work.

## II. PRELIMINARIES

We denote by  $\text{int}(C)$ ,  $\text{cl}(C)$ ,  $\text{bd}(C)$  and  $C^c$  the interior, the closure, the boundary and the complement

§Corresponding author.

of a set  $C \subset \mathbb{R}^p$ , respectively. The cone generated of a set  $C$  is defined by

$$\text{cone}(C) := \bigcup_{\alpha > 0} \alpha C.$$

We say that a set  $C$  is solid if  $\text{int}(C) \neq \emptyset$ . We denote the nonnegative orthant in  $\mathbb{R}^p$  by  $\mathbb{R}_+^p$ .

For a cone  $K \subset \mathbb{R}^p$ , its positive polar cone (resp. strict positive polar cone) is

$$K^+ = \{h \in \mathbb{R}^p : \langle h, d \rangle \geq 0, \forall d \in K\}$$

(resp.  $K^{s+} = \{h \in \mathbb{R}^p : \langle h, d \rangle > 0, \forall d \in K \setminus \{0\}\}$ ).

In this paper, we consider the multiobjective mathematical program

$$\text{Min}\{f(x) : x \in S\}, \quad (\text{P})$$

where  $f : \mathbb{R}^n \rightarrow \mathbb{R}^p$  and  $S \subset \mathbb{R}^n$ ,  $S \neq \emptyset$ . As usual, to solve (P) the following preference relation  $\leq$  defined in  $\mathbb{R}^p$  by a proper solid convex cone  $D \subset \mathbb{R}^p$  is used, which models the preferences stated by the decision maker:

$$y, z \in \mathbb{R}^p, y \leq z \iff y - z \in -D.$$

We do not require that  $0 \in D$  and we suppose that  $D$  is a pointed cone, i.e., a cone such that  $D \cap (-D) \subset \{0\}$  (it is assumed that  $\emptyset \subset A, \forall A \subset \mathbb{R}^p$ ).

*Definition 2.1:* It is said that a feasible point  $x_0 \in S$  is an efficient solution of (P) with respect to  $D$  (or an efficient solution for short) if

$$(f(x_0) - D) \cap f(S) \subset \{f(x_0)\}. \quad (1)$$

Let us note that if  $0 \in D$  (resp.  $0 \notin D$ ) then (1) becomes

$$(f(x_0) - D) \cap f(S) = \{f(x_0)\}$$

(resp.  $(f(x_0) - D) \cap f(S) = \emptyset$ ). We denote the set of efficient solutions of (P) with respect to  $D$  by  $E(f, S, D)$  and with respect to  $\text{int}(D)$  by  $WE(f, S, D)$  (these last efficient solutions of (P) are called weakly efficient solutions). It is clear that  $E(f, S, D) \subset WE(f, S, D)$ .

Next, we give a simple example to show the concepts of efficient and weakly efficient solution.

*Example 2.2:* Let us consider problem (P) with the following data:  $n = p = 2$ ,  $D = \mathbb{R}_+^2$ ,  $f(x, y) = (x, y)$ ,

$$S = \{(x, y) \in \mathbb{R}^2 : x \geq 1, y \geq 1, \max\{x, y\} \geq 2\}.$$

It is clear that

$$E(f, S, \mathbb{R}_+^2) = \{(1, 2), (2, 1)\}$$

and

$$WE(f, S, \mathbb{R}_+^2) = \text{bd}(S).$$

In Figure 1 we have drawn the feasible set  $S$  and its efficient and weakly efficient elements.

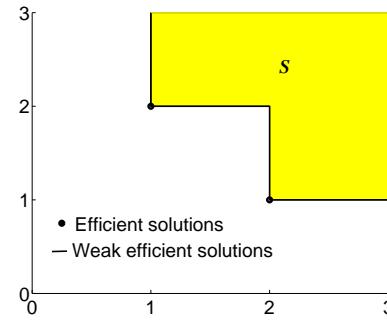


Fig. 1. Efficiency and weak efficiency sets in Example 2.2

### III. A NEW CONCEPT OF APPROXIMATE EFFICIENCY IN MULTIOBJECTIVE MATHEMATICAL PROGRAMMING

In the sequel, we introduce a new approximate solution concept for multiobjective mathematical programs. This notion is motivated in the following idea: an approximate solution of (P) is every feasible point  $x_0 \in S$  such that for all feasible point  $x \in S$  whose image  $f(x)$  is better than  $f(x_0)$ , the improvement  $f(x_0) - f(x)$  is near to zero.

To define this concept, we use a nonempty solid pointed convex set  $C \subset \mathbb{R}^p$  and we assume that  $C$  is co-radiant, i.e., a set such that  $\alpha d \in C, \forall d \in C, \forall \alpha > 1$ . Moreover, we denote  $C(\varepsilon) := \varepsilon C, \forall \varepsilon > 0$  and

$$C(0) := \bigcup_{\varepsilon > 0} C(\varepsilon). \quad (2)$$

*Lemma 3.1:*

- (i)  $C(\varepsilon)$  is a pointed convex co-radiant set,  $\forall \varepsilon > 0$ .
- (ii)  $C(\varepsilon_2) \subset C(\varepsilon_1)$ ,  $\forall \varepsilon_1, \varepsilon_2 > 0$ ,  $\varepsilon_1 < \varepsilon_2$ .
- (iii)  $C + C(\alpha) \subset C$ ,  $\forall \alpha > 0$ .
- (iv)  $C(\varepsilon) + C(\delta) \subset C(\varepsilon)$ ,  $\forall \varepsilon, \delta > 0$ .
- (v)  $C(\varepsilon) + C(0) \subset C(\varepsilon)$ ,  $\forall \varepsilon > 0$ .
- (vi)  $C(0)$  is a pointed convex cone.

*Proof.* Part (i). Consider  $\varepsilon > 0$ . It is obvious that  $C(\varepsilon)$  is a convex set, since  $C$  is convex. Let  $y \in C(\varepsilon)$  and  $\alpha > 1$ . There exists  $d \in C$  such that  $y = \varepsilon d$ . As  $C$  is a co-radiant set, it follows that  $\alpha y = \varepsilon(\alpha d) \in C(\varepsilon)$  and  $C(\varepsilon)$  is a co-radiant set. Moreover, if  $y \in -C(\varepsilon)$  then  $y/\varepsilon \in C \cap (-C) \subset \{0\}$ , and so  $y = 0$ . Thus,  $C(\varepsilon) \cap (-C(\varepsilon)) \subset \{0\}$  and  $C(\varepsilon)$  is a pointed set.

Part (ii). Let  $\varepsilon_1, \varepsilon_2 > 0$ ,  $\varepsilon_1 < \varepsilon_2$  and  $y \in C(\varepsilon_2)$ . There exists  $d \in C$  such that  $y = \varepsilon_2 d$ . For

$$\alpha := 1 + (\varepsilon_2 - \varepsilon_1)/\varepsilon_1$$

we have that  $y = \alpha(\varepsilon_1 d) \in C(\varepsilon_1)$ , since  $\alpha > 1$  and  $C(\varepsilon_1)$  is a co-radiant set. Then,  $C(\varepsilon_2) \subset C(\varepsilon_1)$ .

Part (iii). For each  $d_1, d_2 \in C$  and  $\alpha > 0$  it follows that

$$d_1 + \alpha d_2 = (1 + \alpha) \left( \left(1 - \frac{\alpha}{1 + \alpha}\right) d_1 + \frac{\alpha}{1 + \alpha} d_2 \right)$$

and  $d_1 + \alpha d_2 \in C$ , since  $C$  is a co-radiant convex set.

Part (iv). For each  $\varepsilon, \delta > 0$  it follows from part (iii) that

$$C(\varepsilon) + C(\delta) = \varepsilon(C + C(\delta/\varepsilon)) \subset \varepsilon C = C(\varepsilon).$$

Part (v). By part (iv) we see that

$$C(\varepsilon) + C(0) = \bigcup_{\delta>0} (C(\varepsilon) + C(\delta)) \subset C(\varepsilon), \forall \varepsilon > 0.$$

Part (vi). It is clear that  $C(0) = \text{cone}(C)$ , and so we have that  $C(0)$  is a cone.

If  $y \in C(0) \cap (-C(0))$  then there exist  $\delta, \nu > 0$  such that  $y \in C(\delta) \cap (-C(\nu))$ . Consider  $\beta = \min\{\delta, \nu\} > 0$ . By parts (i)-(ii) we see that  $y \in C(\beta) \cap (-C(\beta)) \subset \{0\}$ , and therefore,  $C(0)$  is a pointed set. Moreover, by a similar reasoning, if  $y_1, y_2 \in C(0)$  then there exists  $\beta > 0$  such that  $y_1, y_2 \in C(\beta)$  and it follows that  $\lambda y_1 + (1 - \lambda)y_2 \in C(\beta)$ ,  $\forall \lambda \in (0, 1)$ , since  $C(\beta)$  is a convex set. Consequently,  $C(0)$  is convex. ■

**Definition 3.2:** Let  $\varepsilon \geq 0$ . We say that a feasible point  $x_0 \in S$  is an  $\varepsilon$ -efficient solution of (P) with respect to  $C$  (or an  $\varepsilon$ -efficient solution for short) if

$$(f(x_0) - C(\varepsilon)) \cap f(S) \subset \{f(x_0)\}. \quad (3)$$

We denote by  $\text{AE}(f, S, C, \varepsilon)$  the set of  $\varepsilon$ -efficient solutions of (P) with respect to  $C$ . Let us observe that when  $\varepsilon = 0$  we have  $\text{AE}(f, S, C, 0) = \text{E}(f, S, C(0))$ .

Taking  $p = 1$  and  $C = (1, \infty)$  in Definition 3.2 we obtain the classical concept of approximate solution in (scalar) mathematical programming. We recall this notion in the following definition.

**Definition 3.3:** Consider  $p = 1$  in program (P) and  $\varepsilon \geq 0$ . It is said that  $x_0 \in S$  is an  $\varepsilon$ -solution of (P) if

$$f(x_0) - \varepsilon \leq f(x), \forall x \in S.$$

We denote the set of all  $\varepsilon$ -solutions of (P) when  $p = 1$  by  $\text{AMin}(f, S, \varepsilon)$ .

As  $C$  is a solid pointed convex co-radiant set, it follows that  $\text{int}(C)$  is a nonempty pointed convex co-radiant set and we can also consider the set of all  $\varepsilon$ -efficient solutions of (P) with respect to  $\text{int}(C)$  (or weakly  $\varepsilon$ -efficient solutions for short):

$$\{x \in S : (f(x) - \text{int}(C)(\varepsilon)) \cap f(S) \subset \{f(x)\}\}.$$

We denote this set by  $\text{WAE}(f, S, C, \varepsilon)$ . Notice that

$$\text{int}(C)(0) = \bigcup_{\varepsilon>0} \varepsilon \text{int}(C) = \bigcup_{\varepsilon>0} \text{int}(C(\varepsilon)). \quad (4)$$

Moreover, as  $C$  is a solid convex set it follows that (see [13, Proposition 2.3(ii)])

$$\begin{aligned} \text{int}(C)(0) &= \text{cone}(\text{int}(C)) = \text{int}(\text{cone}(C)) \\ &= \text{int}(C(0)) \subset C(0). \end{aligned} \quad (5)$$

Therefore,  $\text{AE}(f, S, C, \varepsilon) \subset \text{WAE}(f, S, C, \varepsilon)$ ,  $\forall \varepsilon \geq 0$  and

$$\begin{aligned} \text{WAE}(f, S, C, 0) &= \text{E}(f, S, \text{int}(C(0))) \\ &= \text{WE}(f, S, C(0)). \end{aligned}$$

To illustrate we give now an example (see Section IV for more important notions).

**Example 3.4:** Let  $h \in D^+ \setminus \{0\}$  and define

$$C := \{y \in \mathbb{R}^p : \langle h, y \rangle > 1\}.$$

It is clear that  $C$  is a proper solid pointed convex co-radiant set and  $C(\varepsilon) = \{y \in \mathbb{R}^p : \langle h, y \rangle > \varepsilon\}$ . Then we have for each  $\varepsilon \geq 0$  and  $x_0 \in S$

$$\begin{aligned} x_0 \in \text{AE}(f, S, C, \varepsilon) &\iff (f(x_0) - f(S)) \subset C(\varepsilon)^c \\ &\iff \forall x \in S, \quad \langle h, f(x) \rangle \geq \langle h, f(x_0) \rangle - \varepsilon. \end{aligned}$$

This says that  $x_0$  is an  $\varepsilon$ -efficient solution with respect to  $\langle h, \cdot \rangle$  in the sense introduced by Vályi in [27, Definition 2].

Let us consider the function  $h(x, y) = 2x + y$ . Then, the following Vályi's approximate efficiency set is obtained in problem (P) described in Example 2.2:

$$\text{AE}(f, S, C, \varepsilon) = \{(x, y) \in S : 2x + y \leq 4 + \varepsilon\}.$$

Figure 2 shows this set for  $\varepsilon = 2$  and  $\varepsilon = 0.5$ .

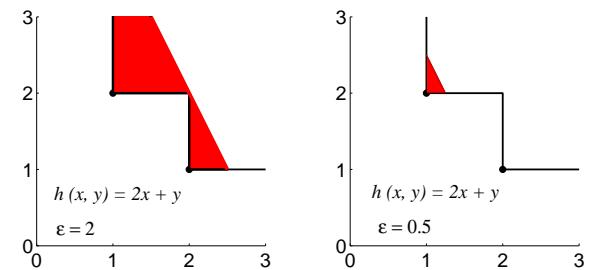


Fig. 2. Vályi's  $\varepsilon$ -efficiency sets in Example 2.2

Theorem 3.5 shows several properties of the family  $\{\text{AE}(f, S, C, \varepsilon)\}_{\varepsilon \geq 0}$ .

**Theorem 3.5:**

- (i)  $\text{AE}(f, S, C, 0) \subset \text{AE}(f, S, C, \varepsilon)$ ,  $\forall \varepsilon > 0$ .
- (ii)  $\text{AE}(f, S, C, \varepsilon_1) \subset \text{AE}(f, S, C, \varepsilon_2)$ ,  $\forall \varepsilon_1, \varepsilon_2 > 0$ ,  $\varepsilon_1 < \varepsilon_2$ .
- (iii)  $\bigcap_{\varepsilon>0} \text{AE}(f, S, C, \varepsilon) = \text{AE}(f, S, C, 0)$ .
- (iv) Let  $(x_n) \subset S$ ,  $(\varepsilon_n) \subset \mathbb{R}_+$  and  $y \in \mathbb{R}^p$  such that  $x_n \in \text{AE}(f, S, C, \varepsilon_n)$ ,  $\varepsilon_n \downarrow 0$  and  $f(x_n) \rightarrow y$ . Then  $f^{-1}(y) \cap S \subset \text{WAE}(f, S, C, 0)$ .
- (v) Let  $(x_n) \subset S$  and  $(\varepsilon_n) \subset \mathbb{R}_+$  such that  $x_n \in \text{AE}(f, S, C, \varepsilon_n)$  and  $\varepsilon_n \downarrow 0$ . Consider

$$K := \bigcap_n (f(x_n) - C(\varepsilon_n)).$$

Then  $f^{-1}(K) \cap S \subset \text{AE}(f, S, C, 0)$ .

*Proof.* Part (i). Let  $\varepsilon > 0$  and  $x \in \text{AE}(f, S, C, 0)$ . It follows that

$$\begin{aligned} (f(x) - C(\varepsilon)) \cap f(S) &\subset \left( f(x) - \bigcup_{\delta > 0} C(\delta) \right) \cap f(S) \\ &= (f(x) - C(0)) \cap f(S) \subset \{f(x)\} \end{aligned}$$

and  $x \in \text{AE}(f, S, C, \varepsilon)$ .

Part (ii). Let  $\varepsilon_1, \varepsilon_2 > 0$ ,  $\varepsilon_1 < \varepsilon_2$  and  $x \in \text{AE}(f, S, C, \varepsilon_1)$ . By Lemma 3.1(ii) we have that  $C(\varepsilon_2) \subset C(\varepsilon_1)$  and we deduce that

$$(f(x) - C(\varepsilon_2)) \cap f(S) \subset (f(x) - C(\varepsilon_1)) \cap f(S) \subset \{f(x)\}.$$

Then,  $x \in \text{AE}(f, S, C, \varepsilon_2)$ .

Part (iii). From part (i) it follows that

$$\text{AE}(f, S, C, 0) \subset \bigcap_{\varepsilon > 0} \text{AE}(f, S, C, \varepsilon).$$

Conversely, let  $x \in \bigcap_{\varepsilon > 0} \text{AE}(f, S, C, \varepsilon)$ . Then,

$$(f(x) - C(\varepsilon)) \cap f(S) \subset \{f(x)\}, \forall \varepsilon > 0.$$

Therefore,

$$\begin{aligned} (f(x) - C(0)) \cap f(S) &= \\ \left( f(x) - \bigcup_{\varepsilon > 0} C(\varepsilon) \right) \cap f(S) &\subset \{f(x)\} \end{aligned}$$

and  $x \in \text{AE}(f, S, C, 0)$ .

Part (iv). Let  $x \in f^{-1}(y) \cap S$  and suppose that there exists  $z \in S$  such that  $f(z) \in f(x) - \text{int}(C)(0)$ . From (4) it follows that there exists  $\varepsilon > 0$  verifying  $f(z) \in f(x) - C(\varepsilon)$ . As  $f(x_n) \rightarrow y$  we deduce that there exists  $n_0 \in \mathbb{N}$  such that

$$f(z) + y - f(x_n) \in f(x) - C(\varepsilon), \forall n \geq n_0.$$

As  $\varepsilon_n \downarrow 0$  it follows from Lemma 3.1(ii) that there exists  $n_1 \geq n_0$  such that

$$f(z) \in f(x_n) - C(\varepsilon_n), \forall n \geq n_1$$

and therefore  $f(z) = f(x_n)$ ,  $\forall n \geq n_1$ , since  $x_n \in \text{AE}(f, S, C, \varepsilon_n)$ . Then, taking the limit, we have  $f(z) = y = f(x)$ ,

$$(f(x) - \text{int}(C)(0)) \cap f(S) \subset \{f(x)\}$$

and  $x \in \text{WAE}(f, S, C, \varepsilon)$ .

Part (v). Consider  $x \in f^{-1}(K) \cap S$ . As  $f(x) \in K$  and  $x_n \in \text{AE}(f, S, C, \varepsilon_n)$  we have that

$$f(x) \in (f(x_n) - C(\varepsilon_n)) \cap f(S) \subset \{f(x_n)\}, \forall n$$

and we deduce that  $f(x) = f(x_n)$ ,  $\forall n$ . Therefore,

$$\begin{aligned} (f(x) - C(\varepsilon_n)) \cap f(S) &= (f(x_n) - C(\varepsilon_n)) \cap f(S) \\ &\subset \{f(x_n)\} = \{f(x)\}, \forall n. \end{aligned}$$

Thus, by (2) we see that

$$(f(x) - C(0)) \cap f(S) \subset \{f(x)\}$$

and we conclude that  $x \in \text{AE}(f, S, C, 0)$ . ■

*Remark 3.6:* From Theorem 3.5(iv) it is clear that if  $f$  is a continuous map at  $x_0 \in S$  and there exist  $(x_n) \subset S$  and  $(\varepsilon_n) \subset \mathbb{R}_+$  such that  $x_n \in \text{AE}(f, S, C, \varepsilon_n)$ ,  $x_n \rightarrow x_0$  and  $\varepsilon_n \downarrow 0$  then  $x_0 \in \text{WAE}(f, S, C, 0)$ .

*Remark 3.7:* Theorem 3.5 holds if we change  $C$  by  $\text{int}(C)$  and  $\text{AE}(f, S, C, \varepsilon)$  by  $\text{WAE}(f, S, C, \varepsilon)$ , since  $\text{int}(C)$  is also a nonempty solid pointed convex co-radiant set. In this case, let us observe that  $\text{WAE}(f, S, \text{int}(C), 0) = \text{WAE}(f, S, C, 0)$ .

#### IV. RELATIONS WITH OTHER $\varepsilon$ -EFFICIENCY CONCEPTS

In this section, we deduce well-known  $\varepsilon$ -efficiency concepts by choosing suitable sets  $C$  in Definition 3.2.

##### A. $\varepsilon$ -efficiency in the senses of Kutateladze and Németh

Suppose that  $0 \in D$  and take  $q \in D \setminus \{0\}$ . It is clear that

$$C := q + D$$

is a solid pointed convex co-radiant set, since  $\text{int}(C) = q + \text{int}(D)$ ,  $C \subset D$  and

$$\alpha C = q + ((\alpha - 1)q + \alpha D) \subset q + D = C, \forall \alpha > 1.$$

Moreover,  $C(\varepsilon) = \varepsilon q + D$ ,  $\forall \varepsilon > 0$ . Then, we can consider the set of  $\varepsilon$ -efficient solutions of (P) with respect to  $C$  and for each  $\varepsilon > 0$  we have

$$x \in \text{AE}(f, S, C, \varepsilon)$$

$$\iff x \in S, (f(x) - \varepsilon q - D) \cap f(S) \subset \{f(x)\}. \quad (6)$$

As  $D$  is a pointed cone, it follows that (6) is equivalent to

$$x \in \text{AE}(f, S, C, \varepsilon)$$

$$\iff x \in S, (f(x) - \varepsilon q - D) \cap f(S) = \emptyset. \quad (7)$$

This concept was introduced by Kutateladze [14] and it is the most popular notion of  $\varepsilon$ -efficiency (see [9], [14], [24], [29], [33] for more details about it). We denote the set of  $\varepsilon$ -efficient solutions of (P) in this sense by  $\text{AEKu}(f, S, D, q, \varepsilon)$ .

*Example 4.1:* Let us consider problem (P) introduced in Example 2.2. Taking  $q = (1, 0)$  and  $\varepsilon > 0$  we obtain that

$$\begin{aligned} \text{AEKu}(f, S, \mathbb{R}_+^2, q, \varepsilon) &= \{(x, y) \in \mathbb{R}^2 : 1 \leq x < 1 + \varepsilon, y \geq 2\} \\ &\cup \{(x, y) \in \mathbb{R}^2 : 2 \leq x < 2 + \varepsilon, 1 \leq y < 2\}. \end{aligned}$$

In Figure 3, this  $\varepsilon$ -efficiency set is drawn for  $\varepsilon = 0.5$  and  $\varepsilon = 0.1$ .

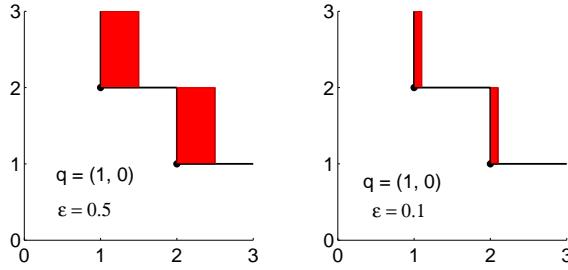


Fig. 3. Kutateladze's  $\varepsilon$ -efficiency sets in Example 2.2

Next, let us consider

$$C := H + D,$$

where  $H \subset D \setminus \{0\}$  is a nonempty  $D$ -convex set, i.e., such that  $H + D$  is a convex set. This set  $C$  becomes the previous one considering  $H = \{q\}$ .  $C$  is a pointed convex set, since  $C \subset D$  and  $D$  is a pointed cone. Moreover, as

$$C = \bigcup_{q \in H} (q + D)$$

and  $q + D$  is a solid co-radiant set,  $\forall q \in H$ , then  $C$  is a solid co-radiant set,

$$C(\varepsilon) = \bigcup_{q \in H} \varepsilon(q + D) = \bigcup_{q \in H} (\varepsilon q + D) = \varepsilon H + D, \quad \forall \varepsilon > 0 \quad (8)$$

and an  $\varepsilon$ -efficiency notion can be deduced from Definition 3.2 by taking  $C = H + D$ . With this notion, for each  $\varepsilon > 0$  the following  $\varepsilon$ -efficiency set is obtained:

$$\begin{aligned} & x \in \text{AE}(f, S, C, \varepsilon) \\ \iff & x \in S, (f(x) - \varepsilon H - D) \cap f(S) \subset \{f(x)\}. \end{aligned} \quad (9)$$

As  $D$  is a pointed cone, for each  $\varepsilon > 0$  condition (9) becomes

$$\begin{aligned} & x \in \text{AE}(f, S, C, \varepsilon) \\ \iff & x \in S, (f(x) - \varepsilon H - D) \cap f(S) = \emptyset. \end{aligned}$$

This notion was introduced by Németh [21]. We denote the set of  $\varepsilon$ -efficiency (resp. weak  $\varepsilon$ -efficiency) in the sense of Németh by  $\text{AENe}(f, S, D, H, \varepsilon)$  (resp.  $\text{WAENe}(f, S, D, H, \varepsilon)$ ).

Some properties of this kind of approximate solutions are collected in Corollary 4.4. The following lemma is necessary.

**Lemma 4.2:**

- (i)  $H \subset \text{int}(D) \iff C(0) = \text{int}(D)$ .
- (ii)  $\text{bd}(D) \cap (D \setminus \{0\}) \subset \text{cone}(H) \Rightarrow C(0) = D \setminus \{0\}$ .

(iii)  $\text{int}(C)(0) = \text{int}(D)$ .

*Proof.* Part (i). Suppose that  $H \subset \text{int}(D)$ . As  $D$  is a solid convex cone, from (8) we have

$$C(\varepsilon) = \varepsilon H + D \subset \varepsilon \text{int}(D) + D \subset \text{int}(D), \quad \forall \varepsilon > 0,$$

and  $C(0) \subset \text{int}(D)$ . Reciprocally, let  $d \in \text{int}(D)$  and consider  $q \in H$ . Then, there exists  $\varepsilon > 0$  such that  $d - \varepsilon q \in D$ . It follows that  $d \in \varepsilon q + D \subset \varepsilon H + D$  and

$$\text{int}(D) \subset \bigcup_{\varepsilon > 0} C(\varepsilon) = C(0).$$

Next, suppose  $C(0) = \text{int}(D)$ . Then, taking  $\varepsilon = 1$  in (8) we deduce that  $H \subset H + D \subset C(0) = \text{int}(D)$ .

Part (ii). If  $\text{bd}(D) \cap (D \setminus \{0\}) = \emptyset$  then  $D \setminus \{0\}$  is an open set and  $H \subset \text{int}(D)$ . Thus, by part (i), we have that  $C(0) = \text{int}(D) = D \setminus \{0\}$ .

Suppose that  $\text{bd}(D) \cap (D \setminus \{0\}) \neq \emptyset$ . From (8) we see that  $C(0) = \bigcup_{\varepsilon > 0} C(\varepsilon) \subset D \setminus \{0\}$ , since  $D$  is a pointed convex cone.

Conversely, by the hypothesis,

$$\text{bd}(D) \cap (D \setminus \{0\}) \subset \text{cone}(H) = \bigcup_{\alpha > 0} \alpha H \subset C(0). \quad (10)$$

Let  $d \in \text{int}(D)$  and take a point  $d_1 \in \text{bd}(D) \cap (D \setminus \{0\})$ . There exist  $d_2 \in D$  and  $\lambda \in (0, 1)$  such that  $d = \lambda d_1 + (1 - \lambda)d_2$ . From (10) we deduce that there exists  $q \in H$  and  $\alpha > 0$  such that  $d_1 = \alpha q$  and so

$$d = \lambda(\alpha q) + (1 - \lambda)d_2 \in \lambda\alpha H + D \subset C(0).$$

Thus,  $\text{int}(D) \subset C(0)$ . From this inclusion and (10) it follows that  $D \setminus \{0\} \subset C(0)$ .

Part (iii). As  $H \subset D$  and  $D$  is a solid convex cone then

$$\text{int}(C)(0) = \bigcup_{\varepsilon > 0} \varepsilon \text{int}(H + D) \subset \text{int}(D).$$

Let  $d \in \text{int}(D)$ . Taking a point  $q \in H$  we deduce that there exists  $\varepsilon > 0$  such that  $d - \varepsilon q \in \text{int}(D)$ . Therefore,  $d \in \text{int}(\varepsilon q + D) \subset \varepsilon \text{int}(H + D)$  and we conclude that  $\text{int}(D) \subset \text{int}(C)(0)$ . ■

*Remark 4.3:* The converse of Lemma 4.2(ii) is false as the following data show:  $p = 3$ ,  $H = \{(\alpha, 1 - \alpha, 0) : \alpha \in [0, 1]\}$  and  $D = \mathbb{R}_+^3 \setminus \{(0, 0, y_3) : y_3 > 0\}$ .

*Proposition 4.4:*

- (i) If  $H \subset \text{int}(D)$  then

$$\bigcap_{\varepsilon > 0} \text{AENe}(f, S, D, H, \varepsilon) = \text{WE}(f, S, D),$$

and if  $\text{bd}(D) \cap (D \setminus \{0\}) \subset \text{cone}(H)$  then

$$\bigcap_{\varepsilon > 0} \text{AENe}(f, S, D, H, \varepsilon) = \text{E}(f, S, D).$$

- (ii) Let  $(x_n) \subset S$ ,  $(\varepsilon_n) \subset \mathbb{R}_+$  and  $y \in \mathbb{R}^p$  such that  $x_n \in \text{AENe}(f, S, D, H, \varepsilon_n)$ ,  $\varepsilon_n \downarrow 0$  and  $f(x_n) \rightarrow y$ . Then  $f^{-1}(y) \cap S \subset \text{WE}(f, S, D)$ .
- (iii) Let  $(x_n) \subset S$  and  $(\varepsilon_n) \subset \mathbb{R}_+$  such that  $x_n \in \text{AENe}(f, S, D, H, \varepsilon_n)$  and  $\varepsilon_n \downarrow 0$ . Consider

$$K := \bigcap_n (f(x_n) - \varepsilon_n H - D).$$

If  $H \subset \text{int}(D)$  then  $f^{-1}(K) \cap S \subset \text{WE}(f, S, D)$  and if  $\text{bd}(D) \cap (D \setminus \{0\}) \subset \text{cone}(H)$  then  $f^{-1}(K) \cap S \subset \text{E}(f, S, D)$ .

*Proof.* If  $H \subset \text{int}(D)$ , then by Lemma 4.2(i) we deduce that  $C(0) = \text{int}(D)$  and  $\text{AENe}(f, S, D, H, 0) = \text{WE}(f, S, D)$ . From Lemma 4.2(iii), it follows that  $\text{int}(C)(0) = \text{int}(D)$  and we have  $\text{WAENe}(f, S, D, H, 0) = \text{WE}(f, S, D)$ . If  $\text{bd}(D) \cap (D \setminus \{0\}) \subset \text{cone}(H)$ , we deduce from Lemma 4.2(ii) that  $C(0) = D \setminus \{0\}$  and  $\text{AENe}(f, S, D, H, 0) = \text{E}(f, S, D)$ . Then properties (i)-(iii) hold by Theorem 3.5(iii)-(v). ■

In Proposition 4.4 we have extended several properties proved in the literature for the  $\varepsilon$ -efficiency set in the sense of Kutateladze (see for example [9, Lemma 3.3 and Theorem 3.4]) to the approximate solutions in the sense of Németh. We can deduce these properties by considering  $H = \{q\}$  in Proposition 4.4.

### B. $\varepsilon$ -efficiency in the sense of Helbig

Let  $h \in D^+ \setminus \{0\}$ . For each  $\alpha \in \mathbb{R}$  we denote

$$[h > \alpha] = \{y \in \mathbb{R}^p : \langle h, y \rangle > \alpha\}.$$

Consider the following set:

$$C := D \cap [h > 1]. \quad (11)$$

*Lemma 4.5:*

- (i)  $C$  is a solid pointed convex co-radiant set.
- (ii)  $C(\varepsilon) = D \cap [h > \varepsilon]$ .
- (iii)  $\text{int}(D) \subset C(0) \subset D \setminus \{0\}$ .
- (iv)  $\text{int}(C)(0) = \text{int}(D)$ , and if  $h \in D^{s+}$  then  $C(0) = D \setminus \{0\}$ .

*Proof.* Part (i). It is obvious that  $C$  is a pointed convex co-radiant set and we prove just that  $C$  is solid. Indeed, as  $D$  is a proper solid cone, there exists  $d \in \text{int}(D)$  such that  $\langle h, d \rangle = \alpha > 0$ . Then  $(2/\alpha)d \in \text{int}(D) \cap [h > 1] = \text{int}(C)$  and  $C$  is solid.

Part (ii) follows easily since  $D$  is a cone and  $\langle h, \cdot \rangle$  is a linear map.

Part (iii). Let  $d \in \text{int}(D)$ . As  $h \in D^+ \setminus \{0\}$  we see that  $\langle h, d \rangle > 0$ . Then, there exists  $\varepsilon > 0$  such that  $d \in [h > \varepsilon]$  and we deduce that  $d \in C(\varepsilon)$ . Thus, it follows that  $\text{int}(D) \subset C(0)$ .

By part (ii) it is obvious that  $C(\varepsilon) \subset D \setminus \{0\}$ ,  $\forall \varepsilon > 0$  and so  $C(0) \subset D \setminus \{0\}$ .

Part (iv). By part (iii) we deduce that  $\text{int}(D) = \text{int}(C(0))$  and by (5) we see that  $\text{int}(C(0)) = \text{int}(C)(0)$ . Therefore it follows that  $\text{int}(D) = \text{int}(C)(0)$ .

If  $h \in D^{s+}$  then  $\langle h, d \rangle > 0$ ,  $\forall d \in D \setminus \{0\}$  and we see that  $D \setminus \{0\} \subset C(0)$ . By part (iii) we have the converse inclusion and, therefore,  $C(0) = D \setminus \{0\}$ . ■

By Definition 3.2, for each  $\varepsilon \geq 0$  we obtain the following  $\varepsilon$ -efficiency set:

$$\begin{aligned} x_0 &\in \text{AE}(f, S, C, \varepsilon) \\ \iff x_0 &\in S, (f(x_0) - (D \cap [h > \varepsilon])) \cap f(S) = \emptyset \\ \iff x_0 &\in S, (f(x_0) - f(S)) \cap (D \cap [h > \varepsilon]) = \emptyset \\ \iff x_0 &\in S, \langle h, f(x_0) \rangle - \varepsilon \leq \langle h, f(x) \rangle, \\ &\forall x \in S, f(x) \in f(x_0) - D. \end{aligned}$$

This notion was introduced by Helbig [8]. We denote the set of  $\varepsilon$ -efficient (resp. weakly  $\varepsilon$ -efficient) solutions in this sense by  $\text{AEHe}(f, S, D, h, \varepsilon)$  (resp.  $\text{WAEHe}(f, S, D, h, \varepsilon)$ ).

*Example 4.6:* In the vector optimization problem considered in Example 2.2, let us take  $h(x, y) = 2x + y$ . Then, the following  $\varepsilon$ -efficiency set in the sense of Helbig is obtained:

$$\begin{aligned} \text{AEHe}(f, S, \mathbb{R}_+^2, h, \varepsilon) \\ = \{(x, y) \in S : 2x + y \leq 4 + \varepsilon, y \geq 2\} \\ \cup \{(x, y) \in S : 2x + y \leq 5 + \varepsilon, y < 2\}. \end{aligned}$$

In Figure 4 we drawn this set for  $\varepsilon = 1$  and  $\varepsilon = 0.5$ .

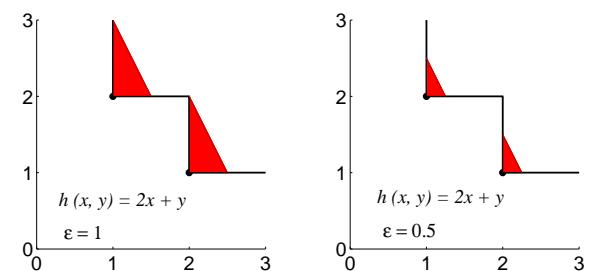


Fig. 4. Helbig's  $\varepsilon$ -efficiency sets in Example 2.2

Helbig's approximate solutions satisfy the following properties.

*Proposition 4.7:*

- (i) If  $h \in D^+ \setminus \{0\}$  then

$$\begin{aligned} \text{E}(f, S, D) &\subset \bigcap_{\varepsilon > 0} \text{AEHe}(f, S, D, h, \varepsilon) \\ &\subset \text{WE}(f, S, D), \end{aligned}$$

and if  $h \in D^{s+}$  then

$$\bigcap_{\varepsilon > 0} \text{AEHe}(f, S, D, h, \varepsilon) = \text{E}(f, S, D).$$

- (ii) Let  $(x_n) \subset S$ ,  $(\varepsilon_n) \subset \mathbb{R}_+$  and  $y \in \mathbb{R}^p$  such that  $x_n \in \text{AEHe}(f, S, D, h, \varepsilon)$ ,  $\varepsilon_n \downarrow 0$  and  $f(x_n) \rightarrow y$ . Then  $f^{-1}(y) \cap S \subset \text{WE}(f, S, D)$ .
- (iii) Let  $(x_n) \subset S$  and  $(\varepsilon_n) \subset \mathbb{R}_+$  such that  $x_n \in \text{AEHe}(f, S, D, h, \varepsilon_n)$  and  $\varepsilon_n \downarrow 0$ . Consider

$$K := \bigcap_n (f(x_n) - (D \cap [h > \varepsilon_n])).$$

If  $h \in D^+ \setminus \{0\}$  then  $f^{-1}(K) \cap S \subset \text{WE}(f, S, D)$  and if  $h \in D^{s+}$  then  $f^{-1}(K) \cap S \subset \text{E}(f, S, D)$ .

*Proof.* From Lemma 4.5(iii) we deduce that

$$\begin{aligned} \text{E}(f, S, D) &\subset \text{AEHe}(f, S, D, h, 0) \\ &\subset \text{WE}(f, S, D), \forall h \in D^+ \setminus \{0\}. \end{aligned}$$

Moreover, by Lemma 4.5(iv), we have that  $\text{AEHe}(f, S, D, h, 0) = \text{E}(f, S, D)$ ,  $\forall h \in D^{s+}$  and for each  $h \in D^+ \setminus \{0\}$  we see that  $\text{WAEHe}(f, S, D, h, 0) = \text{WE}(f, S, D)$ . Then, parts (i)-(iii) follow easily from Theorem 3.5(iii)-(v). ■

### C. $\varepsilon$ -efficiency in the sense of Tanaka

Next, we define the set

$$C := D \cap B^c,$$

where  $D \subset \mathbb{R}_+^p$ ,  $B$  denotes the unit open ball in  $\mathbb{R}^p$ ,

$$B = \{y \in \mathbb{R}^p : \|y\|_1 < 1\},$$

and  $\|\cdot\|_1$  is the  $l_1$  norm in  $\mathbb{R}^p$ .

*Lemma 4.8:*

- (i)  $C$  is a solid pointed convex co-radiant set.
- (ii)  $C(\varepsilon) = D \cap (\varepsilon B)^c$ ,  $\forall \varepsilon > 0$ .
- (iii)  $C(0) = D \setminus \{0\}$  and  $\text{int}(C)(0) = \text{int}(D)$ .

*Proof.* Part (i). It is obvious that  $C$  is convex. Moreover,  $C$  is a pointed set, since  $C \subset D$  and  $D$  is a pointed cone. Let  $y \in C$  and  $\alpha > 1$ . As  $D$  is a cone, we have that  $\alpha y \in D$ . Moreover,  $\|\alpha y\|_1 = \alpha \|y\|_1 > 1$  since  $\alpha > 1$  and  $y \notin B$ . Then  $\alpha y \in C$  and it follows that  $C$  is a co-radiant set.

Next, we prove that  $C$  is a solid set. Indeed, there exists a point  $q \in \text{int}(D)$ ,  $q \neq 0$ , since  $D$  is a solid cone. Then,  $2q/\|q\|_1 \in \text{int}(D) \cap \text{int}(B^c) = \text{int}(C)$  and  $C$  is a solid set.

Part (ii). Let  $y \in C$  and  $\varepsilon > 0$ . It is clear that  $\varepsilon y \in D$  and  $\|\varepsilon y\|_1 = \varepsilon \|y\|_1 \geq \varepsilon$  since  $y \in B^c$ . It follows that  $\varepsilon y \in D \cap (\varepsilon B)^c$  and  $C(\varepsilon) \subset D \cap (\varepsilon B)^c$ . Similarly, if  $y \in D \cap (\varepsilon B)^c$  then  $y/\varepsilon \in D \cap B^c = C$ . Thus,  $y \in \varepsilon C$  and  $C(\varepsilon) = D \cap (\varepsilon B)^c$ .

Part (iii). By part (ii) it is clear that

$$C(0) = \bigcup_{\varepsilon > 0} D \cap (\varepsilon B)^c = D \cap \left( \bigcap_{\varepsilon > 0} \varepsilon B \right)^c = D \setminus \{0\}.$$

Analogously,

$$\begin{aligned} \text{int}(C)(0) &= \bigcup_{\varepsilon > 0} \text{int}(D \cap (\varepsilon B)^c) \\ &= \bigcup_{\varepsilon > 0} \text{int}(D) \cap \text{int}((\varepsilon B)^c) = \text{int}(D) \cap \left( \bigcap_{\varepsilon > 0} \varepsilon \text{cl}(B) \right)^c \\ &= \text{int}(D). \blacksquare \end{aligned}$$

The set of all approximate solutions (resp. weak approximate solutions) with respect to this  $C$  is denoted by  $\text{AETa}(f, S, D, \varepsilon)$  (resp.  $\text{WAETa}(f, S, D, 0)$ ). It follows that

$$\begin{aligned} x \in \text{AETa}(f, S, D, \varepsilon) &\iff (f(x) - (D \cap (\varepsilon B)^c)) \cap f(S) \subset \{f(x)\} \\ &\iff (f(x) - D) \cap f(S) \subset f(x) + \varepsilon B. \end{aligned}$$

This concept of  $\varepsilon$ -efficiency was introduced by Tanaka [26]. Next, we give several properties of this notion, which extend others previously proved by Tanaka in [26, Proposition 3.3].

*Example 4.9:* Let us recall problem (P) considered in Example 2.2. In this problem, the set of approximate solutions in the sense of Tanaka is:

$$\begin{aligned} \text{AETa}(f, S, \mathbb{R}_+^2, \varepsilon) &= \{(x, y) \in S : x + y \leq 3 + \varepsilon\}. \end{aligned}$$

In Figure 5 we show this set for  $\varepsilon = 1$  and  $\varepsilon = 0.5$ .

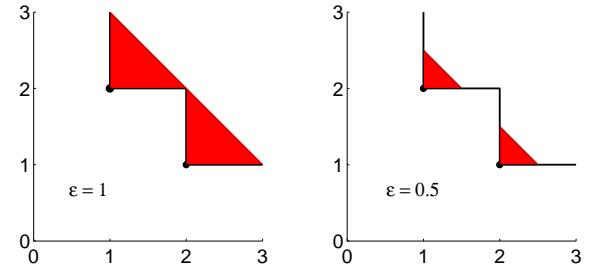


Fig. 5. Tanaka's  $\varepsilon$ -efficiency sets in Example 2.2

*Proposition 4.10:*

- (i)  $\bigcap_{\varepsilon > 0} \text{AETa}(f, S, D, \varepsilon) = \text{E}(f, S, D)$ .
- (ii) Let  $(x_n) \subset S$ ,  $(\varepsilon_n) \subset \mathbb{R}_+$  and  $y \in \mathbb{R}^p$  such that  $x_n \in \text{AETa}(f, S, D, \varepsilon_n)$ ,  $\varepsilon_n \downarrow 0$  and  $f(x_n) \rightarrow y$ . Then  $f^{-1}(y) \cap S \subset \text{WE}(f, S, D)$ .
- (iii) Let  $(x_n) \subset S$  and  $(\varepsilon_n) \subset \mathbb{R}_+$  such that  $x_n \in \text{AETa}(f, S, D, \varepsilon_n)$ ,  $\varepsilon_n \downarrow 0$  and

$$K := \bigcap_n (f(x_n) - (D \cap (\varepsilon_n B)^c)).$$

Then  $f^{-1}(K) \cap S \subset E(f, S, D)$ .

*Proof.* By Lemma 4.8(iii) we deduce that  $AETa(f, S, D, 0) = E(f, S, D)$  and  $WAETa(f, S, D, 0) = WE(f, S, D)$ . Then, the corollary is a consequence of Theorem 3.5(iii)-(v). ■

*Remark 4.11:* In order to better understanding Propositions 4.4, 4.7 and 4.10 let us notice that these results can be easily interpreted via Examples 4.1, 4.6 and 4.9 and Figures 3, 4 and 5, respectively.

## V. LINEAR SCALARIZATION FOR $\varepsilon$ -EFFICIENCY IN CONVEX MULTIOBJECTIVE MATHEMATICAL PROGRAMS

In the literature, approximate solutions of (P) are usually studied in convex problems via the Kutateladze's definition (see for example [29, Lemma 3.2], [2, Theorem 2.1], [18, Theorems 1 and 2], [17, Lemma 2.1] and [4, Theorem 2.1]). However, results about  $\varepsilon$ -efficiency notions different to the Kutateladze's concept are very limited (see [19, Propositions 3.1 and 3.2], [32, Lemmas 4.1 and 4.2] and [7, Lemma 3.1]).

Our objective in this section is to characterize the  $\varepsilon$ -efficiency notion with respect to a set  $C$ , in order to obtain additional results on  $\varepsilon$ -efficient concepts different to the Kutateladze's notion by applying this characterization to suitable sets  $C$ .

Next, necessary conditions for the approximate solutions of (P) with respect to a set  $C$  are obtained via approximate solutions of linear scalarizations, i.e., by the Weighting Method. For it, we suppose that program (P) satisfies certain convexity hypothesis.

*Definition 5.1:* ([12, Definition 2.9]) We say that  $f : S \subset \mathbb{R}^n \rightarrow \mathbb{R}^p$  is a convex-like map on  $S$  with respect to  $D$  if  $\forall x_1, x_2 \in S$  and  $\forall \lambda \in [0, 1]$  there is an  $s \in S$  such that

$$\lambda f(x_1) + (1 - \lambda)f(x_2) - f(s) \in D.$$

The following proposition is well-known (see for example [12, Theorem 2.11]).

*Proposition 5.2:* The map  $f$  is convex-like on  $S$  with respect to  $D$  if and only if  $f(S) + D$  is a convex set.

For each  $K \subset \mathbb{R}^p$  and  $y \in \mathbb{R}^p$  we denote  $d(y, K) = \inf\{\|y - z\| : z \in K\}$ , where  $\|\cdot\|$  is the Euclidean norm.

*Theorem 5.3:* Consider program (P) and suppose that  $f$  is convex-like on  $S$  with respect to  $C(0)$  and  $d(0, C) \leq \delta$ . Then,  $\forall \varepsilon \geq 0$ ,

$$AE(f, S, C, \varepsilon) \subset \bigcup_{l \in C(0)^+, \|l\|=1} AMin(l \cdot f, S, \varepsilon\delta),$$

where  $l \cdot f : \mathbb{R}^n \rightarrow \mathbb{R}$  is the map  $\langle l, f(\cdot) \rangle$ .

*Proof.* Let  $\varepsilon > 0$  and  $x_0 \in AE(f, S, C, \varepsilon)$ . Then  $x_0 \in S$  and

$$(f(x_0) - C(\varepsilon)) \cap f(S) \subset \{f(x_0)\}.$$

By Lemma 3.1(v) it follows that

$$(f(x_0) - C(\varepsilon)) \cap (f(S) + C(0)) \subset \{f(x_0)\}.$$

Moreover, as  $C$  is pointed we have that  $f(x_0) \notin \text{int}(f(x_0) - C(\varepsilon))$ , and by Proposition 5.2 we see that  $f(S) + C(0)$  is a convex set, since  $f$  is a convex-like map on  $S$  with respect to  $C(0)$ . Then, by the Separation Theorem (see, for example, [23, Theorem 2.1.1]) we deduce that there exists  $l \in \mathbb{R}^p \setminus \{0\}$  such that

$$\begin{aligned} \langle l, f(x_0) - \varepsilon d_1 \rangle &\leq \langle l, f(x) + d_2 \rangle, \\ \forall d_1 \in C, \forall d_2 \in C(0), \forall x \in S. \end{aligned} \quad (12)$$

We can suppose that  $\|l\| = 1$  since  $l \neq 0$ , and as  $C(0)$  is a cone, we have from (12) that  $l \in C(0)^+$ . By continuity, we deduce from (12) that

$$\langle l, f(x_0) \rangle - \varepsilon \langle l, d_1 \rangle \leq \langle l, f(x) \rangle, \forall d_1 \in C, \forall x \in S \quad (13)$$

and by the Cauchy-Schwartz inequality we deduce that

$$\langle l, f(x_0) \rangle - \varepsilon \|d_1\| \leq \langle l, f(x) \rangle, \forall d_1 \in C, \forall x \in S. \quad (14)$$

From here, by continuity, we see that

$$\langle l, f(x_0) \rangle - \varepsilon \delta \leq \langle l, f(x_0) \rangle - \varepsilon d(0, C) \quad (15)$$

$$\leq \langle l, f(x) \rangle, \forall x \in S. \quad (16)$$

Thus, for  $l(y) = \langle l, y \rangle$  it follows that  $l \in C(0)^+$ ,  $\|l\| = 1$  and  $x_0 \in AMin(l \cdot f, S, \varepsilon\delta)$ .

If  $\varepsilon = 0$  and  $x_0 \in AE(f, S, C, 0)$  then, repeating the same reasoning we see that

$$\langle l, f(x_0) \rangle \leq \langle l, f(x) \rangle, \forall x \in S, \quad (17)$$

since (12) is true for all  $\varepsilon > 0$ . Therefore, considering  $l(y) = \langle l, y \rangle$  it follows that  $x_0 \in AMin(l \cdot f, S, 0)$  with  $l \in C(0)^+$  and  $\|l\| = 1$ . ■

We obtain sufficient conditions for the  $\varepsilon$ -efficient solutions of (P) with respect to a set  $C$  as a consequence of the following result (see [1, Lemma 2.7] for more detail).

*Lemma 5.4:* Let  $K \subset \mathbb{R}^p$  be a convex cone such that  $K^+$  is solid. Consider  $l \in \text{int}(K^+)$ . Then

$$d(l, \mathbb{R}^p \setminus K^+) \leq \inf\{\langle l, y \rangle : y \in K, \|y\| = 1\}.$$

*Theorem 5.5:* Suppose that  $0 \notin \text{cl}(C)$ ,  $C(0)^+$  is solid and consider  $l \in \text{int}(C(0)^+)$ . Then

$$AMin(l \cdot f, S, \delta) \subset AE(f, S, C, \varepsilon), \forall \varepsilon > \delta/c,$$

where  $c = d(0, C) \cdot d(l, \mathbb{R}^p \setminus C(0)^+)$ .

*Proof.* Let  $x_0 \in AMin(l \cdot f, S, \delta)$  and  $\varepsilon > \delta/c$ . Reasoning "ad absurdum", let us suppose that  $x_0 \notin AE(f, S, C, \varepsilon)$ . Then there exist  $x \in S$  and  $d \in C(\varepsilon)$  such that  $f(x_0) - d = f(x)$ . As  $\langle l, \cdot \rangle$  is linear and  $x_0 \in AMin(l \cdot f, S, \delta)$  we deduce that

$$\langle l, f(x) \rangle = \langle l, f(x_0) \rangle - \langle l, d \rangle \leq \langle l, f(x) \rangle + \delta - \langle l, d \rangle$$

and it follows that  $\langle l, d \rangle \leq \delta$ . By Lemma 5.4 we have that

$$\begin{aligned} \langle l, d \rangle &\geq \|d\|d(l, \mathbb{R}^p \setminus C(0)^+) \\ &\geq \varepsilon d(0, C)d(l, \mathbb{R}^p \setminus C(0)^+) = \varepsilon c > \delta, \end{aligned}$$

which is a contradiction. Therefore,  $x_0 \in \text{AE}(f, S, C, \varepsilon)$ .  $\blacksquare$

## VI. LINEAR SCALARIZATION FOR NÉMETH, HELBIG AND TANAKA'S $\varepsilon$ -EFFICIENCY NOTIONS

In this section we consider that (P) is a Pareto multiobjective mathematical program (i.e.,  $D = \mathbb{R}_+^p$ ) and under the hypothesis that the objective function  $f : S \subset \mathbb{R}^n \rightarrow \mathbb{R}^p$  is convex-like on  $S$  we apply Theorem 5.3 and Theorem 5.5 to obtain through linear scalarization necessary and sufficient conditions for the  $\varepsilon$ -efficient solutions of (P) in the senses of Németh, Helbig and Tanaka.

We denote the components of a vector  $y \in \mathbb{R}^p$  by  $y_i$  and we define the following nonempty solid pointed convex co-radiant sets:

$$C_{\text{Ne}} = H + \mathbb{R}_+^p, \quad C_{\text{He}} = \mathbb{R}_+^p \cap [h > 1], \quad C_{\text{Ta}} = \mathbb{R}_+^p \cap B^c,$$

where  $h \in \text{int}(\mathbb{R}_+^p)$  and

$$H = \left\{ y \in \mathbb{R}_+^p : \sum_{i=1}^p (y_i - 1)^2 < 1 \right\}.$$

**Theorem 6.1:** Consider program (P) and suppose that  $f$  is a convex-like map on  $S$  with respect to  $\mathbb{R}_+^p \setminus \{0\}$ . Then,  $\forall \varepsilon \geq 0$ ,

$$\text{AEHe}(f, S, \mathbb{R}_+^p, h, \varepsilon) \subset \bigcup_{l \in \mathbb{R}_+^p, \|l\|=1} \text{AMin}(l \cdot f, S, \varepsilon/\|h\|), \quad (18)$$

$$\text{AETa}(f, S, \mathbb{R}_+^p, \varepsilon) \subset \bigcup_{l \in \mathbb{R}_+^p, \|l\|=1} \text{AMin}(l \cdot f, S, \varepsilon/\sqrt{p}) \quad (19)$$

and for each  $l \in \text{int}(\mathbb{R}_+^p)$ ,

$$\begin{aligned} \text{AMin}(l \cdot f, S, \delta) &\subset \text{AEHe}(f, S, \mathbb{R}_+^p, h, \varepsilon), \\ \forall \varepsilon &> \|h\| \delta / \min_{1 \leq i \leq p} \{l_i\}, \end{aligned} \quad (20)$$

$$\begin{aligned} \text{AMin}(l \cdot f, S, \delta) &\subset \text{AETa}(f, S, \mathbb{R}_+^p, \varepsilon), \\ \forall \varepsilon &> \sqrt{p} \delta / \min_{1 \leq i \leq p} \{l_i\}. \end{aligned} \quad (21)$$

**Proof.** By Lemma 4.5(iv) and Lemma 4.8(iii) we have that

$$C_{\text{He}}(0) = C_{\text{Ta}}(0) = \mathbb{R}_+^p \setminus \{0\}. \quad (22)$$

Moreover, easy calculations give

$$d(0, C_{\text{He}}) = 1/\|h\|, \quad (23)$$

$$d(0, C_{\text{Ta}}) = 1/\sqrt{p}. \quad (24)$$

Then, necessary conditions (18)-(19) are a direct consequence of Theorem 5.3 and distances (23)-(24), and sufficient conditions (20)-(21) are deduced immediately from Theorem 5.5 and formulae (23)-(24).  $\blacksquare$

**Theorem 6.2:** Consider program (P) and suppose that  $f$  is a convex-like map on  $S$  with respect to  $\text{int}(\mathbb{R}_+^p)$ . Then,  $\forall \varepsilon \geq 0$ ,

$$\begin{aligned} \text{AENe}(f, S, \mathbb{R}_+^p, H, \varepsilon) \\ \subset \bigcup_{l \in \mathbb{R}_+^p, \|l\|=1} \text{AMin}(l \cdot f, S, \varepsilon(\sqrt{p} - 1)), \end{aligned} \quad (25)$$

and for each  $l \in \text{int}(\mathbb{R}_+^p)$ ,

$$\begin{aligned} \text{AMin}(l \cdot f, S, \delta) &\subset \text{AENe}(f, S, \mathbb{R}_+^p, H, \varepsilon), \\ \forall \varepsilon &> \delta / ((\sqrt{p} - 1) \min_{1 \leq i \leq p} \{l_i\}), \end{aligned} \quad (26)$$

**Proof.** From Lemma 4.2(i) it is clear that  $C_{\text{Ne}}(0) = \text{int}(\mathbb{R}_+^p)$ . Moreover,  $d(0, C_{\text{Ne}}) = \sqrt{p} - 1$ . Then (25) and (26) follow by Theorem 5.3 and Theorem 5.5 respectively.  $\blacksquare$

## VII. CONCLUSIONS

In this paper we have introduced a new  $\varepsilon$ -efficiency concept in multiobjective mathematical programs, which extends and unifies several different  $\varepsilon$ -efficient notions previously defined in the literature.

We prove some properties of this new concept and we characterize it in a convex framework via approximate solutions of associate scalar optimization problems.

As final conclusion, we think that several results of this work are useful in order to improve the actual resolution techniques and develop new methods to solve multiobjective mathematical programs.

## ACKNOWLEDGEMENTS

This research was partially supported by Ministerio de Ciencia y Tecnología (Spain), project BFM2003-02194.

## REFERENCES

- [1] S. Bolintineanu, "Vector variational principles;  $\varepsilon$ -efficiency and scalar stationarity," *J. Convex Anal.*, vol. 8, no. 1, pp. 71–85, 2001.
- [2] S. Deng, "On approximate solutions in convex vector optimization," *SIAM J. Control Optim.*, vol. 35, no. 6, pp. 2128–2136, 1997.
- [3] D. Dentcheva and S. Helbig, "On variational principles, level sets, well-posedness, and  $\varepsilon$ -solutions in vector optimization," *J. Optim. Theory Appl.*, vol. 89, no. 2, pp. 325–349, 1996.
- [4] J. Dutta and V. Vetrivel, "On approximate minima in vector optimization," *Numer. Funct. Anal. Optim.*, vol. 22, no. 7&8, pp. 845–859, 2001.
- [5] C. Gutiérrez, "Condiciones de  $\varepsilon$ -Eficiencia en Optimización Vectorial," Ph.D. dissertation, Universidad Nacional de Educación a Distancia, Madrid, November 2004.

- [6] C. Gutiérrez, B. Jiménez, and V. Novo, “A property of efficient and  $\varepsilon$ -efficient solutions in vector optimization,” *Appl. Math. Lett.*, vol. 18, no. 4, pp. 409–414, 2005.
- [7] ——, “Multiplier rules and saddle-point theorems for Helbig’s approximate solutions in convex Pareto problems,” *J. Global Optim.*, vol. 32, no. 3, 2005.
- [8] S. Helbig, “On a new concept for  $\varepsilon$ -efficiency,” talk at *Optimization Days 1992*, Montreal, 1992.
- [9] S. Helbig and D. Pateva, “On several concepts for  $\varepsilon$ -efficiency,” *OR Spektrum*, vol. 16, pp. 179–186, 1994.
- [10] H. Idrissi, P. Loridan, and C. Michelot, “Approximation of solutions for location problems,” *J. Optim. Theory Appl.*, vol. 56, no. 1, pp. 127–143, 1988.
- [11] G. Isac, “The Ekeland’s principle and the Pareto  $\varepsilon$ -efficiency,” ser. Lecture Notes in Economics and Mathematical Systems. New York, Springer, 1996, vol. 432, pp. 148–163.
- [12] J. Jahn, *Vector Optimization*, Berlin: Springer, 2004.
- [13] B. Jiménez and V. Novo, “Second order necessary conditions in set constrained differentiable vector optimization,” *Math. Methods Oper. Res.*, vol. 58, no. 2, pp. 299–317, 2003.
- [14] S. S. Kutateladze, “Convex  $\varepsilon$ -programming,” *Soviet Math. Dokl.*, vol. 20, no. 2, pp. 391–393, 1979.
- [15] J. C. Liu, “ $\varepsilon$ -duality theorem of nondifferentiable nonconvex multiobjective programming,” *J. Optim. Theory Appl.*, vol. 69, no. 1, pp. 153–167, 1991.
- [16] ——, “ $\varepsilon$ -Pareto optimality for nondifferentiable multiobjective programming via penalty function,” *J. Math. Anal. Appl.*, vol. 198, pp. 248–261, 1996.
- [17] ——, “ $\varepsilon$ -properly efficient solutions to nondifferentiable multiobjective programming problems,” *Appl. Math. Lett.*, vol. 12, pp. 109–113, 1999.
- [18] J. C. Liu and K. Yokoyama, “ $\varepsilon$ -optimality and duality for multiobjective fractional programming,” *Comput. Math. Appl.*, vol. 37, pp. 119–128, 1999.
- [19] P. Loridan, “ $\varepsilon$ -solutions in vector minimization problems,” *J. Optim. Theory Appl.*, vol. 43, no. 2, pp. 265–276, 1984.
- [20] ——, “ $\varepsilon$ -duality theorem of nondifferentiable nonconvex multiobjective programming,” *J. Optim. Theory Appl.*, vol. 74, no. 3, pp. 565–566, 1992.
- [21] A. B. Németh, “A nonconvex vector minimization problem,” *Nonlinear Anal.*, vol. 10, no. 7, pp. 669–678, 1986.
- [22] L. G. Ruhe and G. B. Fruhwirth, “ $\varepsilon$ -optimality for bicriteria programs and its application to minimum cost flows,” *Computing*, vol. 44, pp. 21–34, 1990.
- [23] Y. Sawaragi, H. Nakayama and T. Tanino, *Theory of Multiobjective Optimization*, Orlando: Academic Press, 1985.
- [24] T. Staib, “On two generalizations of Pareto minimality,” *J. Optim. Theory Appl.*, vol. 59, no. 2, pp. 289–306, 1988.
- [25] C. Tammer, “A generalization of Ekeland’s variational principle,” *Optimization*, vol. 25, pp. 129–141, 1992.
- [26] T. Tanaka, “A new approach to approximation of solutions in vector optimization problems,” in *Proc. APORS’94*, Singapore, July 1994, pp. 497–504.
- [27] I. Vályi, “Approximate solutions of vector optimization problems,” ser. Systems Analysis and Simulation. Berlin, Germany: Akademie-Verlag, 1985, pp. 246–250.
- [28] ——, “Approximate saddle-point theorems in vector optimization,” *J. Optim. Theory Appl.*, vol. 55, no. 3, pp. 435–448, 1987.
- [29] D. J. White, “Epsilon efficiency,” *J. Optim. Theory Appl.*, vol. 49, no. 2, pp. 319–337, 1986.
- [30] ——, “Epsilon-dominating solutions in mean-variance portfolio analysis,” *European J. Oper. Res.*, vol. 105, pp. 457–466, 1998.
- [31] A. P. Wierzbicki, “On the completeness and constructiveness of parametric characterizations to vector optimization problems,” *OR Spektrum*, vol. 8, pp. 73–87, 1986.
- [32] K. Yokoyama, “Epsilon approximate solutions for multiobjective programming problems,” *J. Math. Anal. Appl.*, vol. 203, pp. 142–149, 1996.
- [33] ——, “Relationships between efficient set and  $\varepsilon$ -efficient set.” in *Proc. International Conference on Nonlinear Analysis and Convex Analysis*, River Edge, New York, 1999, pp. 376–380.

# An Efficient Approach for Solving the Production/Ordering Planning Problem with Time-varying Storage Capacities

José Miguel Gutiérrez\*, Antonio Sedeño-Noda<sup>†</sup>, Marcos Colebrook<sup>‡</sup> and Joaquín Sicilia<sup>§</sup>

University of La Laguna/Dept.de Estadística, Investigación Operativa y Computación

Email: \*jmgrrez@ull.es, <sup>†</sup>asedeno@ull.es, <sup>‡</sup>mcolesan@ull.es, <sup>§</sup>jsicilia@ull.es

**Abstract**—We address the dynamic lot size problem assuming time-varying storage capacities. The planning horizon is divided into  $T$  periods and stockouts are not allowed. Moreover, for each period, we consider a setup cost, a holding unit cost and a production/ordering unit cost, which can vary through the planning horizon. Although this model can be solved using  $O(T^3)$  algorithms, we show that under this cost structure an optimal solution can be obtained in  $O(T \log T)$  time. In addition, we show that when production/ordering unit costs are assumed to be constant, there exists an optimal plan satisfying the Zero Inventory Ordering (ZIO) property. Computational results for randomly generated problems are reported.

**Keywords**—Inventory-Production: Policies, Capacity, Dynamic Programming, Computational Improvement.

## I. INTRODUCTION

WE focus our attention on the dynamic lot size problem assuming that inventory levels are limited by storage capacities. Holding and production/ordering costs are linear and setup costs are also considered. As usual, the planning horizon is divided into  $T$  periods and the demand for each period is known in advance. In addition, we assume that the storage capacity varies with time, and shortages are not permitted. The goal consists of determining a production plan which satisfies the demand for each period at minimum cost.

The uncapacitated version of this problem, that is, when no bound is imposed on either the production quantity or the inventory level, was well solved by Manne (1958) and, independently, by Wagner and Whitin (1958), who devised an  $O(T^2)$  dynamic programming algorithm. Later, Federgruen and Tzur (1991), Wagelmans et al. (1992), as well as Aggarwal and Park (1993) independently proposed  $O(T)$  algorithms, based on different techniques, for the cost structure defined in Wagner and Whitin (1958), and  $O(T \log T)$  procedures for the general version, that is, considering non-constant unit production costs.

Love (1973) was the first to study the dynamic lot size problem with limited inventory and concave cost functions, developing an  $O(T^3)$  algorithm based on the dynamic programming approach. Since then, little effort has been made either to look for more efficient algorithms or to analyse different cost structures for this problem. Accordingly, Gutiérrez et al. (2003) have recently proposed a  $O(T^3)$  dynamic programming algorithm, which reduces the computational effort of Love's procedure to a constant factor. That algorithm runs in  $O(T)$  expected time when the demand in each period varies between zero and the storage capacity for such period. These computational enhancements are a consequence of a new characterization of the optimal plans introduced in their paper. On the other hand, when the cost structure consists of linear holding and production costs and in absence of setup costs, Sedeño-Noda et al. (2004) proved that an optimal plan can be derived applying an  $O(T \log T)$  algorithm. In this paper, the case where holding and production costs are linear and setup costs are significant is analyzed. Indeed, the results show that, when shortages are not allowed, we can adapt the geometrical technique given in Wagelmans et al. (1992) to determine an optimal plan in  $O(T \log T)$ . Moreover, we also consider the cost structure given in Wagner and Whitin (1958), namely, we assume that the production cost remains constant through the planning horizon. Under this assumption, we get an interesting result, which states that among the optimal policies there exists one satisfying the Zero Inventory Ordering (ZIO) property.

The rest of the paper is organized as follows. In Section 2, we introduce the problem statement and the notation. In Section 3, we present a recurrence formula which can be implemented with an  $O(T^2)$  dynamic programming algorithm. Moreover, we propose an adaptation of the geometrical technique of Wagelmans et al. (1992), which allows us to develop a  $O(T \log T)$  solution method. Additionally, when production costs are

constant, i.e., considering the cost structure as in Wagner and Whitin (1958), we show that an optimal solution satisfying the ZIO property can be determined. Furthermore, a numerical example to illustrate these results is reported. A computational experiment is presented in Section 4. Finally, concluding remarks are discussed in Section 5.

## II. NOTATION AND PRELIMINARIES

Let  $T$  be the number of consecutive periods. Then, following the notation introduced in Wagelmans et al. (1992), we denote by  $d_t$ ,  $p_t$ ,  $h_t$  and  $f_t$ , the demand, production unit cost, holding unit cost and setup cost in period  $t$  ( $t = 1, \dots, T$ ). We also denote by  $S_t$  the storage capacity in period  $t$ . For the sake of simplicity, we denote by  $d_{t,j} = \sum_{i=t}^j d_i$  the cumulative demand from period  $t$  to period  $j$ . Remark that, once the cumulative demands  $d_{1,T}, d_{2,T}, \dots, d_{T,T}$  are obtained in  $O(T)$ , any value  $d_{t,j}$  can be determined in  $O(1)$  applying  $d_{t,j} = d_{t,T} - d_{j+1,T}$ . In addition to the previous parameters, let  $x_t$  and  $I_t$  ( $t = 1, \dots, T$ ) be the decision and state variables, respectively, which represent the number of units produced in period  $t$  and the inventory level at the end of period  $t$ . Also, the statement of the cost function requires the following variable related to setup costs:  $y_t = 1$  if  $x_t > 0$ , and  $y_t = 0$  otherwise. Then, we can formulate the dynamic lot size problem with storage capacities as follows:

$$\begin{aligned} & \min \sum_{t=1}^T (f_t y_t + p_t x_t + h_t I_t) \\ & \text{s.t.} \\ & I_0 = I_T = 0 \\ & x_t + I_{t-1} - I_t = d_t \quad t = 1, \dots, T \\ & d_{t,T} y_t - x_t \geq 0 \quad t = 1, \dots, T \\ & 0 \leq I_t \leq S_t - d_t \quad t = 1, \dots, T \\ & x_t \geq 0 \text{ integer, } y_t \in \{0, 1\} \quad t = 1, \dots, T \end{aligned} \quad (1)$$

The first constraint in (1) forces both the initial and final inventory levels of the planning horizon to be zero. The second constraint set in (1) represents the well-known material balance equation. Since  $y_t \in \{0, 1\}$ , the following constraint set ensures that  $y_t = 1$  when  $x_t > 0$ . The next constraint set states the lower and upper bounds for each inventory level and it can be obtained combining the second set of constraints and the following set:  $I_{t-1} + x_t \leq S_t$ , ( $t = 1, \dots, T$ ). This last set guarantees that the sum of the inventory at the end of one period and the production quantity in the consecutive period does not exceed the storage capacity of this last period. The final constraints force the production quantities to be non-negative integers, and

each  $y_t$  to be binary variable. Feasibility is assured by the assumption that  $d_t \leq S_t$  ( $t = 1, \dots, T$ ).

Remark that, as a consequence of the storage constraints, the maximum quantity which can be produced in a period is limited. Accordingly, let  $M_t$  be the maximum quantity that can be produced in period  $t$  ( $t = 1, \dots, T-1$ ), which can be easily derived from the following expression:  $M_t = \min(M_{t+1} + d_t, S_t)$ , where  $M_T = d_T$ . We also denote by  $R_t$  the maximum reachable period such that the demands of periods  $i = t, \dots, R_t$  can be completely satisfied with inventory held from period  $t$  ( $t = 1, \dots, T-1$ ), that is,  $R_t = \max(j : t \leq j \leq T \text{ and } (M_t - d_{t,j}) \geq 0)$ , with  $R_T = T$ . The values  $M_t$  and  $R_t$  ( $t = 1, \dots, T-1$ ) are determined from the demand and storage capacity values in  $O(T)$ .

We introduce in the next section the solution method which determines an optimal plan for problem (1) in  $O(T \log T)$  time.

## III. SOLUTION METHOD

As we previously pointed out, we address the case in which the production and holding costs are linear and setup costs are significant. This cost structure corresponds with a particular case of the concave cost structure studied by Love (1973) and Gutiérrez et al. (2003), respectively. Precisely, Theorem 1 in Gutiérrez et al. (2003) states that an optimal production plan  $\mathbf{x} = (x_1, \dots, x_T)$  can always be found in  $O(T^3)$  such that for each production period  $t$  ( $t = 1, \dots, T$ ), the amount  $I_{t-1} + x_t$  corresponds to either a sum of demands of consecutive periods  $d_{t,j} \leq M_t$  ( $j = t+1, \dots, R_t$ ) or the maximum quantity  $M_t$  that can be produced in that period. Formally, this theorem states that if  $x_t > 0$  then

$$I_{t-1} + x_t = \begin{cases} M_t, \\ \text{or} \\ d_{t,j} \quad \text{for some } j \text{ such that } d_{t,j} \leq M_t \end{cases} \quad (2)$$

Moreover, Theorem 2 in Gutiérrez et al. (2003) states that among the optimal solutions for the problem there exists at least one such that if  $I_{t-1} + x_t = d_{t,j} < M_t$  for some period  $t$ , then  $x_i = 0$  for those periods  $i = t+1, \dots, j$ . Accordingly, this result guarantees the existence of one optimal plan in which the decision in a period  $i$  is not to produce ( $x_i = 0$ ) when the inventory on hand at the end of the previous period consists of a sum of demands, i.e.,  $I_{i-1} = d_{i,j}$ . In other words, that property can be formulated as follows

If  $I_{i-1} = d_{i,j}$  for some  $j$  such that  $d_{i,j} \leq M_i$ , then  $x_i = 0$

$$(3) \quad \text{For a contradiction, let us suppose that } I_{j,n-1}^* + x_{j,n}^* < M_t - d_{t,n-1} \text{ for all } n \in [j, R_t + 1] \text{ and } j \in [t + 1, R_t + 1].$$

Throughout this section we are only interested in generating optimal plans that simultaneously satisfy (2) and (3). These results along with the lemmas below let us develop a new  $O(T^2)$  algorithm for problem (1). Accordingly, let  $G(t)$ ,  $t = 1, \dots, T$ , be the optimal cost of the problem consisting of periods  $t$  to  $T$ , with  $G(T+1) = 0$ , and let  $c_t = p_t + \sum_{i=t}^T h_i$  be the marginal cost of producing an item unit in period  $t$  and holding it up to period  $T$ . Note that adding the same amount to all marginal production costs shifts the objective function of all feasible solutions by the same quantity. Hence, not the values, but rather the differences between marginal production costs play a role in determining the optimal solution. Moreover, let  $x_{s,t}^*$  be the optimal decision in period  $t$  when the problem consisting of periods  $s$  to  $T$  is independently solved, where  $s \leq t$ , and let  $\delta(z)$  denote a delta function such that  $\delta(0) = 1$  and  $\delta(z) = 0$  if  $z \neq 0$ .

#### A. The Recursion Formula

For notational convenience, we denote by  $I_{i,l}^*$  the optimal inventory level at the end of period  $l$  when the problem starting at period  $i \leq l$  is independently solved. Moreover, given a period  $i$ , we assume that the inventory level at the end of the period  $i - 1$  is zero when the problem starting at period  $i$  is independently solved, that is,  $I_{i,i-1}^* = 0$ . In what follows, let  $\hat{x}_t = x_{t,t}^*$  be the optimal production quantity in period  $t$  when the problem with periods from  $t$  to  $T$  is independently solved.

*Lemma 1:* If  $\hat{x}_t = M_t$  for a given period  $t$ , then there exists an optimal plan for the problem starting at period  $t$  with at least one period  $k \in [j, R_t + 1]$  such that  $I_{j,k-1}^* + x_{j,k}^* \geq M_t - d_{t,k-1}$  for some period  $j \in [t + 1, R_t + 1]$ .

*Proof:* We know by (2) that when the problem starting at period  $t$  is independently solved (i.e., assuming that  $I_{t,t-1}^* = 0$ ), the only two decisions to consider are either  $\hat{x}_t = d_{t,l} < M_t$  for some period  $l \in [t, R_t]$ , or  $\hat{x}_t = M_t$ . Moreover, we know that the quantity  $\hat{x}_t = M_t$  is enough to completely satisfy the demands for periods  $t$  to  $R_t$ , and to partially satisfy the demand in period  $R_t + 1$  (i.e.,  $M_t - d_{t,R_t} < d_{R_t+1}$ , or equivalently,  $M_t = d_{t,R_t} + \lambda d_{R_t+1}$  with  $\lambda \in (0, 1)$ ).

Additionally, taking into account the way in which the values  $M_i$  ( $i = 1, \dots, T$ ) are obtained, it is clear that  $M_i - d_{i,l-1} \leq M_l$  for all  $l \in [i + 1, R_i + 1]$ , otherwise there would be a period in  $[i + 1, R_i + 1]$  where the storage constraint is violated.

For a contradiction, let us suppose that  $I_{j,n-1}^* + x_{j,n}^* < M_t - d_{t,n-1}$  for all  $n \in [j, R_t + 1]$  and  $j \in [t + 1, R_t + 1]$ . In particular let us consider period  $j$ , so we obtain that  $I_{j,j-1}^* + x_{j,j}^* = \hat{x}_j < M_t - d_{t,j-1} = d_{j,R_t} + \lambda d_{R_t+1} \leq M_j$  with  $\lambda \in (0, 1)$ . Note that, in this case,  $I_{j,j-1}^* = 0$  since we are independently solving the problem with initial period  $j$ . Additionally, it follows that  $\hat{x}_j < M_j$  and according to (2), there should be an optimal plan where the decision  $\hat{x}_j$  consists of a sum of demands, i.e.,  $\hat{x}_j = d_{j,i-1}$  with  $i \in [j + 1, R_t + 1]$ . Observe that the inventory on hand  $I_{j,l}^*$  at the end of any period  $l \in [j, i - 1]$  corresponds to the sum of demands  $d_{l+1,i-1} = d_{j,i-1} - d_{j,l}$ , and therefore, applying (3) there must be an optimal plan where  $x_{j,l}^* = 0$  for any period  $l \in [j + 1, i - 1]$ . It is worth noting that the plan between periods  $j$  and  $i - 1$  represents a ZIO policy. Moreover, note that the demands for periods  $i, i + 1, \dots, R_t, R_t + 1$  have not been covered yet. Hence, to prevent a shortage, period  $i$  should produce a quantity different from zero. However, by hypothesis, in period  $i$  the following inequality  $I_{j,i-1}^* + x_{j,i}^* < M_t - d_{t,i-1} = d_{i,R_t} + \lambda d_{R_t+1} \leq M_i$  also holds. Since the plan between periods  $j$  and  $i - 1$  is ZIO, the inventory on hand  $I_{j,i-1}^*$  at the end of period  $i - 1$  is zero, and hence  $I_{j,i-1}^* + x_{j,i}^* = x_{j,i}^* < M_t - d_{t,i-1} = d_{i,R_t} + \lambda d_{R_t+1} \leq M_i$ . Equivalently  $x_{j,i}^* < M_i$ , and by (2),  $x_{j,i}^* = \hat{x}_i$  corresponds with a sum of demands from period  $i$  to a period  $m - 1$  so that  $m \in [i + 1, R_t + 1]$ , namely,  $\hat{x}_i = d_{i,m-1}$ . Furthermore, in accordance with (3), there should be an optimal plan where the optimal decision for those periods in  $[i + 1, m - 1]$  is not to produce, and hence  $x_{j,l}^* = 0$  for any period  $l \in [i + 1, m - 1]$ . Consequently the plan between periods  $i$  and  $m - 1$  is ZIO.

If we continue applying the same argument for all intermediate production periods between  $m$  and  $R_t + 1$ , we finally attain period  $R_t + 1$ . Remark that the plan from period  $j$  to period  $R_t$  represents a ZIO policy. Accordingly, it follows that  $I_{j,R_t}^* = 0$  and, by hypothesis, that  $I_{j,R_t}^* + x_{j,R_t+1}^* = x_{j,R_t+1}^* = \hat{x}_{R_t+1} < M_t - d_{t,R_t} = \lambda d_{R_t+1} \leq M_{R_t+1}$ , that is,  $x_{j,R_t+1}^* = \hat{x}_{R_t+1} < M_{R_t+1}$  and  $x_{j,R_t+1}^* = \hat{x}_{R_t+1} < \lambda d_{R_t+1}$ . Thus, according to (2),  $x_{j,R_t+1}^* = \hat{x}_{R_t+1} = 0$ . As a result, the demand for period  $R_t + 1$  has not been produced through periods from  $j$  to  $R_t + 1$  and, therefore, a stockout occurs. Hence, to avoid this infeasible fact, there must be at least a period  $k \in [t + 1, R_t + 1]$  such that  $x_{j,k}^* \geq M_t - d_{t,k-1}$  for some period  $j \in [t + 1, k - 1]$ . ■

For the sake of simplicity, we show in Figure 1 an illustration of Lemma 1 assuming that the demands for periods in  $[t + 1, R_t + 1]$  are equal and they occur at constant rate instead of instantaneously.

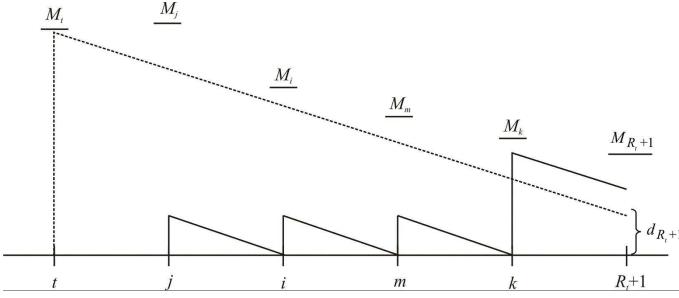


Fig. 1. Illustration of Lemma 1.

In what follows, let  $K_t$  be the set of production periods  $i$  within the interval  $[t+1, R_t + 1]$  such that  $I_{t+1,i-1}^* + x_{t+1,i}^* \geq M_t - d_{t,i-1}$ , that is  $K_t = \{i \in [t+1, R_t + 1] | I_{t+1,i-1}^* + x_{t+1,i}^* \geq M_t - d_{t,i-1} \text{ and } x_{t+1,i}^* > 0\}$ . Thus, the following result states that among those periods  $i \in K_t$  there exists one, say  $k$ , such that if the optimal decision in period  $t$  is to order its maximum quantity, then the optimal decision for those periods  $j \in [t+1, k-1]$  is not to order. Furthermore, in this case, the optimal decision in period  $k$  consists of producing the difference between  $\hat{x}_k$  and  $M_t - d_{t,k-1}$ .

**Lemma 2:** Let  $\hat{x}_t = M_t$  be the optimal decision in period  $t$  and let  $j \in [t+1, R_t + 1]$  be a period such that  $I_{t+1,j-1}^* + x_{t+1,j}^* < M_t - d_{t,j-1}$ , then the optimal decision in period  $j$  is not to produce, i.e.,  $x_{t,j}^* = 0$ .

**Proof:** Since  $I_{t+1,j-1}^* + x_{t+1,j}^* < M_t - d_{t,j-1}$ , by virtue of Lemma 1, we can denote by  $k_1$  the period with smallest index in set  $K_t$ . For a contradiction, assume that the optimal decision in period  $j$  when  $\hat{x}_t = M_t$  consists of producing a quantity  $q \in (0, I_{t+1,k_1-1}^* + x_{t+1,k_1}^* - (M_t - d_{t,k_1-1}))$ . Therefore, the following inequality holds

$$f_j + c_j q < c_{k_1} q$$

which contradicts the fact that  $x_{t+1,k_1}^*$  is the optimal order quantity in period  $k_1$  when problem starting in period  $t+1$  is independently solved, since a better solution can be obtained just producing a quantity  $x_{t+1,j}^* + q$  in period  $j$  and a quantity  $x_{t+1,k_1}^* - q$  in period  $k_1$ . Accordingly, the optimal decision in period  $j$  when  $\hat{x}_t = M_t$  is  $x_{t,j}^* = 0$ . ■

**Lemma 3:** Let  $\hat{x}_t = M_t$  be the optimal decision in period  $t$  and let  $j \in [t+1, R_t + 1]$  be a period such that  $I_{t+1,j-1}^* + x_{t+1,j}^* \geq M_t - d_{t,j-1}$ , where the optimal decision for those periods  $i$  in  $[t+1, j-1]$  is  $x_{t,i}^* = 0$ , then the optimal decision in period  $j$  is either  $x_{t,j}^* = 0$  or  $x_{t,j}^* = I_{t+1,j-1}^* + x_{t+1,j}^* - (M_t - d_{t,j-1})$ .

**Proof:** For a contradiction, let us assume that it is optimal to produce in period  $j$  a quantity  $q > 0$  different from  $I_{t+1,j-1}^* + x_{t+1,j}^* - (M_t - d_{t,j-1})$ . For notational convenience, let  $\Delta$  be equal to  $I_{t+1,j-1}^* + x_{t+1,j}^* - (M_t - d_{t,j-1})$ . Under this assumption we should distinguish two

cases, namely, if  $x_{t,j}^* = q < \Delta$  or the reverse. In the first case, we can apply (2) to obtain that  $I_{t,j-1}^* + q = d_{j,l}$  for some period  $l \geq R_t + 1$  and  $I_{t,j-1}^* = M_t - d_{t,j-1}$ . Note that  $I_{t,j-1}^* = M_t - d_{t,j-1}$  as a result of assuming that  $x_{t,i}^* = 0$  for any period  $i \in [t+1, j-1]$ . Furthermore, by (3), observe that  $x_{t,i}^* = 0$  for any period  $i \in [j+1, l]$ . Thus, the following inequality holds

$$f_j + c_j q + f_{l+1} + c_{l+1}(\Delta - q) < f_j + c_j q + c_j(\Delta - q)$$

or, equivalently

$$f_{l+1} + c_{l+1}(\Delta - q) < c_j(\Delta - q)$$

which yields the contradiction that  $x_{t+1,j}^*$  is not the optimal order quantity in period  $j$  when problem starting in period  $t+1$  is independently solved, since a better solution can be obtained simply producing the quantity  $q < x_{t+1,j}^*$  in period  $j$  and a quantity equal to  $\Delta - q$  in period  $l+1$ . As a result, the optimal decision  $x_{t,j}^*$  in any period  $j \in [t+1, R_t + 1]$  such that  $I_{t+1,j-1}^* + x_{t+1,j}^* \geq M_t - d_{t,j-1}$  cannot be below the quantity  $\Delta$ .

In second place, we address the situation in which it is optimal to produce a quantity  $x_{t,j}^* = q > \Delta$  in period  $j$  when  $\hat{x}_t = M_t$ . Remark that if it is feasible to produce an additional quantity  $q - \Delta$  in period  $j$  then  $I_{t+1,j-1}^* + x_{t+1,j}^* < M_j$ , and by (2),  $I_{t+1,j-1}^* + x_{t+1,j}^* = d_{j,l}$  for some period  $l \geq R_t + 1$ . For a contradiction, we assume that it is preferable to order in period  $j$  a quantity  $q$  instead of ordering a quantity equal to  $\Delta$  in period  $j$  and a quantity  $q - \Delta$  in period  $l+1$ . Formally, this assumption can be stated as follows

$$f_j + c_j(q - \Delta) + c_j\Delta < f_j + c_j\Delta + f_{l+1} + c_{l+1}(q - \Delta)$$

That is,

$$c_j(q - \Delta) < f_{l+1} + c_{l+1}(q - \Delta)$$

and hence, we obtain the following inequality

$$f_j + c_j x_{t+1,j}^* + c_j(q - \Delta) < f_j + c_j x_{t+1,j}^* + f_{l+1} + c_{l+1}(q - \Delta)$$

which implies that instead of being  $x_{t+1,j}^*$  the optimal order quantity in period  $j$  for the problem starting in period  $t+1$ , it should have been  $x_{t+1,j}^* + q - \Delta$ . Consequently, the optimal decision  $x_{t,j}^*$  in any period  $j \in [t+1, R_t + 1]$  such that  $I_{t+1,j-1}^* + x_{t+1,j}^* \geq M_t - d_{t,j-1}$  cannot be above the amount  $\Delta$ . Therefore, when  $\hat{x}_t = M_t$ , the optimal decision for those periods  $j \in [t+1, R_t + 1]$  such that  $I_{t+1,j-1}^* + x_{t+1,j}^* \geq M_t - d_{t,j-1}$  is either  $x_{t,j}^* = 0$  or  $x_{t,j}^* = \Delta = I_{t+1,j-1}^* + x_{t+1,j}^* - (M_t - d_{t,j-1})$ . ■

We can use the results above to introduce a functional equation, which characterizes a subset of optimal plans to problem (1).

*Theorem 4:* An optimal production plan for problem (1) is given by the following recurrence formula

$$\begin{aligned} \text{if } d_t > 0 : \\ G(t) &= \min \left[ \min_{\substack{t < j \leq R_t + 1 \\ \hat{x}_j \geq M_t - d_{t,j-1}}} A(t, j), \min_{\substack{t < j \leq R_t + 1 \\ \hat{x}_j \geq M_t - d_{t,j-1}}} B(t, j) \right], \\ \text{if } d_t = 0 : \\ G(t) &= \min [G(t+1), \min_{\substack{t+1 < j \leq R_t + 1 \\ t < j \leq R_t + 1 \\ \hat{x}_j \geq M_t - d_{t,j-1}}} A(t, j), \\ &\quad \min_{\substack{t < j \leq R_t + 1 \\ \hat{x}_j \geq M_t - d_{t,j-1}}} B(t, j)] \end{aligned} \tag{4}$$

where  $B(t, j) = f_t + c_t M_t + G(j) - (c_j(M_t - d_{t,j-1}) + \delta(M_t - d_{t,j-1} - \hat{x}_j)f_j)$  and  $A(t, j) = f_t + c_t d_{t,j-1} + G(j)$ .

*Proof:* Assuming that  $I_{t-1} = 0$  ( $t = 1, \dots, T$ ), expression (2) states that the production quantity in period  $t$  consists of either a sum of demands corresponding to consecutive periods or  $M_t$ . The former decision corresponds to the terms denoted by  $A(t, j)$  in (4). On the other hand, the latter decision concerns the expression  $B(t, j)$ . Indeed, when  $\hat{x}_t = M_t$ , only those periods  $j$  in  $[t+1, R_t + 1]$  satisfying  $\hat{x}_j \geq M_t - d_{t,j-1}$  must be considered as it was proved in Lemma 3. ■

It is clear that a straightforward implementation of the recursion formula above yields an  $O(T^2)$  algorithm, reducing the complexity  $O(T^3)$  of both Love's procedure (1973) and the method given in Gutiérrez et al. (2003), which have been devised for more general cost structures.

### B. The Geometrical Technique

A more efficient algorithm can be developed applying a procedure based on the approach proposed by Wagelmans et al. (1992). In particular, these authors argued that only the efficient periods should be considered for the determination of  $\min_{t < j \leq T+1} \{c_t d_{t,j-1} + G(j)\}$  in the uncapacitated case. Accordingly, assume that we plot the points  $(d_{t,T}, G(t))$ ,  $t = 1, \dots, T+1$ , such that the cumulative demands are put on the horizontal axis and the vertical axis represents the optimal costs. Thus, a period is said to be efficient when it corresponds to a breakpoint of the lower convex envelope of points  $(d_{t,T}, G(t))$ ,  $t = 1, \dots, T+1$ . The implementation of this technique consists of evaluating the periods from  $T$  to 1 and holding the efficient periods in a list  $L$ . This list is sorted by ratios which represent the slopes of the line segments joining consecutive efficient periods (breakpoints) of the lower convex envelope. Each time a

new period  $j$  is considered, the procedure looks for the smallest efficient period  $q(j)$  in  $L$  with ratio smaller than  $c_j$ , and the lower envelope is updated removing from  $L$  the non-efficient periods  $j+1$  to the predecessor of  $q(j)$  in  $L$ .

Unfortunately, this technique cannot be used directly when the inventory levels are limited. Unlike the geometrical approach proposed by Wagelmans et al. (1992), in our procedure the non-efficient periods cannot be discarded since a period that is not efficient for a problem consisting of periods  $j$  to  $R_j + 1$  could be efficient for the problem involving periods  $t$  to  $R_t + 1$ , with  $j > t$ . However, we can adapt this geometrical technique to our model in the following way. We should define two lists  $L_E$  and  $L_{NE}$  containing, respectively, the efficient and non-efficient periods. These lists are also sorted by the slopes-ratios. When evaluating period  $j$ , if  $q(j)$  is smaller than  $R_j + 1$ , then the new procedure proceeds in the same way that the approach in Wagelmans et al. (1992), i.e., producing  $d_{j,q(j)-1}$  units. In case of  $q(j)$  equals  $R_j + 1$ , we can make two decisions, namely, to order either  $M_j$  or  $d_{j,R_j}$ . Nevertheless, when  $c_j < c_{q(j)}$ , it can be easily proved that the optimal decision consists of producing  $M_j$ . Otherwise, the optimal decision is  $d_{j,q(j)-1}$ . Finally, when  $q(j) > R_j + 1$ , the efficient period  $q(j)$  is not feasible for the problem starting in period  $j$ , and hence we must compare both the efficient period  $q_E(j) \leq R_j + 1$  in  $L_E$  with smallest ratio and the non-efficient period  $q_{NE}(j) \leq R_j + 1$  in  $L_{NE}$ . Accordingly, we denote by  $G_E(j) = f_j + c_j d_{j,q_E(j)-1} + G(q_E(j))$  and  $G_{NE}(j) = f_j + c_j d_{j,q_{NE}(j)-1} + G(q_{NE}(j))$  the costs associated to, respectively, periods  $q_E(j)$  and  $q_{NE}(j)$ , which are the successors to  $j$ . If evaluating both costs we obtain that  $G_E(j) \leq G_{NE}(j)$ , then period  $q_E(j)$  remains to be efficient. Otherwise, since  $G_E(j) > G_{NE}(j)$ , the following proposition shows that period  $q_{NE}(j)$  should be inserted in list  $L_E$  and the rest of periods in this list have to be moved to  $L_{NE}$ . Actually, this process to transfer periods from one list to the other represents an update of the lower envelope.

*Proposition 5:* If evaluating a period  $j$ , both  $q(j) > R_j + 1$  and  $G_E(j) > G_{NE}(j)$  hold, then period  $q_{NE}(j)$  should be included in list  $L_E$  and the rest of periods in this list must be moved to list  $L_{NE}$ .

*Proof:* Without loss of generality, we assume that  $q(q_E(j)) = q(j)$  and  $q(j)$  is the period successor to  $q_E(j)$  in  $L_E$ . Note that  $G(q_{NE}(j)) + c_j d_{l,q_{NE}(j)-1} < G(l)$  for any period  $l$  in  $L_E$  smaller than or equal to  $q_E(j)$ , and hence  $G(q_{NE}(j)) < G(l)$ . Otherwise,  $f_j + c_j d_{j,k-1} + G(k) < f_j + c_j d_{j,q_{NE}(j)-1} + G_{NE}(j)$  for some  $k \leq q_E(j)$  in  $L_E$ , and therefore  $q_E(j) = k$  with  $G_E(j) < G_{NE}(j)$ , which contradicts the hypothesis.

Recall that for a production/reordering period  $t$ ,  $\hat{x}_t \in \{M_t, d_{t,q(t)-1}\}$ . In addition, since  $q_{NE}(j) < q_E(j)$ , the straight line connecting points  $(d_{j,T}, G_{NE}(j))$  and  $(d_{q_{NE}(j),T} - (\hat{x}_j - d_{j,q_{NE}(j)-1}), G(q_{NE}(j)) - c_{q_{NE}(j)}(\hat{x}_j - d_{j,q_{NE}(j)-1}))$  intercepts the line segment joining points  $(d_{q_E(j),T}, G(q_E(j)))$  and  $(d_{q_E(j),T} - (\hat{x}_{q_E(j)} - d_{q_E(j),q(j)-1}), G(q(j)) - c_{q(j)}(\hat{x}_{q_E(j)} - d_{q_E(j),q(j)-1}))$  in a point smaller than  $q_E(j)$ , and hence the result below follows

$$\frac{G_{NE}(j) - (G(q_{NE}(j)) - c_{q_{NE}(j)}(\hat{x}_j - d_{j,q_{NE}(j)-1}))}{\hat{x}_j} < \frac{G(q_E(j)) - (G(q(j)) - c_{q(j)}(\hat{x}_{q_E(j)} - d_{q_E(j),q(j)-1}))}{\hat{x}_{q_E(j)}}$$

Moreover, given that the term on the right-hand side in the above expression is smaller than the ratio  $\frac{G(k) - G(q(k))}{d_{k,q(k)-1}}$ , for any period  $k < q_E(j)$  in  $L_E$ , these periods are to be dominated by  $q_{NE}(j)$ . For that reason, these periods should be moved to list  $L_{NE}$ . ■

Figure 2 shows the case where  $\hat{x}_j = d_{j,q_{NE}(j)-1}$ . Note that periods highlighted by the gray line are not accessible from period  $j$ .

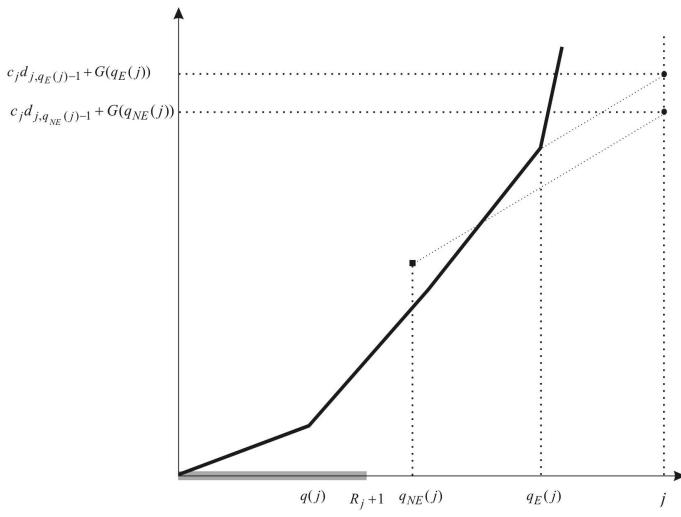


Fig. 2. Illustration of the case  $q(j) > R_j+1$  and  $G_E(j) > G_{NE}(j)$ .

Following a similar argument to the previous proposition, we can state the following result

*Proposition 6:* If evaluating a period  $j$ , it holds that  $q(j) > R_j+1$ ,  $G_E(j) < G_{NE}(j)$  and the ratio related to period  $q_E(j)$  is greater than  $\frac{G_E(j) - G(q_E(j))}{d_{j,q_E(j)-1}}$ , then every period  $k \leq q_E(j)$  in  $L_E$  should be moved to list  $L_{NE}$ .

The method outlined above is shown in Algorithm 1, where  $pred(j)$  and  $succ(j)$  denote, respectively, the period predecessor and successor to period  $j$  in both lists. We also follow the convention that if  $d_j = 0$ , then the efficient period  $j+1$  is replaced by the efficient period  $j$ . Regarding the complexity of this procedure, note that the value  $q(j)$  can be obtained by binary search in  $O(\log T)$ .

In case of  $q(j) > R_j+1$ , the procedure should inspect by sequential search in both  $L_E$  and  $L_{NE}$  to determine the actual period,  $q_E$  or  $q_{NE}$ , successor to  $j$ . Specifically, if we are evaluating period  $j$ , there would be, at most,  $(T-j)$  periods distributed in both lists. Each time the sequential search reaches a period greater than  $R_j+1$ , this period is removed from the corresponding list, and it will not be considered in subsequent search processes. Observe that each comparison in any of the two lists, when  $q(j) > R_j+1$ , yields a deletion of the corresponding period. Hence, the overall number of comparisons is  $O(T)$ . Therefore, the process of searching all values  $q(j)'s$  ( $j = 1, \dots, T$ ) runs in  $O(T \log T) + O(T)$ . On the other hand, any period  $j$  can be moved between both lists at most two times, and so, the transferring process can be carried out in  $O(T)$  time. According to the previous arguments, the algorithm runs in  $O(T \log T)$ .

### C. The Wagner and Whitin Cost Structure

In addition to the case where production/ordering costs are time-varying, we also address the problem admitting that production/ordering costs are constant, i.e., when  $p_t = p$  for all  $t$ . Under this assumption, the formulation of problem (1) adopts a form equivalent to that in the Wagner and Whitin model and, hence,  $c_1, c_2, \dots, c_T$  represent a non increasing sequence of values. Therefore, speculative motives for holding stock are not allowed. It is well-known that, under this cost structure and in absence of capacities, the problem admits an optimal plan  $\mathbf{x} = (x_1, \dots, x_T)$  verifying  $I_{t-1}x_t = 0$ , for  $t = 1, \dots, T$ . This result is commonly refereed to as Zero Inventory Ordering (ZIO) property. Indeed, the ZIO property still holds when the cost functions are concave in general (see Wagner (1960) and Zangwill (1968)). Moreover, as we show in Proposition 7, the ZIO property holds even when inventory levels are limited. Therefore, the use of the ZIO property is not conditioned to limitations on the inventory levels.

*Proposition 7:* When production/ordering costs in problem (1) are constant, there always exists an optimal policy  $\mathbf{x} = (x_1, \dots, x_T)$  such that  $I_{t-1}x_t = 0$ ,  $t = 1, \dots, T$ .

*Proof:* Let us assume that there exists an optimal plan  $\mathbf{x}$  with at least one period  $j$  such that  $I_{j-1}x_j \neq 0$ . According to Lemma 3, since  $x_j \neq 0$ , then there must exists a period  $t$ ,  $t < j$ , such that  $\hat{x}_j$  is strictly greater than  $M_t - d_{t,j-1}$ , which corresponds to  $I_{j-1}$ . Therefore, the following inequality holds

$$f_t + c_t M_t + G(j) - c_j(M_t - d_{t,j-1}) < f_t + c_t d_{t,j-1} + G(j)$$

that is,  $c_t < c_j$ , which contradicts the fact that  $c_t \geq c_j$ . ■

---

**Algorithm 1** Determine an optimal plan  $\mathbf{x} = (x_1, \dots, x_T)$  for problem  $P$ 


---

```

1: DATA: vectors  $d, c, h, f, S$  and the number of
   periods  $T$ 
2: calculate  $C_t, M_t$  and  $p_t$ ,  $t = 1, \dots, T + 1$ 
3:  $G(T + 1) \leftarrow 0$ 
4: insert  $T + 1$  in  $L_E$ 
5: for  $i \leftarrow T$  downto 1 do
6:   search for  $q(i) \leftarrow \min[T + 1, \min\{j \in L_E : \frac{G(j)-G(\text{succ}(j))}{\hat{x}_j} < c_i\}]$ 
7:   if  $(q(i) < R_i + 1)$  or  $(q(i) = R_i + 1 \text{ and } c_i > c_{q(i)})$  then
8:      $G(i) \leftarrow f_i + c_i d_{i,q(i)-1} + G(q(i)); \hat{x}_i \leftarrow d_{i,q(i)-1}$ 
9:   else
10:    if  $q(i) = R_i + 1$  then
11:       $G(i) = f_i + c_i M_i + G(q(i)) - c_{q(i)}(M_i - d_{i,q(i)-1}); \hat{x}_i \leftarrow M_i$ 
12:    else
13:       $j \leftarrow \text{pred}(q(i)); \text{while } j > R_i + 1 \text{ do } j \leftarrow \text{pred}(j)$ 
14:      delete all  $k : q(i) \leq k < j$  from  $L_E$ 
15:       $q_E(i) \leftarrow j; G_E(i) \leftarrow f_i + c_i d_{i,q_E(i)-1} + G(q_E(i)); G_{NE}(i) \leftarrow -1$ 
16:      if  $L_{NE}$  is not empty then
17:         $j \leftarrow \text{first element in } L_{NE}; \text{while } j > R_i + 1 \text{ do } j \leftarrow \text{pred}(j)$ 
18:        delete all  $k : 1 \leq k < j$  from  $L_{NE}; q_{NE}(i) \leftarrow j$ 
19:        if  $(q_{NE}(i) < R_i + 1)$  or  $(q_{NE}(i) = R_i + 1 \text{ and } c_i > c_{q_{NE}(i)})$  then
20:           $G_{NE}(i) \leftarrow f_i + c_i d_{i,q_{NE}(i)-1} + G(q_{NE}(i)); z \leftarrow d_{i,q_{NE}(i)-1}$ 
21:        else
22:           $G_{NE}(i) \leftarrow f_i + c_i M_i + G(q_{NE}(i)) - c_{q_{NE}(i)}(M_i - d_{i,q_{NE}(i)-1}); z \leftarrow M_i$ 
23:        end if
24:      end if
25:      if  $G_{NE}(i) \geq 0$  and  $G_{NE}(i) < G_E(i)$  then
26:         $G(i) \leftarrow G_{NE}(i); q(i) \leftarrow q_{NE}(i); \hat{x}_i \leftarrow z$ 
27:      else
28:         $G(i) \leftarrow G_E(i); q(i) \leftarrow q_E(i); \hat{x}_i \leftarrow z$ 
29:      end if
30:    end if
31:  end if
32:  at this point, values  $G(i)$  and  $\hat{x}_i$  have been already
   determined
33:  call Algorithm 2 to update the lower envelope
34: end for
35: call Algorithm 3 to arrange the optimal solution

```

---



---

**Algorithm 2** Routine to update the lower envelope

---

```

1: if  $q(i) \leq R_i + 1$  or  $(q(i) = q_E(i) \text{ and } \frac{G(i)-G(q(i))}{\hat{x}_i} > \frac{G(q(i))-G(\text{succ}(q(i)))}{\hat{x}_{q(i)}})$  then
2:   if  $d_i = 0$  and  $G(i + 1) < G(i)$  then
3:      $G(i) \leftarrow G(i + 1); s \leftarrow \text{succ}(i + 1)$ 
4:   else
5:     if  $d_i > 0$ , then  $s \leftarrow i + 1$  else  $s \leftarrow \text{succ}(i + 1)$ 
6:     while  $\frac{G(i)-G(s)}{d_{i,s-1}} \leq \frac{G(s)-G(\text{succ}(s))}{\hat{x}_s}$  and  $s < q(i)$ 
      do
7:        $s \leftarrow \text{succ}(s)$ 
8:     end while
9:   end if
10:  move all  $k : i + 1 \leq k < s$  from  $L_E$  to  $L_{NE}$ ;
    insert  $i$  in  $L_E$ 
11: else
12:   move all periods in  $L_E$  to  $L_{NE}$ ; insert  $i$  in  $L_E$ 
13: end if

```

---

The above proposition allow us to reformulate expression (4) as follows

$$G(t) = \begin{cases} \min_{t < j \leq R_t + 1} A(t, j) & \text{if } d_t > 0 \\ \min[G(t + 1), \min_{t+1 < j \leq R_t + 1} A(t, j)] & \text{if } d_t = 0 \end{cases} \quad (5)$$

$A(t, j) = f_t + c_t d_{t,j-1} + G(j)$ . Remark that expression (5) only differs from that proposed by Wagelmans et al. (1992) in the range of  $j$ . Unfortunately, this result does not imply a computational improvement since each non-efficient period should be sorted in  $O(\log T)$  when it is inserted in  $L_{NE}$ .

As an illustration of this latter result, we present the following numerical example. Assuming that the production/ordering unit costs are equal to zero, the rest of the input data are shown in Table I, where the first column corresponds to the period and the subsequent columns represent, respectively, the demand, the setup cost and the storage capacity.

The corresponding trace to the problem introduced in Table I is shown in Table II. In particular, the rows in this table stand for the iterations (periods), and the second and third columns show the maximum quantity to be produced/ordered and the maximum reachable period for each period, respectively. Additionally, we show in columns four to six the values of  $q(j)$ ,  $G(j)$  and  $\text{Ratio} = \frac{G(j)-G(\text{succ}(j))}{d_{j,\text{succ}(j)-1}}$  following (5). Finally, the last two columns contain lists  $L_E$  and  $L_{NE}$ , where the symbol  $\{\emptyset\}$  indicates that the list is empty. Note that, in absence of capacities, the optimal solution for the example in Table I is  $(22, 0, 0, 24, 0, 22, 0, 0, 8, 0)$

**Algorithm 3** Routine to arrange the optimal solution

```

1:  $Cost \leftarrow 0; i \leftarrow 1; D \leftarrow 0; Rest \leftarrow 0$ 
2: while  $i \leq T$  do
3:   if  $d_i = 0$  and  $G(i) = G(i + 1)$  then
4:      $i \leftarrow i + 1$ 
5:   else
6:     if  $q(i) = R_i + 1$  and  $c_i < c_{q(i)}$  then
7:        $\hat{x}_i \leftarrow M_i - Rest; D \leftarrow \hat{x}_i + Rest - d_i; Rest \leftarrow M_i - d_{i,q(i)-1}$ 
8:     else
9:        $\hat{x}_i \leftarrow d_{i,q(i)-1} - Rest; D \leftarrow \hat{x}_i + Rest - d_i; Rest \leftarrow 0$ 
10:    end if
11:    if  $\hat{x}_i = 0$  then  $f \leftarrow 0$  else  $f \leftarrow f_i$ 
12:     $Cost \leftarrow Cost + f + c_i \hat{x}_i + Dh_i$ 
13:    for  $k \leftarrow i + 1$  to  $q(i) - 1$  do
14:       $D \leftarrow D - d_k; Cost \leftarrow Cost + Dh_k$ 
15:    end for
16:     $i \leftarrow q(i)$ 
17:  end if
18: end while
19: return  $Cost$ 

```

$j$	$d_j$	$f_j$	$c_j$	$S_j$
1	5	1	10	10
2	10	30	9	15
3	7	20	8	10
4	9	2	7	20
5	15	40	6	25
6	4	1	5	22
7	8	30	4	10
8	10	25	3	10
9	2	10	2	10
10	6	28	1	10

TABLE I

INPUT DATA FOR ONE INSTANCE OF PROBLEM (1).

whereas considering capacities yields the optimal plan  $(5, 10, 7, 9, 15, 12, 0, 10, 8, 0)$ .

## IV. COMPUTATIONAL RESULTS

We show in Table III the average running times of Algorithm 1 introduced in Section 3, and the average running times of the dynamic programming algorithm developed from the recurrence formula (4). Both algorithms have been implemented using C++ along with LEDA 4.2.1 libraries and were tested in a HP-712/80 (80 MHz) workstation. For simplicity, we denote by  $T \log T$  the *Alg. 1* and by *DPA* the algorithm obtained from (4), respectively. The different values for the maximum storage capacity ( $S$ ) and the number of periods ( $T$ ) are

$j$	$M_j$	$R_j$	$q(j)$	$G(j)$	$Ratio$	$L_E$	$L_{NE}$
10	6	10	11	34	5.66	{11, 10}	{ $\emptyset$ }
9	8	10	11	26	3.25	{11, 9}	{10}
8	10	8	9	81	5.50	{11, 9, 8}	{ $\emptyset$ }
7	10	7	8	143	7.75	{11, 9, 8, 7}	{ $\emptyset$ }
6	14	7	8	142	5.08	{9, 6}	{8, 7}
5	25	6	6	272	8.66	{9, 6, 5}	{8, 7}
4	20	4	5	337	7.22	{6, 4}	{5}
3	10	3	4	413	10.85	{6, 4, 3}	{5}
2	15	2	3	533	12.00	{6, 3, 2}	{ $\emptyset$ }
1	10	1	2	584	10.20	{3, 1}	{2}

TABLE II  
THE OUTPUT DATA CORRESPONDING TO THE INSTANCE IN TABLE  
1.

$S$	$T$					
	25		50		75	
$S$	$DPA$	<i>Alg.1</i>	$DPA$	<i>Alg.1</i>	$DPA$	<i>Alg.1</i>
100	0.016	0.005	0.046	0.012	0.088	0.023
500	0.015	0.005	0.040	0.013	0.070	0.025
1000	0.015	0.006	0.041	0.013	0.071	0.023
2000	0.015	0.006	0.041	0.013	0.069	0.022
5000	0.016	0.006	0.043	0.014	0.074	0.024
10000	0.014	0.005	0.044	0.013	0.074	0.024
100000	0.015	0.005	0.040	0.015	0.071	0.024

$S$	$T$			
	100		150	
$S$	$DPA$	<i>Alg.1</i>	$DPA$	<i>Alg.1</i>
100	0.138	0.035	0.274	0.066
500	0.103	0.037	0.179	0.069
1000	0.104	0.036	0.180	0.069
2000	0.106	0.037	0.186	0.069
5000	0.108	0.036	0.193	0.070
10000	0.111	0.036	0.193	0.069
100000	0.105	0.038	0.188	0.069

TABLE III  
AVERAGE RUNNING TIMES IN SECONDS FOR ALGORITHM 1  
(*Alg.1*) AND FOR THE DYNAMIC PROGRAMMING ALGORITHM  
(*DPA*) BASED ON RECURRENCE FORMULA GIVEN IN (4).

shown in the first row and column, respectively. For each pair  $(S, T)$ , we have run thirty instances with  $d_t$  varying in  $[0, S_t]$ ,  $t = 1, \dots, T$ . Moreover, for each pair  $(S, T)$ , we show two columns: the first containing the average running times of the dynamic programming algorithm based on the recurrence formula (4) and the second showing the average running times of Algorithm 1. From Table III, two main points can be highlighted. In first place, the running times of both algorithms support the theoretical complexities obtained in previous sections. Secondly, the running times of both procedures for a fixed number of periods do not seem to be affected by the values of the maximum storage capacity.

## V. CONCLUSIONS AND FINAL REMARKS

We have analysed the dynamic lot size problem with time-varying storage capacities. We have presented results which allow formulating an efficient recurrence expression for the problem. It has been shown that an appropriate implementation of this recurrence formula leads to an  $O(T^2)$  dynamic programming algorithm. Moreover, we have adapted the geometrical technique of Wagelmans et al. (1992) to develop an  $O(T \log T)$  algorithm. Another relevant aspect introduced in this paper is that the Zero Inventory Ordering (ZIO) property holds when cost specifications are considered as in the model by Wagner and Whitin (1958). These results represent a significant improvement with respect to the previous algorithm of Love (1973) and Gutiérrez et al. (2003). We intend to extend the results in this paper to the case in which shortages are allowed, and to the case with multiple items.

## REFERENCES

- [1] Aggarwal, A. and J.K. Park, "Improved Algorithms for Economic Lot Size Problems," *Oper. Res.* 41 (1993), 549-571.
- [2] Federgruen, A. and M. Tzur, "A Simple Forward Algorithm to Solve General Dynamic Lot Sizing Models with  $n$  Periods in  $O(n\log n)$  or  $O(n)$  Time," *Mgmt. Sci.* 37 (1991), 909-925.
- [3] Gutiérrez, J., A. Sedeño-Noda, M. Colebrook and J. Sicilia, "A New Characterization For The Dynamic Lot Size Problem with Bounded Inventory," *Comput. Oper. Res.* 30 (2003), 383-395.
- [4] LEDA User Manual (version 4.2.1) at [www.mpi-sb.mpg.de/LEDA/MANUAL/MANUAL.html](http://www.mpi-sb.mpg.de/LEDA/MANUAL/MANUAL.html)
- [5] Love, S.F., "Bounded Production and Inventory Models with Piecewise Concave Costs," *Mgmt. Sci.* 20 (1973), 313-318.
- [6] Manne, A. S., "Programming of Economic Lot Sizes," *Mgmt. Sci.* 4 (1958), 115-135.
- [7] Sedeño-Noda, A., J. Gutiérrez, B. Abdul-Jalbar, J. Sicilia, "An  $O(T\log T)$  Algorithm for the Dynamic Lot Size Problem with Limited Storage and Linear Costs," *Comput. Optim. Appl.* 28 (2004), 311-323.
- [8] Wagelmans, A., 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," *Oper. Res.* 40 (1992), S145-S156.
- [9] Wagner, H. M., "A Postscript to Dynamic Problems in the Theory of the Firm," *Naval Res. Logist. Quart.* 7 (1960), 7-12.
- [10] Wagner, H. M. and T. M. Whitin, "Dynamic Version of the Economic Lot Size Model," *Mgmt. Sci.* 5 (1958), 89-96.
- [11] Zangwill, W. I., "Minimum Concave Cost Flows in Certain Networks," *Mgmt. Sci.* 14 (1968), 429-450.



# Optimizing the service capacity by using a speed up simulation

Isolina Alberto\*, Fermin Mallor<sup>†</sup> and Pedro M. Mateo<sup>‡</sup>

\* Universidad de Zaragoza, Depto. Métodos Estadísticos

E.U. de Ingeniería Técnica Industrial, Email: isolina@unizar.es

<sup>†</sup> Universidad Pública de Navarra, Depto. Estadística e Investigación

Operativa, Email: mallor@unavarra.es

<sup>‡</sup> Universidad de Zaragoza, Depto. Métodos Estadísticos

Facultad de Ciencias, Email: mateo@unizar.es

**Abstract**— The aim of this work is to provide an effective method, based on a speed up simulation+optimization tool, that allows us to optimize the capacity of a queue model system subject to the completion of a quality of service criterion. This criterion is expressed in terms of a very small probability of loosing a customer due to system overload. To evaluate the performance of the system, multiple simulations with different capacity levels have to be made. Because of the characteristics of this kind of systems, any speed up tool will be necessary in order to keep the computational time reasonably bounded. Our tool uses a speed up technique, called RESTART, once, and it does not need any other entire execution of the method when a new capacity level is considered. Our tool only requires a stage (partial execution) of the RESTART method in order to update the probabilities when the capacity of the system is changed, so this involves an important saving of computational time.

**Keywords**— simulation, queuing system, optimization

## I. INTRODUCTION

MODERN telecommunication systems require high levels of quality of service (QoS) which is measured in terms of blocking or cell loss probabilities. The target values for these probabilities are lower than  $10^{-6}$  and  $10^{-9}$ , respectively. Because of this, when studying the performance of a real system defined by a capacity of service, a queue discipline and a buffer size, these features turn into a rare event. The use of Crude Monte Carlo techniques for evaluating such small probabilities is inefficient, because it requires a sample size that tends to infinity when the probability tends to zero, for a fixed relative error. For example, let  $X$  be a random variable and consider the estimation of  $\gamma = P(X \in A)$  for some event  $A$ . To estimate  $\gamma$  by crude Monte Carlo simulation, we draw  $N$  samples  $X_1, \dots, X_N$ , we denote

$I_n = I_{(X_n \in A)}$  and build the sample mean,

$$\hat{\gamma} = \frac{1}{N} \sum_{n=1}^N I_n, \quad E[\hat{\gamma}] = \gamma,$$

and

$$\text{Var}(\hat{\gamma}) = \frac{1}{N} \gamma(1 - \gamma), \quad \text{RE}(\hat{\gamma}) = \frac{\sqrt{\text{Var}(\hat{\gamma})}}{E[\hat{\gamma}]} \sim \frac{1}{\sqrt{\gamma N}},$$

so, the relative error is unbounded as the event becomes rarer, or equivalently, the smaller  $\gamma$ , the higher number of trials needed.

In most interesting real systems, it is necessary to speed up the simulation because the simulation with naive Monte Carlo techniques could require a prohibitively large number of trials. Several speed up techniques have been proposed: parallel programming, importance sampling, cross entropy and importance splitting (RESTART).

The first one takes advantage of the capacity of new technologies in order to develop or adapt the existing ones. The second one, importance sampling, changes the sampling distribution into a new one and takes samples from this new distribution. The new distribution is selected in such a way that the probability of the rare event is higher. In order to retain unbiased estimations, the observations must be corrected (because the samples are from the new distribution instead of the original one), that is done by means of the so-called likelihood ratio. The main problem of these methods is to find an appropriate new density. Cross entropy is used in conjunction with importance sampling, providing a simple adaptive procedure for estimating the optimal parameters needed for the importance sampling method.

Finally, importance splitting methods, inside of which we can locate the RESTART methods, use a different approach. The idea is based on restarting the simulation in states of the system that provoke rare events more

often. Then, the calculated probability for the rare event must be appropriately weighted.

In this paper we consider the problem of dimensioning the capacity of service to provide a fixed QoS. Analytical results providing an exact formula relating QoS and capacity of service are only available for simple models as, for instance, M/M/s/k [2] [8]. Also, it is possible to find asymptotic results for general models but assuming infinite buffer which can be very far from the exact ones for moderate sizes of the buffer. Then, if we consider general distributions and no asymptotic results are desired, we have to approach the problem by combining some optimization procedure with simulation. The integration of optimization techniques with simulation is widely spread in many fields as in industry [1], management, etc. In [9] the optimization of the service rate in a queue model using importance sampling method has been studied, but as these authors say “the idea of estimating rare-event probabilities and stochastic optimization over rare events is still in its infancy and considerable progress can be expected, specially in the domain of dynamic (queuing) networks”. The novelty in our case is that the values that are necessary to be estimated by using the simulation are very small and need to integrate a speed up procedure. During the last decade, many researchers have done important theoretical and practical advances in the development of efficient techniques for the study of systems involving rare events by using simulation. Directly related is the stochastic optimization over rare events. The scenario of competition in which modern firms have to develop their economic activities obliges them to be extremely efficient and to meet high levels of quality in their products and services. That means to reduce to almost zero the probability of failures in the production, of shortcoming in inventories, of disruptions in manufacturing chains or of having overflow in buffers.

The dimensioning of the resources that meet these extremely hard effectiveness parameters becomes an important problem, although it is usually very complicated to solve it in practice because of the complexity and stochastic nature of the elements involved.

## II. SPEED UP SIMULATION METHODS. RESTART ALGORITHM

Consider that we want to estimate the probability  $\gamma$  of a rare event  $E_v$ ,  $\gamma = P\{\text{rare event } E_v \text{ is observed}\}$ . The RESTART method, introduced in [12], is based on the idea of restarting the simulation in certain system states in order to do the rare event more likely to be observed. For this purpose, a sequence of nested events is defined, being the rare event the intersection of all of them. Then, the probability of the rare event

is the product of successive conditional probabilities, each of which can be estimated more accurately than the rare event probability for a given simulation effort. Let  $\{A_i\}_{i=1}^m$  be a sequence of events verifying  $E_v = A_m \subset A_{m-1} \subset \dots \subset A_1$  with probabilities  $p_i = P(A_i/A_{i-1}) \forall i = 2, \dots, m$  and  $p_1 = P(A_1)$ . Then, the probability of the rare event  $E_v$ ,  $\gamma = P(E_v)$ , can be expressed as  $\gamma = \prod_{i=2}^m P(A_i/A_{i-1}) \times P(A_1) = p_1 \times p_2 \times \dots \times p_m$ .

In order to illustrate the method, let us consider a system where customers arrive according to a certain interarrival time distribution and where each customer is served according to a preestablished service time distribution. For the moment, we do not need to establish neither the queue discipline nor the number of servers. Finally, we assume that the maximum number of clients in the system is  $E$ . For this system we count the number of clients in the system,  $X$ , and we wish to calculate the rejection probability, that is to say,  $P(X = E)$ .

In the following we show how the RESTART method works. We divide the maximum number of clients  $E$  into  $m + 1$  levels,  $I_i$ , verifying  $I_0 = 0$ ,  $I_m = E$  and  $I_{i-1} < I_i \ i = 1, \dots, m$ . Then, we define the events  $A_i$  in the following way,  $A_i = \{w / X(w) \geq I_i\}$ ,  $i = 1, \dots, m$ .

In this point, two different strategies are possible, basically step-by-step approach and global approach. The step-by-step approach works taking into account a set  $A_i$  each time and the global approach all steps are simulated simultaneously. Another possible classification divides these methods in fixed effort and fixed splitting RESTART [6]. In the first type, the number of samples for each  $A_i$  is fixed and in the second one, each trajectory that reaches an  $A_i$  level is split in a prefixed number of new trajectories.

Another possibility is founded in the combination of RESTART method with LRE method, RESTART/LRE method. LRE can be used to control systematically the number of trials needed in a simulation run by a formula which depends on a maximum relative error preestablished [10] [11]. In this case, it is mandatory to take the sample in special instants of time. In this case, the observations are made just before any new client arrives at the system. Under these assumptions of the observation times, a discrete-state Markov chain can be built, the state-transition diagram is shown in figure 1. The number of clients in the system,  $X$ , can be increased one by one, and can be decreased in any amount (maintaining  $X \geq 0$ ).

The original RESTART implementation [12] [13] corresponds to a global fixed split approach. In this section we describe a generic step-by-step approach like in [5].

The algorithm accomplishes  $m$  stages. The first stage

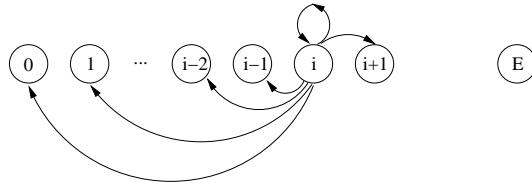


Fig. 1. State-transition diagram

calculates the value  $p_1 = P(X \geq I_1 / X \geq I_0) = P(X \geq I_1)$ . The simulation process evolves and when a new customer arrives at the system, the following values are updated:  $n_1$ , the number of customer arrivals until this moment;  $h_1$ , the number of times that, just before the entrance of a client, the system already had  $I_1$  or more customers.

If the number of clients in the system when a new customer arrives is  $I_1$ , the state of the system will be stored (number of clients, remaining service times, and so on).

When the first stage is finished, we estimate the probability  $p_1$  by means of the expression  $\hat{p}_1 = h_1/n_1$ .

Now, we consider a generic stage  $i$  of the RESTART algorithm. We randomly pick out a stored system state from the stage  $i-1$ , and we continue the simulation until we detect that the number of customers in the system is at most  $I_{i-1}$ . If so, we randomly take another stored system state and the process is repeated. As it was done in the first stage, when a customer arrives, the variable  $n_i$  is increased by one; if the current number of clients is greater than or equal to  $I_i$ , the variable  $h_i$  is increased by one; and if the number of clients is exactly equal to  $I_i$ , the system state is stored for the next stage.

When the process finishes, the value of  $p_i$  is estimated by means of  $\hat{p}_i = h_i/n_i$ . When all the stages are considered, the rare event probability  $\gamma = P(X = E) = P(X \geq E) = P(X \geq I_m)$  is estimated by means of

$$\hat{\gamma} = \hat{p}_1 \times \hat{p}_2 \times \dots \times \hat{p}_{m-1} \times \hat{p}_m$$

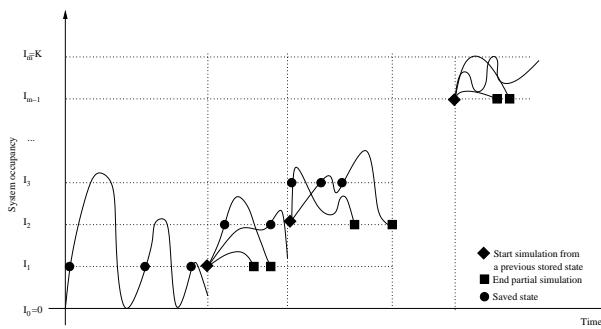


Fig. 2. step-by-step RESTART

### III. OPTIMIZATION OF QUEUING SERVICE CAPACITY

#### A. Definition of the problem

Dimensioning a queuing system means to determine the capacity of service and the size of the buffer that accomplish certain quality of service criteria. We consider the case in which the quality criterion is given by a small probability  $\gamma_0$  of rejecting a customer because the system is full. Thus, we wish to minimize the dimensions of the system that meet this rejection probability. First we consider the case in which the dimension of the system is determined by the maximum number of customers  $C$  that simultaneously can be in the system. Then our problem is

$$\text{Minimize } C \text{ subject to } P(X \geq C) \leq \gamma_0$$

where  $X$  is the state variable number of customers in the system. Note that in some real applications, like in telecommunication systems or in reliability, the probability  $\gamma_0$  is very small, of order  $10^{-7}$  or smaller.

We will use the following result for regenerative processes (see [7]).

*Theorem 1:* Let  $\{Z(t)\}_{t \geq 0}$  be a regenerative process with state space  $\{0, 1, 2, \dots\}$  and right continuous sample paths with left limits. Let  $S_1$  be a regenerative epoch and  $U_j$  be the time that the process spends in state  $j$  during  $[0, S_1]$ . If  $S_1$  is aperiodic with  $E(S_1) < \infty$ ,

$$\lim_{t \rightarrow \infty} P(Z(t) = j) = \frac{E(U_j)}{E(S_1)}$$

#### B. The algorithm

As a general idea, the algorithm works as follows. The first step is to define an initial dimension for the system based on a theoretical analysis for a similar but simpler system. Once the capacity is determined, we simulate the system to assess the quality of service. To estimate the associated probability we use the classical RESTART method to speed up the simulation. Depending on the results, we modify the capacity of the system: we increase the capacity of the system, if the probability is too large; or we decrease the capacity of the system, if the probability is too small. In order to determine the new rejection probability, we propose a method that updates the probabilities estimated in the previous step, saving a lot of simulation effort. Again, the new estimated probability is compared with the target probability and as a consequence, the capacity of the system is redefined. In the successive steps, the estimated probability will be closer to the target probability and then, its corresponding capacity will be

closer to the optimum. A stopping criterion is used to end the procedure.

**Step 1. Determination of an initial capacity for the system.** In order to get the optimal gain of the RESTART method, several authors [6] have set that the optimal number of intermediate levels is  $m \approx -\log(\gamma_0)/2$  and that the successive levels verify that the probability of reaching one from the previous is approximately  $e^{-2}$ . The theoretical analysis of queuing systems such us  $M/M/s/k$  or  $M/M/s/\infty$  provides us with these optimal levels for a given probability. We find the value  $C_1$  which is a function of the number  $E_1$  verifying  $P(Z = E_1) \approx \gamma_0$ . The variable  $Z$  is the number of customers in the system selected as a reference. We also determine the optimal values for the parameters of the RESTART method if we were to apply them to estimate the probability  $\gamma_0$  in the reference system. That is, we also find the optimal value for the number  $m$  of intermediate levels and the optimal values  $I_1, I_2, \dots, I_m$  for each one.

**Step 2. Use the classical RESTART procedure to estimate the probability  $\hat{\gamma}_1 = P(X = E_1)$ .** Applying the classical RESTART method we estimate the conditional probabilities  $p_i^{(1)} = P(X \geq I_i / X \geq I_{i-1})$  by using

$$\hat{p}_i^{(1)} = \frac{h_i^{(1)}}{n_i^{(1)}}$$

To apply the next steps of the algorithm we need to find a lower bound for the capacity of the system, that is to say, we have to determine a value  $I_{inf}$  verifying that  $P(X = I_{inf} / E = I_{inf}) \geq \gamma_0$ . Thus, the optimal capacity of the system,  $I^*$ , associated to the probability  $\gamma_0$  will satisfy  $I^* \geq I_{inf}$ .

We propose a method to estimate such level  $I_{inf}$  which is based in the following two results.

*Proposition 1:* If the product  $\prod_{i=1}^j \hat{p}_i^{(1)} < \gamma_0$ , then  $\hat{\gamma}_1 = \hat{P}(X = E) < \gamma_0$ .

*Proposition 2:* If the product

$$\prod_{i=1}^j \frac{\hat{p}_i^{(1)} - \prod_{k=i}^j \hat{p}_k^{(1)}}{1 - \prod_{k=i}^j \hat{p}_k^{(1)}} \geq \gamma_0$$

then  $E = I_j$  is an estimated lower bound for the optimal level, that is,  $\hat{P}(X = I_j / E = I_j) \geq \gamma_0$ .

Then, we can simulate the system using the RESTART procedure to successively estimate the probabilities  $\hat{p}_i^{(1)}$  and in each level  $j$  evaluate the expression  $\prod_{i=1}^j \frac{\hat{p}_i^{(1)} - \prod_{k=i}^j \hat{p}_k^{(1)}}{1 - \prod_{k=i}^j \hat{p}_k^{(1)}}$ . We will estimate the lower level  $I^*$  by  $I_j$ , where  $j$  corresponds to the maximum index for which the previous expression is greater than  $\gamma_0$ .

Because it is not necessary a high precision in the determination of the value of  $I_{inf}$ , we will not spend too much simulation effort in this first application of the RESTART method. Now we set  $E = I_j$  and  $m = j$  and apply again, with these new parameters, the RESTART method to estimate more accurately the probabilities  $p_i^{(1)}$ .

Then, the estimated probability of rejecting a customer when the capacity is  $E_1$  is

$$\hat{\gamma} = \hat{p}_1^{(1)} \times \hat{p}_2^{(1)} \times \dots \times \hat{p}_m^{(1)}$$

**Step 3. Stopping criterion.** If  $\hat{\gamma} \geq \gamma_0$ , the probability of system overflow has been so high that we have to increase the capacity of service. We do  $m = m + 1$ ,  $E_1 = E_2$ , we load the new values  $p_i^{(2)}$  in  $p_i^{(1)}$  variables and we determine the new threshold,  $E_2 = I_{m+1}$ . A simple way is to use a linear approximation defining the value of  $I_{m+1}$  as

$$I_{m+1} = I_m + \frac{(I_m - I_{m-1})(\hat{\gamma} - \gamma_0)}{\hat{\gamma}/\hat{p}_m^{(2)}}.$$

When we execute this step for the first time, only the value  $I_{m+1}$  has to be calculated.

If  $\hat{\gamma} < \gamma_0$ , then we will discard the last iteration. If  $I_{m+1} = 1 + I_m$  the algorithm finishes, the last threshold  $E_1$  is the solution. If  $I_{m+1} > 1 + I_m$ , we calculate a new  $I_{m+1}$ . Similarly to the other case we can consider the value:

$$I_m + \frac{(I_{m+1} - I_m)(\gamma_0 - \hat{\gamma})}{\hat{\gamma}/\hat{p}_{m+1}^{(2)}}.$$

If this value for  $I_{m+1}$  is equal to  $I_m$ , then we set  $I_{m+1} = I_m + 1$ . Finally, we remove all the data of the previous iteration and go to step 4.

**Step 4 Estimation of the new  $p_{m+1}^{(2)}$ .** At this moment, we execute a new stage of the RESTART method. We are going to estimate  $p_{m+1}^{(2)} = P(X \geq I_{m+1} / X \geq I_m) = P(X \geq E_2 / X \geq E_1)$ , we collect the appropriate values:  $\omega_{m+1}^{(2)}$ , the number of hits in level  $I_{m+1} = E_2$ ;  $\nu_{m+1}^{(2)}$ , the number of runs of hits in level  $E_2$ ;  $n_{m+1}$ , the number of trials;  $N_{m+1}$ , the number of cycles. We get

$$\hat{p}_{m+1}^{(2)} = \frac{\omega_{m+1}^{(2)}}{n_{m+1}}$$

**Step 5. Estimation of probabilities  $p_i^{(2)}$**   $\forall i = 1, \dots, m$ .

The  $p_i^{(2)}$ ,  $i = 1, \dots, m-1$  are updated according to:

$$\hat{p}_i^{(2)} = \frac{\hat{p}_i^{(1)} + A_i}{1 + A_i}$$

and

$$\hat{p}_{m-1}^{(2)} = \frac{\nu_m^{(1)} \hat{E}(L_{(E_1, E_2)})}{(n_m - \omega_m^{(1)}) + \nu_m^{(1)} \hat{E}(L_{(E_1, E_2)})}$$

where

$$A_i = \prod_{j=i}^{m-1} \hat{p}_j^{(1)} \left( \frac{\hat{E}(L_{(E_1, E_2)}) \nu_m^{(1)} - \omega_m^{(1)}}{n_m} \right)$$

and

$$\hat{E}(L_{(E_1, E_2)}) = \frac{n_{m+1}}{N_{m+1}}$$

We consider the following elements: a run of hits is defined as a sequence of states in the top level  $E_1$  preceded and followed by states different from  $E_1$ ;  $\nu_i^{(1)}$  is the number of runs of hits to level  $E_1$  in the  $N_i$  trajectories starting in level  $I_{i-1}$ ;  $h_i^{(1)}$  and  $\omega_i^{(1)}$  are, respectively, the total number of trials greater than or equal to level  $I_i$  and the number of hits to level  $E_1$  observed in the  $N_i$  simulated trajectories starting in level  $I_{i-1}$ ;  $E(L_{(E_1, E_2)})$  is the expected length of a trajectory starting and ending in  $E_1$  with top level  $E_2$ .

With these elements we can establish the result used in the above updating.

*Proposition 3:* Given  $\hat{p}_i^{(1)}$ ,  $i = 1, \dots, m$ ,  $\nu_m^{(1)}$ ,  $\omega_m^{(1)}$ ,  $n_m$  and  $\hat{E}(L_{(E_1, E_2)})$ , then for  $i = 1, \dots, m-1$

$$\hat{p}_i^{(2)} = \frac{\hat{E}(U_{i1}) + \hat{E}(U_{i2})}{\hat{E}(S_i)} = \frac{\hat{p}_i^{(1)} - A_i}{1 - A_i}$$

and

$$\hat{p}_{m-1}^{(2)} = \frac{\nu_m^{(1)} \hat{E}(L_{(E_1, E_2)})}{(n_m - \omega_m^{(1)}) + \nu_m^{(1)} \hat{E}(L_{(E_1, E_2)})}$$

where

$$A_i = \prod_{j=i}^{m-1} \hat{p}_j^{(1)} \left( \frac{\hat{E}(L_{(E_1, E_2)}) \nu_m^{(1)} - \omega_m^{(1)}}{n_m} \right)$$

**Step 6.** Go to step 3.

### C. An illustrative example

Let us consider the following example. A system receives packets of data that must be processed by an only server that works at constant rate  $C = 2$ . The arrival of data at the system occurs according to a Modulated Markov Poisson Process (MMPP) with two states with parameters  $\lambda_1 = 2$ ,  $\lambda_2 = 1$  and transition probabilities matrix equal to

$$P = \begin{pmatrix} 0.9 & 0.1 \\ 0.5 & 0.5 \end{pmatrix}$$

In our MMPP model, the number of packets arrived by unit time follows a Poisson distribution of rate  $\lambda_1 = 2$  when the system is in state 1, and follows a Poisson distribution with rate  $\lambda_2 = 1$  when the system state is 2. In each time unit, the state can remain or change to another state according to the probabilities established in  $P$ . The packets that arrive at the system and could not be served, are placed in a buffer with finite capacity  $B$ . The aim is to determine the capacity of the buffer (multiple of 10; 10, 20, 30 and so on) to have a probability of saturating the buffer,  $\gamma$ , lower than or equal to  $10^{-5}$ .

We have considered a simplified version of our algorithm, the initial capacity (buffer) of the system, steps 1 and 2, is established to 10 packets, and a two level RESTART is executed to get the first  $\hat{\gamma}_1$ . In step 3, the increments of  $I[m]$  are of ten units each time.

We have executed the algorithm 10 times with 10 different seeds (for the simulation) and we have generated  $10^6$  trials for each level, that is to say,  $2 \times 10^6$  trials for steps 1 and 2, and  $10^6$  trials for each new level  $I[m+1]$ . In Table I the minimum value for the buffer, 70, and the guaranteed probability are showed.

TABLE I  
RESULTS OBTAINED WITH THE METHOD PROPOSED

Buffer size	Probability
70	$5.7666 \times 10^{-6}$
70	$5.4633 \times 10^{-6}$
70	$5.1146 \times 10^{-6}$
70	$5.2696 \times 10^{-6}$
70	$5.3188 \times 10^{-6}$
70	$5.3784 \times 10^{-6}$
70	$5.4476 \times 10^{-6}$
70	$5.2750 \times 10^{-6}$
70	$5.4247 \times 10^{-6}$
70	$5.7615 \times 10^{-6}$

The confidence interval for the probability obtained with these data is  $[5.273439763 \times 10^{-6}, 5.570580237 \times 10^{-6}]$ . To get this precision, we have used 8 million trials: two million for the initial RESTART and one million for each one of the new levels (20, 30, ..., 70).

To solve the same problem using classic RESTART, we have to solve successive problems considering  $E_1 = 10$ ,  $E_1 = 20$  until reaching the value of 70.

Table II shows the probabilities obtained when considering directly a buffer size of 70, and a RESTART with two levels and with  $10^6$  trials in each level. In this case, the number of trials required is 2 million for each RESTART execution, because we need 7 executions for reaching the level 70, the total number of trials is 14 million. Let us observe that this quantity is almost twice as much as the initial computational effort.

TABLE II  
RESULTS OBTAINED USING RESTART

Probability
$6.0709 \times 10^{-6}$
$5.3372 \times 10^{-6}$
$3.4351 \times 10^{-6}$
$5.5160 \times 10^{-6}$
$4.2647 \times 10^{-6}$
$5.4391 \times 10^{-6}$
$4.1832 \times 10^{-6}$
$4.0846 \times 10^{-6}$
$4.5652 \times 10^{-6}$
$3.3703 \times 10^{-6}$

The confidence interval obtained with these data is  $[3.96649387 \times 10^{-6}, 5.28676613 \times 10^{-6}]$ . Considering these data, we have to point out that the effort that should be done when applying successive steps of RESTART is greater than the effort done with the method we propose (approx. 2 to 1). In fact, the number of trials that should be considered in each execution to get a similar precision to the one obtained with our method should be greater, and this will imply a higher computational effort.

#### IV. APPLICATIONS AND CONCLUDING REMARKS

Particular interest has received in teletraffic engineering where the discrete event simulation has become in an indispensable tool for performance evaluation of modern telecommunication systems. The new telecommunication networks pose extreme requirements about the quality of service, being the cell loss probabilities in ATM (Asynchronous Transfer Mode) networks in the order of  $10^{-9}$ . Traffic in telecommunications networks exhibit long range dependence which can be explained because heavy tailed distributions are involved in the system. Because of this, also the rare event simulation with heavy tails has received attention. Our idea is to adapt our algorithm to the optimization of systems where heavy tailed distributions appear. In fact, the aggregation of ON-OFF sources with heavy tailed distributions for the ON and/or OFF periods behaves like a Fractional Brownian Motion [14]. Because of this, also the rare event simulation in systems where heavy tailed distributions are involved is currently being investigated [3] [4]. The authors also met these questions when studying a real telecommunication problem, analyzing the dimensioning of an antenna for an international mobile telecommunications company. Then it turns natural to apply the algorithm that we present in this paper to the optimization of such systems which include heavy tailed distributions.

Other important systems where speed up simulation+optimization has a niche are the reliability

systems. We can think of a lot of systems where the correct operation of a machine is vital: big mainframes in banks, hospitals, failure detection systems in nuclear power plants, and so on. A failure in such kind of systems will become a disaster and then, this probability must be very small. These systems are so complex that analytical methods are not applicable and so, it is necessary the use of simulation models that require speed up techniques in order to estimate the failure probabilities in a reasonable computer time.

Usually, the decision-maker has to decide how many “devices” set to guarantee that the failure probability is smaller than certain security limit. This situation generates directly an optimization problem, the decision-maker has to determine the minimum amount of a collection of resources in order to guarantee the probability. For example, in the case of a important mainframe, the decision-maker could have to decide how many supply powers to connect, how many hard disk/network cards to install in order to replicate information, how many UPS’s systems, etc. But that is not all, in real systems, in general, the decision-maker would have more than one objective, not only the failure probability but the money, for example. In real systems, problems tend to have a multiobjective character.

When multiobjective problems are to be considered the usual optimization algorithms do no work; multiobjective problems do not have one solution that optimize all the objectives simultaneously and it is necessary to use approximations to the problem, determination of efficient solutions, lexicographic optimization, weighted functions, and so on.

If we join all these issues, complex systems that require speed up simulation techniques in order to evaluate their performance and multiple objectives that require special approximations, the complexity to solve the problem increases exponentially, because of this, a future line of work is to study models where more than one objective are selected, in [1] a multiobjective optimization+simulation model was developed in a industrial context.

As conclusions of the work we have to point out that the developed method is a doubly speed up technique for optimization. First, because it uses the RESTART ideas in order to calculate in a faster way the objective probability associated to a capacity level. Second, because when we need to re-evaluate with another capacity level, the method only needs to execute one new stage of the RESTART method: the work corresponding to a new threshold. In a direct application a complete execution of RESTART will be necessary for each new level, obviously the computational effort saved is very

important.

## APPENDIX

*Proposition 1:* If the product  $\prod_{i=1}^j \hat{p}_i^{(1)} < \gamma_0$ , then  $\hat{\gamma}_1 = \hat{P}(X = E) < \gamma_0$ .

*Proof:* Observe that  $\hat{\gamma}_1 = \hat{P}(X = E) = \prod_{i=1}^m \hat{p}_i^{(1)} < \prod_{i=1}^j \hat{p}_i^{(1)} < \gamma_0$  ■

*Proposition 2:* If the product

$$\prod_{i=1}^j \frac{\hat{p}_i^{(1)} - \prod_{k=i}^j \hat{p}_k^{(1)}}{1 - \prod_{k=i}^j \hat{p}_k^{(1)}} \geq \gamma_0$$

then  $E = I_j$  is an estimated lower bound for the optimal level, that is,  $\hat{P}(X = I_j / E = I_j) \geq \gamma_0$ .

*Proof:* We consider the conditional probabilities with the new top level  $E = I_j$ ,  $p_i^{(2)} = P(X \geq I_i / X \geq I_{i-1} \wedge E = I_j)$ . Using the theorem 1 for the regenerative process formed by the sequence of simulated trajectories starting at level  $I_{i-1}$  and top level  $E = I_j$  the following result holds:

$$p_i^{(2)} = \frac{E(U_i)}{E(S_i)} = \frac{E(U_{i1}) + E(U_{i2})}{E(S_{i1}) + E(S_{i2})}$$

where

- $E(U_{i1})$  is the expected number of trials per cycle in states  $[I_i, I_j]$
- $E(U_{i2})$  is the expected number of trials per cycle in the state  $I_j$
- $E(S_{i1})$  is the expected number of trials per cycle in states  $[I_{i-1}, I_j]$
- $E(S_{i2})$  is the expected number of trials per cycle in the state  $I_j$

Observe that  $E(U_{i2}) = E(S_{i2})$ , and then

$$\frac{E(U_{i1})}{E(S_{i1})} < \frac{E(U_{i1}) + E(U_{i2})}{E(S_{i1}) + E(S_{i2})}$$

Thus, we define

$$\begin{aligned} \hat{p}_i^{(2)} &= \frac{\hat{E}(U_i)}{\hat{E}(S_i)} = \frac{\hat{E}(U_{i1}) + \hat{E}(U_{i2})}{\hat{E}(S_{i1}) + \hat{E}(S_{i2})} \\ \hat{p}_i^{(*)} &= \frac{\hat{E}(U_{i1})}{\hat{E}(S_{i1})} \end{aligned}$$

For the estimation of  $E(U_{i1})/E(S_{i1})$  only the trials that take a value strictly lower than level  $I_j$  are involved. Observe that in this case, when we are restricted to states below  $I_j$ , the trajectories, in both cases when the top level is  $I_j$  and when the top level is  $E_1$ , are stochastically

equivalents. Then, the expected number of trials in a cycle, starting at  $I_{i-1}$ , over level  $I_i$  and strictly below  $I_j$ ,  $E(U_{i1})$ , can be estimated by the probability that a trial is in this range multiplied by the expected number of trials in a cycle,  $E(S)$ , when the top level is  $E_1$ .

$$\begin{aligned} \hat{E}(U_{i1}) &= \hat{P}(X \geq I_i, X < I_j / \\ &\quad X \geq I_{i-1}, E = E_1) \times E(S) = \\ &= \left( \hat{p}_i^{(1)} - \prod_{k=i}^j \hat{p}_k^{(1)} \right) \times E(S) \end{aligned}$$

Similar argument gives

$$\begin{aligned} \hat{E}(S_{i1}) &= \hat{P}(X < I_j / X \geq I_{i-1}, E = E_1) \times E(S) \\ &= \left( 1 - \prod_{k=i}^j \hat{p}_k^{(1)} \right) \times E(S) \end{aligned}$$

And therefore,

$$\hat{p}_i^{(*)} = \frac{\hat{E}(U_{i1})}{\hat{E}(S_{i1})} = \frac{\hat{p}_i^{(1)} - \prod_{k=i}^j \hat{p}_k^{(1)}}{1 - \prod_{k=i}^j \hat{p}_k^{(1)}}$$

Then, readily

$$\prod_{i=1}^j \hat{p}_i^{(2)} \geq \prod_{i=1}^j \hat{p}_i^{(*)} = \prod_{i=1}^j \frac{\hat{p}_i^{(1)} - \prod_{k=i}^j \hat{p}_k^{(1)}}{1 - \prod_{k=i}^j \hat{p}_k^{(1)}} \geq \gamma_0$$

*Proposition 3:* Given  $\hat{p}_i^{(1)}$ ,  $i = 1, \dots, m$ ,  $\nu_m^{(1)}$ ,  $\omega_m^{(1)}$ ,  $n_m$  and  $\hat{E}(L_{(E_1, E_2)})$ , then for  $i = 1, \dots, m-1$

$$\hat{p}_i^{(2)} = \frac{\hat{E}(U_{i1}) + \hat{E}(U_{i2})}{\hat{E}(S_i)} = \frac{\hat{p}_i^{(1)} - A_i}{1 - A_i}$$

and

$$\hat{p}_{m-1}^{(2)} = \frac{\nu_m^{(1)} \hat{E}(L_{(E_1, E_2)})}{(n_m - \omega_m^{(1)}) + \nu_m^{(1)} \hat{E}(L_{(E_1, E_2)})}$$

where

$$A_i = \prod_{j=i}^{m-1} \hat{p}_j^{(1)} \left( \frac{\hat{E}(L_{(E_1, E_2)}) \nu_m^{(1)} - \omega_m^{(1)}}{n_m} \right)$$

*Proof:*

We use the theory of regenerative processes, concretely the result set in the previous theorem, to update all the probabilities. It is clear that

$$p_i^{(2)} = P(X \geq I_i / X \geq I_{i-1} \wedge E = E_2) = \frac{E(U_i^{(2)})}{E(S_i^{(2)})}$$

where the numerator,  $E(U_i^{(2)})$ , is the expected number of trials taking value greater than or equal to level  $I_i$  in a cycle, that is, in a trajectory starting in level  $I_{i-1}$  and

ending the first time that, again, it is observed the level  $I_{i-1}$ . The denominator,  $E(S_i^{(2)})$ , is the expected value for the length of a cycle, measured in number of trials. We separate the numerator into two terms,  $E(U_i^{(2)}) = E(U_{i1}^{(2)}) + E(U_{i2}^{(2)})$ . The first one,  $E(U_{i1}^{(2)})$ , is the expected number of trials greater than or equal to level  $I_i$  and strictly below the level  $E_1$ ; meanwhile the second term,  $E(U_{i2}^{(2)})$ , represents the expected number of trials greater than or equal to level  $E_1$ .

Clearly,

$$\hat{E}(U_{i1}^{(2)}) = (h_i^{(1)} - \omega_i^{(1)})/N_i$$

The second term can be expressed as

$$\hat{E}(U_{i2}^{(2)}) = \frac{\nu_i^{(1)}}{N_i} \hat{E}(L_{(E_1, E_2)})$$

Because the observation of  $E_1$  can remain as a rare event for the lower levels  $I_i$ , it could be quite usual to have  $\nu_i^{(1)}/N_i = 0$ . In these cases the probabilities would remain unchanged, although we know that they would not. To avoid this problem we can substitute this ratio for

$$\frac{\nu_i^{(1)}}{N_i} \approx \frac{\nu_m^{(1)}}{n_m} \frac{n_i}{N_i} \prod_{j=i}^{m-1} \hat{p}_j^{(1)} = \frac{\nu_m^{(1)}}{n_m} \hat{E}[S_i^{(1)}] \prod_{j=i}^{m-1} \hat{p}_j^{(1)}$$

Observe that  $\prod_{j=i}^{m-1} \hat{p}_j^{(1)}$  represents the probability that a trial belonging to a trajectory starting in level  $I_{i-1}$  is over the level  $I_{m-1}$ ;  $\frac{n_i}{N_i} = \hat{E}[S_i^{(1)}]$  is the average number of trials per cycle starting in  $I_{i-1}$ ; and finally,  $\frac{\nu_m^{(1)}}{n_m}$  represents the ratio of runs of hits per trial in trajectories over  $I_{m-1}$ .

Then,

$$\hat{E}(U_{i2}^{(2)}) = \hat{E}(L_{(E_1, E_2)}) \frac{\nu_m^{(1)}}{n_m} \hat{E}[S_i^{(1)}] \prod_{j=i}^{m-1} \hat{p}_j^{(1)}$$

Using the same arguments as before,  $\hat{E}(U_{i1}^{(2)})$  can be expressed as

$$\begin{aligned} \hat{E}(U_{i1}^{(2)}) &= \frac{(h_i^{(1)} - \omega_i^{(1)})}{N_i} = \\ &= \hat{p}_i^{(1)} \hat{E}(S_i^{(1)}) - \frac{\omega_m^{(1)}}{n_m} \hat{E}(S_i^{(1)}) \prod_{j=i}^{m-1} \hat{p}_j^{(1)} \end{aligned}$$

Reasoning in a similar way we obtain the following expression for  $E(S_i^{(2)})$ :

$$\begin{aligned} \hat{E}(S_i^{(2)}) &= \frac{n_i - \omega_i^{(1)}}{N_i} + \frac{\nu_i^{(1)}}{N_i} \hat{E}(L_{(E_1, E_2)}) \\ &= \hat{E}(S_i^{(1)}) - \frac{\omega_m^{(1)}}{n_m} \hat{E}(S_i^{(1)}) \prod_{j=i}^{m-1} \hat{p}_j^{(1)} + \\ &\quad \hat{E}(L_{(E_1, E_2)}) \frac{\nu_m^{(1)}}{n_m} \hat{E}[S_i^{(1)}] \prod_{j=i}^{m-1} \hat{p}_j^{(1)} \end{aligned}$$

If we join the above expressions, we get the proposition. ■

## REFERENCES

- [1] Alberto, I., Azcárate, C., Mallor, F. and Mateo, P.M., "Optimization with simulation and multiobjective analysis in industrial decision-making: A case study". *European Journal of Operational Research*, Vol. 140, pp. 373-383, 2002.
- [2] Asmussen, S., "Applied Probability and Queues". Springer Verlag, 2nd ed. 2003.
- [3] Asmussen, S., Binswanger, K., Hojgaard, B., "Rare events simulation for heavy tailed distributions". *Bernoulli*, Vol. 6, No. 2, pp. 303-322, 2000.
- [4] Asmussen, S., Kroese, D.P., Rubinstein, R.Y., "Heavy tails, importance sampling and cross entropy". *Stochastic Models*, Vol. 21, pp. 57-76, 2005.
- [5] Below, K., Battaglia, L. and Killat, U., "RESTART/LRE Simulation: The reliability issue". *Proceedings of the Second International Workshop on Rare Event Simulation*, 1999.
- [6] Garvels, M.J.J. and Kroese, D.P., "A comparison of RESTART implementations". *Proceedings of the 1998 Winter Simulation Conference*, Ed. D.J. Medeiros, E.F. Watson, J.S. Carson and M.S. Manivannan, pp. 601-608, 1998.
- [7] Kulkarni, V. G., "Modeling and Analysis of Stochastic Systems", Chapman Hall, 1995.
- [8] Ross, D. and Harris, C.M., "Fundamentals of queueing theory". John Wiley and Sons, Singapore, 1985.
- [9] Rubinstein, R.Y. and Melamed, B., "Modern Simulation and Modeling". John Wiley and Sons, 1998.
- [10] Schreiber, F., "Effective control of simulation runs by a new evaluation algorithm for correlated random sequences". *AEÜ*, Vol. 42, pp. 347-354, 1988.
- [11] Schreiber, F. and Görg, C., "Stochastic Simulation: a simplified LRE-Algorithm for Discrete Random Sequences *AEÜ*", Vol. 50, pp. 233-239, 1996.
- [12] Villén-Altamirano, M., Villén-Altamirano, J., "RESTART: A method for accelerating rare events simulations". *Proceedings of the 13th International Teletraffic Congress, Queueing performance and control in ATM.*, Ed. J. W. Cohen and C.D. Pack, pp. 71-76, North-Holland, 1991.
- [13] Villén-Altamirano, M., Villén-Altamirano, J., "RESTART: A straightforward method for fast simulation of rare events". *Proceedings of the 1994 Winter Simulation Conference*, Ed. J. D. Tew, S. Manivannan, D.A. Sadowski and A.F. Seila., pp. 282-289, 1994.
- [14] Willinger, W., Taqqu, M., Sherman, R. and Wilson, D., "Self-similarity through high-variability: statistical analysis of ethernet LAN traffic at the source level". *IEEE ACM Transactions on Networking*, Vol. 5, No. 1, pp. 71-86, 1997.

# Exact Algorithms for Procurement Problems under a Total Quantity Discount Structure

D.R. Goossens\*, A.J.T. Maas†, F.C.R. Spieksma‡ and J.J. van de Klundert§

\*Department of Applied Economics, Katholieke Universiteit Leuven, Belgium

Email: dries.goossens@econ.kuleuven.be

†TRIKS BV, Maastricht, The Netherlands

‡Department of Applied Economics, Katholieke Universiteit Leuven, Belgium

§Department of Mathematics, Maastricht University, The Netherlands

**Abstract**—In this paper, we study the procurement problem faced by a buyer who needs to purchase a variety of goods from suppliers applying a so-called total quantity discount policy. This policy implies that every supplier announces a number of volume intervals and that the volume interval in which the total amount ordered lies determines the discount. Moreover, the discounted prices apply to all goods bought from the supplier, not only to those goods exceeding the volume threshold. We refer to this cost-minimization problem as the total quantity discount (TQD) problem. We give a mathematical formulation for this problem and argue that not only it is NP-hard, but also that there exists no polynomial-time approximation algorithm with a constant ratio (unless  $P = NP$ ). Apart from the basic form of the TQD problem, we describe four variants. In a first variant, the market share that one or more suppliers can obtain is constrained. Another variant allows the buyer to procure more goods than strictly needed, in order to reach a lower total cost. We also consider a setting where the buyer needs to pay a disposal cost for the extra goods bought. In a third variant, the number of winning suppliers is limited, both in general and per product. Finally, we investigate a multi-period variant, where the buyer not only needs to decide what goods to buy from what supplier, but also when to do this, while considering the inventory costs. We show that the TQD problem and its variants can be solved by solving a series of min-cost flow problems. Finally, we investigate the performance of three exact algorithms (min-cost flow based branch-and-bound, linear programming based branch-and-bound, and branch-and-cut) on randomly generated instances involving 50 suppliers and 100 goods. It turns out that even the large instances of the basic problem are solved to optimality within a limited amount of time. However, we find that different algorithms perform best in terms of computation time for different variants.

**Keywords**—procurement, volume discounts, exact algorithm, complexity, min-cost flow, reverse auction

## I. INTRODUCTION

IT is a widespread economic phenomenon that the price of a good depends - among many other things - on the amount ordered. Indeed, there are many reasons for suppliers to offer discounts based on the volume sold to a buyer. Consequently, when it comes to procuring amounts of different goods from different suppliers, it makes sense to consider various alternatives. In fact, choosing the right suppliers to deliver the right products has become a major concern in many large companies. Reliability, quality, and price are important criteria that guide the choice for suppliers. Moreover, the ever-increasing opportunities that e-commerce and web-based procurement offer for dealing with procurement issues, explain the increased usage of so-called reverse auctions. While traditional auctions involve a single seller and multiple buyers, a reverse auction involves multiple sellers that express bids to provide goods or services and one buyer that chooses the best bids.

In this work, we investigate a basic procurement problem from the viewpoint of a buyer who faces different suppliers that offer a variety of goods using specific discount policies. The discount policy we investigate is one where the supplier has specified a number of volume intervals, and the price per good depends on the volume interval in which the total amount ordered lies. Obviously, a supplier is assumed not to increase its prices in a higher interval. This structure is called total quantity discount (TQD). Furthermore, the prices apply to all units bought from the supplier, which is called an all-unit discount policy (a discussion and classification of various quantity discount policies can be found in Munson and Rosenblatt (1998)). We assume that a preselection of suppliers has been made, excluding those suppliers who do not attain the required standards with respect to quality, reliability and other relevant considerations (see Degraeve et

al. (2000) for a discussion of these considerations). Thus, we assume that the only remaining criterion upon which the further supplier selection decision is based, is the price these suppliers charge for the different goods. Given a final demand for each good, the TQD problem is to satisfy demand against minimal cost.

Procurement problems involving discount policies have been studied by many authors. Katz et al. (1994) (see also Sadrian and Yoon (1994)) discuss a procurement problem where they distinguish between purchases on a commitment basis and purchases on an as-ordered basis. They stress the importance of sourcing flexibility and model explicitly the fact that not all future goods should be purchased via committed contracts. In addition, they explicitly consider the number of vendors for each good, and the percentages of the total supply given to each of the vendors. In their discount policy, a supplier discounts the price of each good by the same percentage based on the total dollar value of all goods purchased from the supplier, whereas our policy allows a different discount percentage for each good.

Crama et al. (2004) investigate another procurement problem, characterized by a discount policy very similar to the one used here, in the sense that it also expresses the discount as a function of the total quantity of goods purchased. However, it also differs since it uses one single discount rate for all products. Furthermore, Crama et al. face the additional problem of deciding how to use the purchased raw materials to manufacture the desired quantities of the endproducts.

Austin and Hogan (1976) is an early reference to procurement problems characterized by a lower and upper bound for each supplier between which the ordered amount needs to lie, provided that that supplier is used. In this paper, the government needs to purchase a given amount of aviation fuel from one or more suppliers, where prices differ depending on how the fuel is transported. This problem differs from our setting in that the goods considered are independent and there are no discounts. The authors solve the problem using a branch-and-bound algorithm, exploiting the network structure of the core problem.

The TQD problem can also be viewed in the context of combinatorial auctions. Combinatorial auctions are relevant when the value of a set of goods is not equal to the sum of the values of the individual goods. Then there are so-called complementary or substitute-effects, and in such a setting it can be beneficial to consider

pricing sets of goods instead of pricing only individual goods. The discount policy described above is a way to price a set of goods: the cardinality of the set of all goods ordered determines in which interval the buyer is, and the all-unit discount policy leads to prices that imply complementary effects.

Bichler et al. (2004) outline a classification of allocation problems based on the number of participants and the type of traded goods. According to this classification, the TQD problem is an  $n$ -bilateral allocation problem, since there are only two types of participants, i.e., buyers and sellers. In our case, there is only one buyer, which makes it a single-sided auction. Furthermore, the TQD problem is characterized by single-attribute, multi-item, multi-unit bids, because bids can be made on any quantity of a number of heterogeneous goods and all other attributes besides the price are predefined.

Davenport and Kalagnanam (2002) report on a volume discount auction in which discounts are based on quantities for each individual good. Furthermore, they use an incremental discount policy, meaning that the discounts apply only to the additional units above the threshold of the volume interval. Hohner et al. (2003) describe a web-based implementation of this procurement auction at Mars Incorporated.

Eso et al. (2001) also elaborate on the work of Davenport and Kalagnanam. They study a volume discount auction with piece-wise linear supply curves, allowing discontinuities and all-unit discounts. However, they do require additive separable supply curves, which boils down to assuming that the prices charged by a supplier for different commodities are independent. This makes their problem not truly combinatorial, since synergies or substitutability between different goods cannot be reflected in the total price charged by the suppliers. As a result, a total quantity discount structure is not possible in their setting. The authors formulate a column generation based heuristic that provides near-optimal solutions to the bid evaluation problem.

Another procurement auction with marginal-decreasing piecewise-constant supply curves is described in Kothari et al. (2003). This auction also allows all-unit discounts, but it deals only with a single good. Kothari et al. present fully polynomial-time approximation schemes for the winner determination problem and the computation of the corresponding payments of this auction.

The TQD problem is also related to the so-called deal splitting problem introduced by Shachnai et al. (2004). In this problem, a buyer needs to split an order of multiple units from a set of heterogeneous goods among a set of sellers, each having bounded amounts of the goods, so as to minimize the total cost of the deal. Two variants of the deal splitting problem can be discerned, depending on whether the seller offers packages containing combinations of the goods or whether the buyer can generate such combinations using seller-specified price tables. Shachnai et al. show that for both variants an exact solution can be found in pseudo-polynomial time if the number of heterogeneous goods is fixed. Moreover, they develop polynomial-time approximation schemes for several subclasses of instances of practical interest.

We now describe shortly the practical application that originally motivated this problem (see Van de Klundert et al. (2003)). Consider a telecommunication company that needs to acquire capacity to accommodate its international calls. This capacity is offered by various so-called carriers, i.e., for each destination, each carrier offers capacity, priced in eurocents per minute. Prices of carriers differ, and - which is particularly relevant for our setting - each carrier uses an interval structure to arrive at a certain price. In other words, the total amount of call-minutes handled by a certain carrier determines the price. The problem is to acquire the right amount of capacity for each destination at minimal cost.

We give the following results. We show that no polynomial-time algorithm for the TQD problem can achieve a constant worst-case ratio (unless  $P = NP$ ); this contrasts with the case of a single good for which Chauhan et al. (2005) established NP-completeness and gave a fully polynomial time approximation scheme. Then, we prove that (a generalization of) the linear programming relaxation of a straightforward formulation of the problem can be solved by min-cost flow. Thus, we prove that a combinatorial algorithm solves the LP-relaxation of the TQD problem. Furthermore, we extend the basic TQD problem incorporating market share constraints, the more-for-less paradox, limits to the number of winning suppliers, and a multi-period perspective with inventory costs. Finally, we perform computational experiments comparing three exact algorithms: a min-cost flow based branch-and-bound approach (using the network solver of Ilog Cplex 8.1), a linear programming based branch-and-bound approach (using the MIP solver of Ilog Cplex 8.1) and

a branch-and-cut approach (also using the MIP solver of Ilog Cplex 8.1). Section II presents the mathematical formulation of our problem, section III describes the theoretical results, and section IV presents four variants of the TQD problem. In section V, the exact algorithms for the TQD problem and its variants are described and finally section VI gives our computational results.

## II. MATHEMATICAL FORMULATION

To state a mathematical formulation of the TQD problem, we use the following notation. We define  $G$  as the set of  $m$  goods, indexed by  $k$ , and  $S$  as the set of  $n$  suppliers, indexed by  $i$ . For each good  $k$  in  $G$ , we define  $d_k$  as the amount of good  $k$  to be procured. To each supplier  $i$  in  $S$  we associate a sequence of intervals  $Z_i = \{0, 1, \dots, max_i\}$ , indexed by  $j$ . Furthermore, for each supplier  $i \in S$  and interval  $j \in Z_i$ ,  $l_{ij}$  and  $u_{ij}$  define the minimum and maximum number of goods respectively that needs to be ordered from supplier  $i$  to be in interval  $j$ . Finally, for each supplier  $i \in S$ , for each interval  $j \in Z_i$  and each good  $k \in G$ , let  $c_{ijk}$  be the price for one item of good  $k$  purchased from supplier  $i$  in its  $j$ -th interval.

We assume that these parameters satisfy the following assumptions:

$$\forall i \in S, j \neq j' \in Z_i : [l_{ij}, u_{ij}] \cap [l_{ij'}, u_{ij'}] = \emptyset, \quad (1)$$

$$\forall i \in S, j \in Z_i \setminus \{max_i\}, k \in G : c_{ijk} \geq c_{i,j+1,k}, \quad (2)$$

$$\forall i \in S, j \in Z_i, k \in G : c_{ijk} \geq 0, l_{ij} \geq 0, u_{ij} \geq 0, d_k \geq 0. \quad (3)$$

Assumption (1) states that a supplier's intervals should not overlap. The requirement that prices should not increase from one interval to the next is expressed in the second assumption. The last assumption reflects that all prices and all quantities ordered are nonnegative.

We define the decision variable  $x_{ijk}$  as the amount of good  $k$  purchased from supplier  $i$  in interval  $j$ . Further, we define a binary decision variable  $y_{ij}$  which is 1 if interval  $j$  is selected for supplier  $i$  and 0 otherwise. This leads to the following formulation of the TQD problem, referred to as TQDF.

minimize

$$\sum_{i \in S} \sum_{j \in Z_i} \sum_{k \in G} c_{ijk} x_{ijk} \quad (4)$$

subject to

$$\sum_{i \in S} \sum_{j \in Z_i} x_{ijk} = d_k \quad \forall k \in G \quad (5)$$

$$\sum_{j \in Z_i} y_{ij} \leq 1 \quad \forall i \in S \quad (6)$$

$$\sum_{k \in G} x_{ijk} - y_{ij} l_{ij} \geq 0 \quad \forall i \in S, j \in Z_i \quad (7)$$

$$\sum_{k \in G} x_{ijk} - y_{ij} u_{ij} \leq 0 \quad \forall i \in S, j \in Z_i \quad (8)$$

$$x_{ijk} \geq 0 \quad \forall i \in S, j \in Z_i, k \in G \quad (9)$$

$$y_{ij} \in \{0, 1\} \quad \forall i \in S, j \in Z_i \quad (10)$$

The objective function (4) states that the amount of goods  $k$  ordered from supplier  $i$  when in interval  $j$ , times the corresponding price must be minimal. Constraints (5) make sure that the demand for each good is met, while constraints (6) guarantee that at most one interval per supplier is selected. Constraints (7) and (8) ensure that if an interval  $j$  is selected, the total amount of goods purchased from supplier  $i$  is between the bounds of that interval. If an interval  $j$  is not selected, these constraints ensure that  $x_{ijk} = 0$ . Constraints (9) state that only a nonnegative amount can be purchased, while constraints (10) define  $y$  as a boolean variable. Notice that this formulation allows to order nothing from a supplier. Notice also that we do not require integrality of the  $x$ -variables; if the demands and the lower and upper bounds of each volume interval are integral however, then, assuming the existence of a feasible solution, there always exists an optimal solution of TQDF with integral  $x$ -values (see section III).

Let us now discuss how this formulation relates to known classes of integer programming formulations. The TQD problem is related to fixed charge network flow problems (see [13]). In fact, when omitting constraints (6) from the formulation above, the resulting problem can be formulated as a (special) fixed charge network flow problem. Indeed, when one builds a network involving a source with supply  $\sum d_k$ , a 'demand' node for each good  $k$  with demand  $d_k$ , and an 'interval' node for each interval of each supplier, the variable  $x_{ijk}$  in the formulation above represents nothing else but the flow on the arc from an 'interval' node to a 'demand' node. In particular, this implies that inequalities that are valid for this formulation of the fixed charge network flow problem are also valid for TQDF. However, due to the presence of constraints (6), the TQDF formulation is more general than a fixed charge network flow problem. Notice, though, that in the objective function (4), there

is no fixed cost associated to choosing some interval of some supplier, i.e., in terms of the fixed charge network flow problem, the fixed cost of using an arc is 0.

Finally, one can view the TQD problem as a direct generalization of the ordinary, well-known, transportation problem: given a set of demand nodes, each with demand  $d_k$ , given a set of supply nodes each with a supply between a given lower bound  $l_j$  and upper bound  $u_j$ , given costs per good for each combination of demand node and supply node, and finally, given a collection of subsets of the supply nodes such that at most one node of each subset is allowed to supply a positive amount, find a solution of minimum cost. TQD belongs to this class of generalized transportation problem; as far as we are aware, this problem has not been investigated before. A special case of this generalized transportation problem where for each demand node  $k$ , pairs of supply nodes are given such that at most one supply node of each pair is allowed to supply demand node  $k$  is studied by Sun (2002).

### III. PROPERTIES OF THE TQD PROBLEM

In this section we establish the complexity of the TQD problem (section III-A). We also show that the LP-relaxation of TQDF can be solved by solving a min-cost flow problem (section III-B).

#### A. On the complexity of the TQD problem

We show that the TQD problem is a hard problem to solve when aiming for optimal solutions.

*Theorem 1:* The decision version of the TQD problem is strongly NP-complete.

In fact, we can also make the following statement on the approximability of the TQD problem:

*Theorem 2:* No polynomial-time approximation algorithm with constant worst-case ratio exists for the TQD problem (unless  $P = NP$ ).

Next, consider the TQD problem, where - instead of prices for all intervals for each supplier - only prices for the first interval and a discount rate is given. This discount rate  $\delta$  determines the price  $c_{i,j,k}$  of good  $k$  in interval  $j$  as a function of the price in interval  $j - 1$  as follows:

$$c_{ijk} = (1 - \delta)c_{i,j-1,k} \quad \forall i, k \text{ and } \forall j > 1 \quad (11)$$

We claim that this special case of the TQD problem is still a hard problem.

*Theorem 3:* The decision version of the TQD problem with a common discount rate  $\delta$  is strongly NP-complete.

Finally, consider the variant of the TQD problem where the amounts purchased must be *at least as large* as the demands  $d_k$ . In such a setting, it might happen that buying more than what is strictly needed reduces the total cost. We refer to this problem as the more-for-less variant of the TQD problem (see section IV-B). For the special case of this variant where only one good needs to be purchased, Chauhan et al. (2005) showed that there exists a fully polynomial time approximation scheme. We claim that this variant remains a hard problem.

*Theorem 4:* The decision version of the more-for-less variant of TQD problem is strongly NP-complete.

For the proofs of Theorems 1, 2, 3, and 4, we refer to the appendix.

### B. Min-cost flow and the TQD problem

We now show that the LP-relaxation of TQDF can be solved by solving a min-cost flow problem. In fact, even in the more general case where for some suppliers intervals are prespecified, the LP-relaxation of the resulting model can still be found by solving a min-cost flow problem.

Let us first state a model which assumes that for an arbitrary given subset of suppliers, referred to as  $D$  ( $D \subseteq S$ ), an interval, say  $s(i) \in Z_i$ , has been selected, while for the remaining suppliers no interval has been selected. We refer to the following formulation as GENTQDF.

minimize

$$\sum_{i \in S} \sum_{j \in Z_i} \sum_{k \in G} c_{ijk} x_{ijk} \quad (12)$$

subject to

$$\sum_{i \in S} \sum_{j \in Z_i} x_{ijk} = d_k \quad \forall k \in G \quad (13)$$

$$\sum_{j \in Z_i} y_{ij} \leq 1 \quad \forall i \in S \setminus D \quad (14)$$

$$\sum_{k \in G} x_{ijk} - y_{ij} l_{ij} \geq 0 \quad \forall i \in S \setminus D, j \in Z_i \quad (15)$$

$$\sum_{k \in G} x_{ijk} - y_{ij} u_{ij} \leq 0 \quad \forall i \in S \setminus D, j \in Z_i \quad (16)$$

$$\sum_{k \in G} x_{i,s(i),k} - l_{i,s(i)} \geq 0 \quad \forall i \in D \quad (17)$$

$$\sum_{k \in G} x_{i,s(i),k} - u_{i,s(i)} \leq 0 \quad \forall i \in D \quad (18)$$

$$x_{ijk} \geq 0 \quad \forall i \in S \setminus D, j \in Z_i, k \in G \quad (19)$$

$$x_{ijk} = 0 \quad \forall i \in D, j \neq s(i), k \in G \quad (20)$$

$$0 \leq y_{ij} \leq 1 \quad \forall i \in S \setminus D, j \in Z_i \quad (21)$$

Observe that if  $D = \emptyset$ , the resulting model is the LP-relaxation of TQDF, whereas if  $D = S$ , we arrive at the situation where an interval has been selected for each supplier (see [10]). Introducing  $D$  allows us to develop an enumeration approach based algorithm, solving only min-cost flow problems (see section V).

*Theorem 5:* GENTQDF can be polynomially transformed to min-cost flow.

**PROOF.** We organize the proof by first showing that an optimal solution of GENTQDF has a structural property. Then we construct a min-cost flow instance and show the correspondence between optimal solutions of this instance and GENTQDF.

**Claim:** There exists an optimal solution  $(x^*, y^*)$  of GENTQDF in which for each  $i \in S \setminus D$ :

$$\begin{aligned} x_{ijk}^* &= 0 \quad \forall j \neq \max_i, \forall k \in G, \text{ and} \\ y_{ij}^* &= 0 \quad \forall j \neq \max_i. \end{aligned} \quad (22)$$

Thus, the claim states that there exists an optimal solution in which all  $x$ - and  $y$ -variables equal 0, except those corresponding to the highest interval of each supplier. In other words, goods are bought only at the lowest prices of each supplier.

**Argument:** given some feasible solution  $(x, y)$  of GENTQDF, we show how to modify  $(x, y)$  to  $(x^*, y^*)$  such that  $(x^*, y^*)$  is a feasible solution of GENTQDF satisfying (22) and such that the cost of  $(x^*, y^*)$  does not exceed the cost of  $(x, y)$ .

For each  $k \in G$  and each  $i \in S \setminus D$ , we set

$$x_{i,max_i,k}^* = \sum_{j=0}^{max_i} x_{ijk}, \text{ and} \quad (23)$$

$$x_{ijk}^* = 0 \quad \text{for } j = 0, 1, \dots, max_i - 1. \quad (24)$$

Further, for each  $i \in S \setminus D$ , we set

$$y_{i,max_i}^* = y_{i,max_i} + \frac{\sum_{j=0}^{max_i-1} \sum_{k \in G} x_{ijk}}{u_{i,max_i}}, \text{ and} \quad (25)$$

$$y_{ij}^* = 0 \quad \text{for } j = 0, 1, \dots, max_i - 1. \quad (26)$$

All other variables remain the same, that is

$$x_{ijk}^* = x_{ijk} \quad \forall i \in D, j \in Z_i, k \in G. \quad (27)$$

It is obvious that the costs of  $(x^*, y^*)$  cannot exceed the costs of  $(x, y)$  since the total amount of goods has remained the same for each supplier, while in  $(x^*, y^*)$  all goods are purchased in the highest interval (and we have  $c_{i,max_i,k} \leq c_{ijk} \forall i, j, k$ , see (1)). Let us now argue that  $(x^*, y^*)$  is a feasible solution of GENTQDF.

Evidently,  $(x^*, y^*)$  satisfies (13), (17), (18), (19) and (20). To show that  $(x^*, y^*)$  satisfies (14) and (21), we need to show that  $y_{i,max_i}^* \leq 1$  for  $i \in S \setminus D$ . Observe that for  $j = 0, 1, \dots, max_i - 1$  we have  $\sum_{k \in G} x_{ijk}/u_{ij} \leq y_{ij}$  (using the feasibility of  $(x, y)$  with respect to (16)) and thus  $\sum_{k \in G} x_{ijk}/u_{i,max_i} \leq y_{ij}$  for  $j = 0, 1, \dots, max_i - 1$ . Summing over  $j = 0, 1, \dots, max_i - 1$  implies that  $\sum_{j=0}^{max_i-1} (\sum_{k \in G} x_{ijk})/u_{i,max_i} \leq \sum_{j=0}^{max_i-1} y_{ij}$  and together with the feasibility of  $(x, y)$  with respect to (14) this leads to  $(x^*, y^*)$  satisfying (14) and (21).

Consider now for some  $i \in S \setminus D$  constraints (15), written alternatively as  $\sum_{k \in G} x_{ijk} \geq l_{ij} y_{ij}$  for  $j = 0, 1, \dots, max_i$ . In case  $j < max_i$ , the right-hand side equals 0 (since  $y_{ij}^* = 0$  for  $j < max_i$  by construction) and feasibility follows. In case  $j = max_i$ , we have, using feasibility of  $(x, y)$ , that

$$\sum_{k \in G} x_{i,max_i,k} \geq l_{i,max_i} y_{i,max_i}. \quad (28)$$

Also it is true that

$$\sum_{j=0}^{max_i-1} \sum_{k \in G} x_{ijk} \geq \frac{\sum_{j=0}^{max_i-1} \sum_{k \in G} x_{ijk}}{u_{i,max_i}} l_{i,max_i}. \quad (29)$$

Summing (28) and (29) yields:

$$\begin{aligned} \sum_{k \in G} x_{i,max_i,k}^* &= \sum_{k \in G} (x_{i,max_i,k} + \sum_{j=0}^{max_i-1} x_{ijk}) \\ &\geq l_{i,max_i} (y_{i,max_i} + \sum_{j=0}^{max_i-1} \sum_{k \in G} \frac{x_{ijk}}{u_{i,max_i}}) \\ &= l_{i,max_i} y_{i,max_i}^*. \end{aligned} \quad (30)$$

Thus  $(x^*, y^*)$  satisfies constraints (15).

To verify that  $(x^*, y^*)$  satisfies constraints (16), observe that for  $i \in S \setminus D$  and for  $j = 0, 1, \dots, max_i - 1$   $\sum_{k \in G} x_{ijk}^* = 0$  and  $y_{ij}^* = 0$  (this follows by construction of  $x^*$  and  $y^*$ ). Finally, in case  $j = max_i$  we have

$$\sum_{k \in G} x_{i,max_i,k} \leq u_{i,max_i} y_{i,max_i}, \text{ and} \quad (31)$$

$$\sum_{j=0}^{max_i-1} \sum_{k \in G} x_{ijk} = \frac{\sum_{j=0}^{max_i-1} \sum_{k \in G} x_{ijk}}{u_{i,max_i}} u_{i,max_i}. \quad (32)$$

Summing (31) and (32) yields

$$\begin{aligned} \sum_{k \in G} x_{i,max_i,k}^* &= \sum_{k \in G} (x_{i,max_i,k} + \sum_{j=0}^{max_i-1} x_{ijk}) \\ &\leq u_{i,max_i} (y_{i,max_i} + \sum_{j=0}^{max_i-1} \sum_{k \in G} \frac{x_{ijk}}{u_{i,max_i}}) \\ &= u_{i,max_i} y_{i,max_i}^*, \end{aligned} \quad (33)$$

which shows that constraints (16) are also satisfied by  $(x^*, y^*)$  and allows us to conclude that  $(x^*, y^*)$  is indeed a feasible solution of GENTQDF.

Let us now build the network. We have three sets of nodes: there is a node for each supplier (a ‘supplier node’), there is a node for each good (a ‘good node’) and there is a single source node. The supply of the source node equals  $\sum_{k \in G} d_k$  and the demand of each good node equals  $d_k$ . All other demands are 0. Furthermore, there is an arc from the source node to each supplier node. If this supplier is in  $D$ , the corresponding lower and upper bounds of this arc are  $l_{i,s(i)}$  and  $u_{i,s(i)}$ ; if this supplier is not in  $D$ , the lower and upper bounds are 0 and  $u_{i,max_i}$ . (The choice for a lower bound of 0 for suppliers not in  $D$ , even if  $l_{i,0}$  is strictly positive, may seem surprising at first sight. It can however be verified that because the  $y$ -values are relaxed in GENTQDF,  $l_{i,0}$  no longer constrains the  $x$ -values.) The cost of an arc between the source node and each supplier node equals 0. There are also arcs from each supplier node to each good node. These arcs are not constrained by lower or upper bounds, but do

have a cost equal to  $c_{i,s(i),k}$  if the corresponding supplier is in  $D$  and equal to  $c_{i,max_i,k}$  if this supplier is not in  $D$ . This completes the description of the min-cost flow instance. A schematic representation is given in Figure 1.

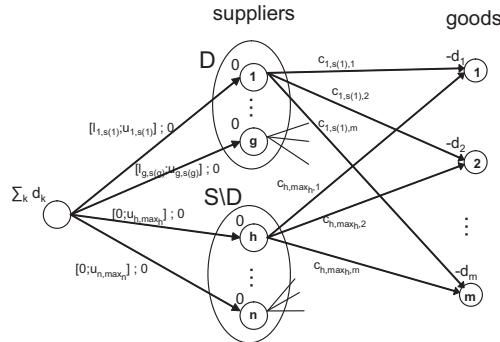


Fig. 1. GENTQDF as min-cost flow

A solution of this min-cost flow instance is characterized by flows  $f_{ik}$  on each arc from supplier  $i$  to good  $k$ . It corresponds to a solution of GENTQDF as follows:

$$x_{i,s(i),k} = f_{ik} \quad \forall i \in D, k \in G, \quad (34)$$

$$x_{i,max_i,k} = f_{ik} \quad \forall i \notin D, k \in G, \quad (35)$$

$$y_{i,s(i)} = 1 \quad \forall i \in D, \quad (36)$$

$$y_{i,max_i} = \sum_{k \in G} \frac{f_{ik}}{u_{i,max_i}} \quad \forall i \notin D. \quad (37)$$

All other  $x$ - and  $y$ -variables of GENTQDF are set equal to 0.

Given (22), we conclude that an optimal solution of the min-cost flow problem in Figure 1 corresponds to an optimal solution of GENTQDF. It can now easily be seen that an optimal solution of GENTQDF also corresponds to an optimal flow in the min-cost flow problem. Thus, we have shown how GENTQDF can be polynomially transformed to min-cost flow.  $\square$

Notice that as a consequence of Theorem 5, the LP-relaxation of TQDF can be found by solving a min-cost flow problem. This result is the foundation for an exact algorithm to be discussed in section VI.

#### IV. VARIANTS OF THE TQD PROBLEM

When procuring goods, other considerations besides the price can be relevant. Although our model does not incorporate criteria like quality or reliability, we now consider a number of variants of the TQD problem

that are common in both practice and literature. A first variant adds constraints on the amount of goods the buyer is willing to purchase from a supplier (section IV-A). In another variant (section IV-B), the buyer is allowed to buy more goods than strictly needed, while the third variant (section IV-C) imposes a restriction on the number of winning suppliers (suppliers that end up selling some amount of any of the goods are called winning suppliers). Finally, a variant that incorporates a multi-period perspective with inventory costs is described (section IV-D). We show that results similar to that of Theorem 5 hold for each of these variants.

##### A. Market share constraints

Suppose that the buyer wants to impose upper and/or lower bounds on the amount of a good that must be ordered from a supplier. Forcing that some supplier  $i$  must be allocated an amount of at least  $q_{ik}$  and at most  $Q_{ik}$  of good  $k$  can be done by adding the following constraint to GENTQDF:

$$q_{ik} \leq \sum_{j \in Z_i} x_{ijk} \leq Q_{ik}. \quad (38)$$

On a more global level the buyer could provide bounds on the total allocation for a supplier, across all goods. Forcing the total amount of goods purchased from a supplier  $i$  to lie between  $w_i$  and  $W_i$  can be done by adding the following constraint to GENTQDF:

$$w_i \leq \sum_{j \in Z_i} \sum_{k \in G} x_{ijk} \leq W_i. \quad (39)$$

These market share constraints are often mentioned in literature (see [5], [7], [8], and [9]). Notice that none of these extra constraints invalidate property (22). Constraints (38) can easily be implemented in the min-cost flow graph by changing the lower and upper bounds of the arcs from supplier  $i$  to good  $k$ . Constraints (39) can be realized via the lower and upper bounds of the arcs from the root node to supplier  $i$ . Thus, we obtain the following statement:

*Theorem 6:* GENTQDF with constraints (38) and/or (39) can be polynomially transformed to min-cost flow.

##### B. More-for-less

As described in section III-A, it can be advantageous to obtain more of some good  $k$  than the required amount  $d_k$ , since this might allow the buyer to use the cheaper

prices of a higher interval (see also [4] and [14]). If we wish to allow this, constraints (13) in GENTQDF should be replaced by

$$\sum_{i \in S} \sum_{j \in Z_i} x_{ijk} \geq d_k \quad \forall k \in G. \quad (40)$$

Notice that for the special case where  $D = \emptyset$ , all units are already bought in the highest intervals in an optimal solution of GENTQDF (see (22)). Therefore, there is no need to buy more than  $d_k$  of any good  $k$  and an optimal solution can be found by solving the min-cost flow problem in Figure 1. In general however, we can formulate the following result:

*Theorem 7:* GENTQDF with constraints (13) replaced by (40) can be polynomially transformed to min-cost flow.

**PROOF.** Consider the graph in Figure 2. It has supplier and good nodes, with demands and connecting arcs like in Figure 1. The lower and upper bounds and the costs for these arcs are the same as in Figure 1 but in order not to overload the figure, they have been omitted. There is however also a dummy node, corresponding to the additional goods that are bought once the demand  $d_k$  is fulfilled. The dummy node has a demand of  $M$ , being at least  $\sum_{i \in D} l_{i,s(i)}$ . The supply of the source node is increased by this same amount  $M$ . Furthermore, there is an arc from the source node to the dummy node with cost 0 and an upper bound of  $M$ . Notice that any flow in the network in Figure 1 is still a feasible flow in the network in Figure 2. There are also arcs from each supplier  $i \in D$  to the dummy node. These arcs have a cost equal to the price of the supplier's cheapest good in its selected interval  $s(i)$ . In Figure 2, we refer to this good as  $q(i)$ , i.e.  $q(i) = \arg \min_k c_{i,s(i),k}$ . Notice that this is the good we will buy additionally from that supplier to reach the threshold of a higher interval; it would be pointless to buy a more expensive good instead to achieve this. There are no arcs to the dummy node from suppliers not in  $D$ . Since for these suppliers the goods are already bought at their lowest prices (see (22)), there is no use in buying additional goods.

Observe that in GENTQDF it can happen that because of the interval selections made for suppliers in  $D$ , no feasible solution exists. This is the case if the demands  $d_k$  are not high enough to reach the required lower bounds of the selected intervals. In the more-for-less variant of GENTQDF, however, this is no longer possible since it is allowed to buy more than the amounts  $d_k$ . Indeed,

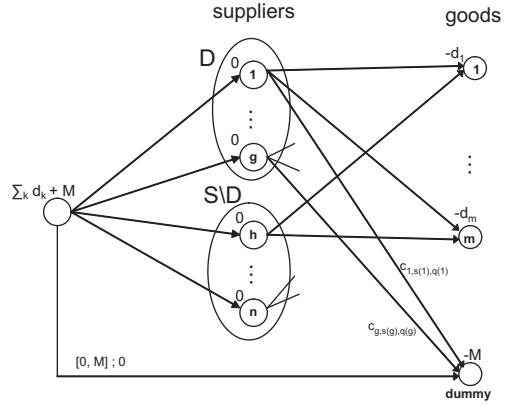


Fig. 2. GENTQDF with more-for-less as min-cost flow

these extra amounts correspond to the flows on the arcs from suppliers in  $D$  to the dummy node. If we refer to the flow from a supplier  $i$  to the dummy node as  $f_{id}$ , then a solution of the min-cost flow model in Figure 2 corresponds to a solution of GENTQDF with constraints (13) replaced by (40) as follows:

$$x_{i,s(i),k} = f_{ik} \quad \forall i \in D, k \in G \setminus \{q(i)\}, \quad (41)$$

$$x_{i,s(i),q(i)} = f_{i,q(i)} + f_{id} \quad \forall i \in D, \quad (42)$$

$$x_{i,max_i,k} = f_{ik} \quad \forall i \notin D, k \in G, \quad (43)$$

$$y_{i,s(i)} = 1 \quad \forall i \in D, \quad (44)$$

$$y_{i,max_i} = \sum_{k \in G} \frac{f_{ik}}{u_{i,max_i}} \quad \forall i \notin D. \quad (45)$$

□

Until now, we implicitly made the assumption that the buyer can simply buy more than what is demanded and enjoy a higher discount without any further consequence. However, as described in [4], in practice, overbuying often leads to an extra cost for the buyer. The buyer may for instance need extra storage capacity. Furthermore, the buyer may not be able to use the additional goods as profitably as the goods of the original demand, or even be forced to pay a cost for the disposal of these goods. Let us assume that  $p_k$  is this non-negative cost the buyer incurs for each additional unit of good  $k$ , in addition to the purchasing cost. Let us define  $x'_{ijk}$  as the amount of good  $k$  that is bought in addition to the demand in the  $j$ -th interval of supplier  $i$ . We can now generalize more-for-less GENTQDF as follows:

minimize

$$\sum_{i \in S} \sum_{j \in Z_i} \sum_{k \in G} (c_{ijk} x_{ijk} + (c_{ijk} + p_k) x'_{ijk}) \quad (46)$$

subject to

$$\sum_{i \in S} \sum_{j \in Z_i} x_{ijk} = d_k \quad \forall k \in G \quad (47)$$

$$\sum_{j \in Z_i} y_{ij} \leq 1 \quad \forall i \in S \setminus D \quad (48)$$

$$\sum_{k \in G} (x_{ijk} + x'_{ijk}) - y_{ij} l_{ij} \geq 0 \quad \forall i \in S \setminus D, j \in Z_i \quad (49)$$

$$\sum_{k \in G} (x_{ijk} + x'_{ijk}) - y_{ij} u_{ij} \leq 0 \quad \forall i \in S \setminus D, j \in Z_i \quad (50)$$

$$\sum_{k \in G} (x_{i,s(i),k} + x'_{i,s(i),k}) - l_{i,s(i)} \geq 0 \quad \forall i \in D \quad (51)$$

$$\sum_{k \in G} (x_{i,s(i),k} + x'_{i,s(i),k}) - u_{i,s(i)} \leq 0 \quad \forall i \in D \quad (52)$$

$$x_{ijk} \geq 0 \quad \forall i \in S \setminus D, j \in Z_i, k \in G \quad (53)$$

$$x'_{ijk} \geq 0 \quad \forall i \in S \setminus D, j \in Z_i, k \in G \quad (54)$$

$$x_{ijk} = x'_{ijk} = 0 \quad \forall i \in D, j \neq s(i), k \in G \quad (55)$$

$$0 \leq y_{ij} \leq 1 \quad \forall i \in S \setminus D, j \in Z_i \quad (56)$$

Consider a min-cost flow network like the one in Figure 2, but with the difference that the cost on the arcs from supplier  $i \in D$  to the dummy  $d$  node equals  $c_{i,s(i),q(i)} + p_{q(i)}$ , with

$$q(i) = \arg \min_k (c_{i,s(i),k} + p_k). \quad (57)$$

Let us now argue how a solution of this min-cost flow network corresponds to a solution of generalized more-for-less GENTQDF. It is clear that for suppliers not in  $D$ , it remains pointless to buy any additional good, since the buyer can already get the lowest possible price by ordering in the highest intervals. Notice that property (22) thus remains valid. For suppliers for which an interval has been prespecified, it can be necessary to buy additional goods, namely if the demands  $d_k$  are insufficiently high to reach the lower bounds of the selected intervals. In this case, the buyer will obviously buy the cheapest additional good, namely the good for which  $c_{i,s(i),k} + p_k$  is minimal. Notice that this is exactly how we defined  $q(i)$ . It is now easy to see that a solution  $f$  of the min-cost flow network corresponds to a solution of generalized more-for-less GENTQDF

as follows:

$$x_{i,s(i),k} = f_{ik} \quad \forall i \in D, k \in G, \quad (58)$$

$$x_{i,max_i,k} = f_{ik} \quad \forall i \notin D, k \in G, \quad (59)$$

$$x'_{i,s(i),q(i)} = f_{id} \quad \forall i \in D, \quad (60)$$

$$y_{i,s(i)} = 1 \quad \forall i \in D, \quad (61)$$

$$y_{i,max_i} = \sum_{k \in G} \frac{f_{ik}}{u_{i,max_i}} \quad \forall i \notin D. \quad (62)$$

All other  $x$ -,  $x'$ - and  $y$ -variables are set equal to 0. Hence we have proven the following theorem:

*Theorem 8:* The generalization of more-for-less GENTQDF can be polynomially transformed to min-cost flow.

### C. Limited number of winning suppliers

Another important consideration apart from cost minimization is to make sure that the demand is not procured from too many suppliers (see also [5], [7], [8], [9] and [14]). Otherwise, overhead costs increase due to managing this large amount of suppliers. Limiting the total number of winning suppliers can be done for the order as a whole (section IV-C.1) or per product (section IV-C.1).

1) *Limited total number of winning suppliers:* In order to model the requirement that a limited number of suppliers is selected, we need to understand exactly when a supplier receives a positive amount. This happens when  $y_{ij} = 1$  for some  $j$ , except possibly when  $j = 0$ , and  $l_{i,0} = 0$ ; the latter situation refers to the case where interval 0, with a lower bound of 0, is selected. Then a supplier might receive nothing, while there is a  $y$ -variable with a positive value. To handle this situation, we ‘split’ each interval that has a lower bound of 0 and a positive upper bound into two intervals: one interval with a lower bound and an upper bound of 0 (the dummy interval), and one interval with a lower bound of 1 and an upper bound equal to the original upper bound (interval 0). Notice that by setting this lower bound to 1, we assume that the demands and the lower and upper bounds are or can be scaled to integers. Thus, we have redefined interval 0 by excluding the option of a zero amount of goods. Moreover, we let  $y_{i,0}$  correspond to this new interval 0. Obviously, selecting a supplier’s dummy interval comes down to not selecting this supplier at all, in which case the supplier can simply be removed

from the problem. Selecting another interval of a supplier implies that this is a winning supplier. This approach leads to a set  $D$ , containing only winning suppliers. In fact, without loss of generality, we can now focus on constraining the winning suppliers not in  $D$ , and limit their number to  $K$  by adding the following constraint to GENTQDF:

$$\sum_{i \in S \setminus D} \sum_{j \in Z_i} y_{ij} \leq K. \quad (63)$$

If we assume that the highest volume interval of every supplier in  $S \setminus D$  has the same upper bound, we can prove a similar result to that of Theorem 5. We refer to this common upper bound as  $u_{max}$ . Given the fact that in most real-life applications suppliers pose no upper bound at all to the amount of goods they are willing to sell, this assumption is quite reasonable.

**Theorem 9:** If  $u_{max_i} = u_{max} \forall i \in S \setminus D$ , then GENTQDF with constraint (41) added can be polynomially transformed to a min-cost flow problem.

**PROOF.** First, notice that property (22) remains valid in this setting. Indeed, given the  $x$ -values, we can find  $y$ -values for each supplier  $i \in S \setminus D$  and each volume interval  $j \in Z_i$  satisfying constraints (15) and (16) in the following interval:

$$[\frac{\sum_{k \in G} x_{ijk}}{u_{i,j}}, \frac{\sum_{k \in G} x_{ijk}}{l_{i,j}}]. \quad (64)$$

Naturally, in order to fulfill constraints (21), the  $y$ -values cannot exceed 1. It is easy to verify that shifting goods from a supplier's highest interval to one or more lower intervals can never decrease the total  $y$ -value of this supplier. Therefore, constraint (41) will never force the optimal solution of GENTQDF away from the highest intervals and property (22) still holds.

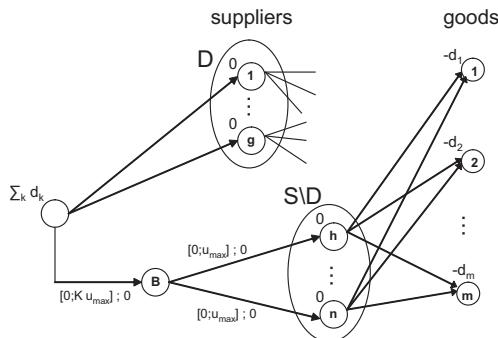


Fig. 3. GENTQDF with a limited number of winning suppliers as min-cost flow

We can now construct a min-cost flow network (see Figure 3). Compared to Figure 1, an extra node, referred to as node  $B$ , is added. The arc from the root node to node  $B$  has an upper bound of  $Ku_{max}$ , and the arcs from node  $B$  to the supplier nodes have upper bounds of  $u_{max}$ .

Let  $d_{min}$  be the minimal amount of goods that needs to be purchased from suppliers not in  $D$  in order to have a feasible solution, i.e.,  $d_{min} = \max(\sum_{k \in G} d_k - \sum_{i \in D} u_{i,s(i)}, 0)$ . The min-cost flow problem can only be infeasible if this demand  $d_{min}$  is too high for the upper bounds on the arcs, i.e., if  $d_{min} > Ku_{max}$ . In this case however, GENTQDF with constraint (41) is infeasible as well. Indeed, even when choosing the  $y$ -values as low as possible, namely as  $f_i/u_{max}$ , we fail to meet constraint (41):

$$\begin{aligned} \sum_{i \in S \setminus D} \sum_{j \in Z_i} y_{ij} &= \sum_{i \in S \setminus D} f_i/u_{max} \\ &\geq d_{min}/u_{max} \\ &> K. \end{aligned}$$

If there exists a feasible flow  $f$  to the min-cost flow problem, then we can always find a solution to GENTQDF with constraint (41) by setting the  $x$ - and  $y$ -variables as in (34)-(37). From Theorem 5, it is clear that this solution satisfies (13)-(21). Let  $d_{max}$  be the maximal amount of goods that can be purchased from suppliers not in  $D$  in order to keep the solution feasible, i.e.,  $d_{max} = \sum_{k \in G} d_k - \sum_{i \in D} l_{i,s(i)}$ . Obviously, a feasible flow will have  $d_{max} \leq Ku_{max}$ . Therefore, the resulting  $y$ -variables will also satisfy (41), as shown below:

$$\begin{aligned} \sum_{i \in S \setminus D} \sum_{j \in Z_i} y_{ij} &= \sum_{i \in S \setminus D} \sum_{k \in G} f_{i,k}/u_{max} \\ &\leq d_{max}/u_{max} \\ &\leq K. \end{aligned}$$

□

Notice that this proof no longer holds when each supplier  $i$  has an arbitrary value for  $u_{max_i}$ . For instance, if we set the upper bound on the arc from the source to node  $B$  equal to the sum of the  $K$  highest upper bounds, then it may happen that there exists a feasible flow  $f$  such that the corresponding  $x$ - and  $y$ -variables according to (34)-(37) are no feasible solution to GENTQDF. Indeed, consider the setting in Figure 4, assuming  $K = 1$ . A flow of 2 to node  $B$ , splitting into flows of 1 to supplier 1 and supplier 2 is feasible to

the min-cost flow model. However, its corresponding  $y$ -values in GENTQDF, 0.5 and 1 respectively, clearly violate constraint (41). Analogously, setting the upper bound of the arc to node  $B$  equal to the sum of the  $K$  lowest upper bounds results in the existence of a solution of GENTQDF for which the corresponding flow is no feasible solution of the min-cost flow model.

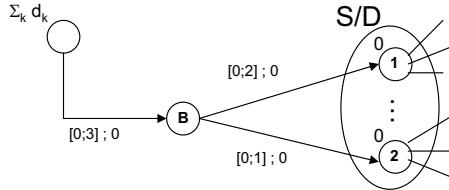


Fig. 4. Necessity of common  $u_{max_i}$

Property (22) is crucial for the possibility to use min-cost flow to solve LP-relaxations of GENTQDF-type formulations. For instance, one could also argue that the number of winning suppliers must be at least a minimum number, say  $L$ . Indeed, depending on too few suppliers could move the buyer in a vulnerable position if one of these suppliers is unable to supply as agreed. This could be encoded by adding the following constraint to GENTQDF:

$$\sum_{i \in S} \sum_{j \in Z_i} y_{ij} \geq L. \quad (65)$$

Property (22) is however no longer valid in this setting, since constraint (43) pushes the optimal solution away from the highest intervals. Indeed, moving the goods towards one or more lower intervals can increase the total  $y$ -value of each supplier. This is illustrated by the following example.

	Supplier A	Supplier B	Supplier C
Interval	1-10	1-10	1-5
Unit cost	5	1	3

Consider a setting where 14 units of one single good need to be bought from three suppliers with volume intervals and costs as indicated in the table above. Also, we wish to order from at least 2 suppliers ( $L = 2$ ). Solving GENTQDF for this example results in the following optimal solution:

$$\begin{array}{ll} x_A = 0 & y_A = 0 \\ x_B = 10 & y_B = 1 \\ x_{C1} = 0.4 & y_{C1} = 0.4 \\ x_{C2} = 3.6 & y_{C2} = 0.6 \end{array}$$

It is clear that property (22) is not valid for this solution, since it makes use of supplier C's lowest interval. Notice that a solution using only the highest intervals of suppliers B and C can never satisfy constraint (43) with  $L = 2$ . Especially the fact that the optimal solution makes use of more than one interval per supplier, prevents us from following a similar reasoning as in Theorem 5 to transform this variant to a min-cost flow problem.

### 2) Limited number of winning suppliers per good:

Suppose now that the buyer is interested in limiting the number of winning suppliers for one or more specific goods only. Forcing that good  $k$  can be supplied by at most  $Q_k$  suppliers can be done by adding the following constraints to TQDF:

$$z_{ik} \leq \sum_{j \in Z_i} x_{ijk} \leq M_{ik} z_{ik} \quad \forall i \in S, k \in G \quad (66)$$

$$\sum_{i \in S} z_{ik} \leq Q_k \quad \forall k \in G \quad (67)$$

$$z_{ik} \in \{0, 1\} \quad \forall i \in S, k \in G. \quad (68)$$

We introduced a new variable  $z_{ik}$  which is 1 if supplier  $k$  procures at least 1 unit of good  $k$  and 0 otherwise. This is guaranteed by constraints (66) and (68). In constraint (66), the parameter  $M_{ik}$  can be set equal to  $\min(d_k, u_{max_i})$ . Constraints (67) state that no more than  $Q_k$  suppliers should procure good  $k$ . We refer to TQDF with constraints (66)-(68) added as TQDF'.

When constructing GENTQDF from TQDF, we assume that for some suppliers an interval is prespecified. Also, we assume that the  $y$ -variables are relaxed. Let us now make the additional assumption that some  $z_{ik}$  variables get value 1 on beforehand, some get  $z_{ik}$  variables value 0, whilst for others no value is prespecified. When also the  $z$ -variables are relaxed, so that they can take any value between 0 and 1, this results in a relaxation of this generalization of TQDF', to which we refer as GENTQDF'.

It can easily be verified that property (22) remains valid for GENTQDF'. Indeed, also in this setting we can improve any solution that makes use of intervals other than the highest by shifting goods bought in these intervals to the highest interval. As we argued in Theorem 5, it is always possible to adjust the  $y$ -variables in such a way that the solution remains feasible. Furthermore, this shift has no influence at all on the  $z$ -variables, since the  $x$ -variables are summed over all intervals in constraint (66).

We can now construct a min-cost flow network like in Figure 1. However, for this variant, the arc from a supplier  $i$  to a good node  $k$  has a lower bound of 1 and an upper bound of  $M_{ik}$  if supplier  $i$  is chosen to be one of the  $Q_k$  suppliers that will be procuring good  $k$ . On the other hand, if supplier  $i$  is chosen not to be part of the winning suppliers for good  $k$ , the arc from node  $i$  to node  $k$  is deleted. A solution of this min-cost flow problem is characterized by flows  $f_{ik}$  on each arc from supplier  $i$  to good  $k$ . This solution corresponds to the  $x$ - and  $y$ -variables of the optimal solution of GENTQDF' as indicated in (34) to (37). The  $z$ -variables in GENTQDF' follow from the min-cost flow solution as follows:

$$z_{ik} = \frac{f_{ik}}{M_{ik}} \quad \forall i \in S, k \in G. \quad (69)$$

Indeed, constraints (67) force the  $z$ -variables towards the lowest value they can get, which is  $\sum_{j \in Z_i} x_{ijk}/M_{ik}$ . However, from property (22), it follows that  $\sum_{j \in Z_i} x_{ijk}$  equals  $x_{i,s(i),k}$  for suppliers in  $D$  and  $x_{i,max_i,k}$  for those not in  $D$ , which is exactly  $f_{ik}$  (see (34) and (35)). Thus we obtain the following statement:

*Theorem 10:* GENTQDF' can be polynomially transformed to min-cost flow.

#### D. Multi-period procurement

A lot of research on quantity discount policies has been done in the context of lot sizing problems (see e.g. Xu et al. (2000)). Lot sizing problems typically deal with when to order what amount of goods and include inventory costs. Whereas in the basic TQD problem we assumed a single-period perspective, we generalize to a multi-period procurement problem in this variant. Indeed, it no longer suffices for the buyer to decide what goods to purchase from what supplier, but the buyer also needs to decide when to order what goods, taking into account the inventory costs.

We define  $P$  as a series of  $r$  periods, indexed by  $p$ . For each good  $k$ ,  $d_{kp}$  is now the demand for good  $k$  in period  $p$ . We also define  $h_{kp}$  as the cost of holding one unit of good  $k$  in inventory at the end of period  $p$  and  $c_{ijkp}$  as the cost of purchasing one unit of good  $k$  in period  $p$  in the  $j$ -th interval of supplier  $i$ . In order to model this variant, we need to generalize the  $x$ - and  $y$ -variables with an extra index  $p$ , referring to the period in which the good is bought. We also generalize the set  $D$  to  $D_p$ , being the set of suppliers for which an interval has been prespecified for the period  $p$ . We refer to this interval as  $s(i, p)$ . We also introduce the variable  $v_{kp}$  as the inventory of good  $k$  at the end of period  $p$ . The generalized formulation, to which we refer as multi-period GENTQDF then looks as follows:

minimize

$$\sum_{i \in S} \sum_{j \in Z_i} \sum_{k \in G} \sum_{p \in P} c_{ijkp} x_{ijkp} + \sum_{k \in G} \sum_{p \in P} h_{kp} v_{kp} \quad (70)$$

subject to

$$v_{k,1} = \sum_{i \in S} \sum_{j \in Z_i} x_{i,j,k,1} - d_{k,1} \quad \forall k \in G \quad (71)$$

$$v_{kp} = v_{k,p-1} + \sum_{i \in S} \sum_{j \in Z_i} x_{ijkp} - d_{kp} \quad \forall k \in G, p \in P \quad (72)$$

$$\sum_{j \in Z_i} y_{ijp} \leq 1 \quad \forall i \in S \setminus D_p, p \in P \quad (73)$$

$$\sum_{k \in G} x_{ijkp} - y_{ijp} l_{ij} \geq 0 \quad \forall i \in S \setminus D_p, j \in Z_i, p \in P \quad (74)$$

$$\sum_{k \in G} x_{ijkp} - y_{ijp} u_{ij} \leq 0 \quad \forall i \in S \setminus D_p, j \in Z_i, p \in P \quad (75)$$

$$\sum_{k \in G} x_{i,s(i,p),k,p} - l_{i,s(i,p)} \geq 0 \quad \forall i \in D_p \quad (76)$$

$$\sum_{k \in G} x_{i,s(i,p),k,p} - u_{i,s(i,p)} \leq 0 \quad \forall i \in D_p \quad (77)$$

$$x_{ijkp} \geq 0 \quad \forall i \in S \setminus D_p, j \in Z_i, k \in G, p \in P \quad (78)$$

$$x_{ijkp} = 0 \quad \forall i \in D_p, j \neq s(i, p), k \in G, p \in P \quad (79)$$

$$0 \leq y_{ijp} \leq 1 \quad \forall i \in S \setminus D_p, j \in Z_i, p \in P \quad (80)$$

Generalizing from (22), we claim that there exists an optimal solution  $(x^*, y^*)$  of multi-period GENTQDF in which for each  $p \in P$  and for each  $i \in S \setminus D_p$ :

$$\begin{aligned} x_{ijkp}^* &= 0 \quad \forall j \neq \max_i, \forall k \in G, \text{ and} \\ y_{ijp}^* &= 0 \quad \forall j \neq \max_i. \end{aligned} \quad (81)$$

Notice that this claim can be proven in a similar way as (22).

Let us now construct a min-cost flow problem similar to the one in Figure 1, but now there are supplier nodes  $(i, p)$  for each supplier  $i$  in each period  $p$ . Also, there are good nodes for each good in each period. Each good node  $(k, p)$ , corresponding with good  $k$  in period  $p$ , has a demand of  $d_{kp}$ . The source node has a supply equal to  $\sum_{k \in G} \sum_{p \in P} d_{kp}$ . There are arcs from a supplier node  $(i, q)$  to a good node  $(k, r)$  if  $q \leq r$ . These arcs have a cost equal to  $c_{i,s(i,q),k,p} + \sum_{p=q}^{r-1} h_{kp}$  if the corresponding supplier is in  $D_q$  and equal to  $c_{i,\max_i,k,p} + \sum_{p=q}^{r-1} h_{kp}$  if this supplier is not in  $D_q$ . A schematic representation is given in Figure 5. However, in order not to overload the figure, only one supplier and one good are drawn. Also, we assume that for this supplier no interval is prespecified for any period.

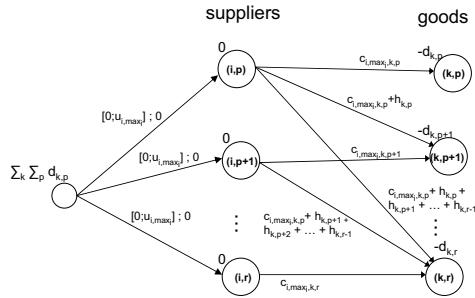


Fig. 5. Multi-period GENTQDF as min-cost flow

A solution of the min-cost flow network in Figure 5 is characterized by flows  $f_{ip}$  from the source node to each supplier node  $(i, p)$  and by flows  $f_{ipkq}$  from supplier node  $(i, p)$  to good node  $(q, r)$ . This solution can be written as a solution of multi-period GENTQDF as follows:

$$x_{i,s(i,p),k,p} = \sum_{q=p}^r f_{ipkq} \quad \forall i \in D, k \in G, p \in P, \quad (82)$$

$$x_{i,\max_i,k,p} = \sum_{q=p}^r f_{ipkq} \quad \forall i \notin D, k \in G, p \in P, \quad (83)$$

$$y_{i,s(i,p),p} = 1 \quad \forall i \in D_p, \forall p \in P, \quad (84)$$

$$y_{i,\max_i,p} = \frac{f_{ip}}{u_{i,\max_i}} \quad \forall i \notin D_p, \forall p \in P. \quad (85)$$

All other  $x$ - and  $y$ -variables of multi-period GENTQDF are set equal to 0. The  $v$ -variables can now be computed from the  $x$ -variables using (71) and (72).

*Theorem 11:* Multi-period GENTQDF can be polynomially transformed to min-cost flow.

## V. EXACT ALGORITHMS

In this section we describe the three exact algorithms used to solve instances of the TQD problem and its variants. First, we explain the min-cost flow based branch-and-bound algorithm. We build a branching tree such that in every node a min-cost flow problem needs to be solved (see Theorem 5). The branching tree is constructed in such a way that every level in the tree corresponds to a supplier, and that there is a branch for every volume interval of that supplier.

In the root node, the LP relaxation of the TQD problem is solved as explained in section III-B. For each supplier, the sum of its  $x$ -values lies between the lower and upper bound of one of its intervals, to which we refer as its LP-interval. We can now compute for each supplier its priority as the number of volume intervals minus the index of the LP-interval. Thus, suppliers that announce a lot of volume intervals but receive little in the LP-relaxation, are accorded a high priority. We use this priority to build up the search tree, as we start with the supplier with the highest priority, creating branches from the root node for each of its intervals. In the node from the first branch, we fix the LP-interval of the supplier with the highest priority. In the next branch of that level, we fix the interval directly above this interval; in the following branch and still within this level, we fix the interval directly below it and so on (provided that these intervals exist). In the following level of the branching tree we continue with the supplier with the second highest priority, again branching on its intervals as just explained, and so on (see Figure 6). Naturally, there is no need to create a node in the branching tree for a supplier with only one interval, since we can fix this interval right away. To traverse the tree, we use a standard depth-first search strategy where, as usual, a node is fathomed if its solution is dominated by the current best solution or if it is infeasible. We experimented with different priority settings, and the choice described above seems to work best. A partial explanation for this observation can be that in the current priority setting, suppliers who receive little are explored first. Given a good solution, the other

branches of this supplier should be eliminated by the resulting bound.

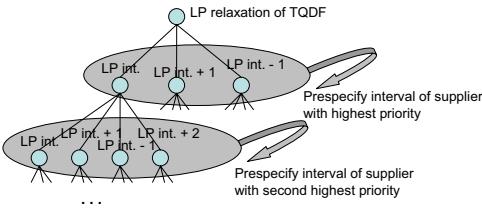


Fig. 6. Branching tree for min-cost flow based branch-and-bound

The branching tree for both the market share and the more-for-less variant is very similar. In the first variant, we prune the tree by deleting those volume intervals that fall outside the range imposed by the market-share constraints. Afterwards, we can adapt the upper and lower bounds of the highest and lowest interval respectively according to the market share constraints. As a result, the branching tree is typically sparser in the market share variant than in the basic case. In the more-for-less variant on the other hand, the branching tree is in general dense compared to its counterpart in the basic case, because less nodes are infeasible in the more-for-less setting.

The branching tree for the variant that limits the number of winning suppliers to  $K$  differs from the branching tree of the basic case, because we need to introduce an extra branch on every level of the tree. This branch corresponds to the dummy interval as introduced in section IV-C and imposes that the corresponding supplier is not to be used in the solution. Whereas suppliers with only one interval are left out of the tree completely in the basic case, they now appear in the tree with two branches, representing the decision to buy from that supplier or not. On the other hand, a node needs no further branching as soon as  $K$  suppliers have been selected. For all three variants, we use the same depth-first strategy as for the basic case.

The min-cost flow based branch-and-bound algorithm has been programmed in C and compiled using Microsoft Visual C++ 6.0. To solve the min-cost flow problems, we have used the network solver of Ilog Cplex 8.1.

The description of the other two algorithms is straightforward. The branch-and-cut algorithm simply uses the default settings of the MIP solver of Ilog Cplex 8.1. These default setting include the use of

so-called flow cover cuts that are valid for the TQD problem and its variants (see Nemhauser and Wolsey (1988)). To study these effect of the cuts, we have also investigated another algorithm in which we disallow the Ilog Cplex MIP solver to generate cuts. We refer to this algorithm as the linear programming based branch-and-bound algorithm. This algorithm uses a best-bound node-selection strategy instead of a depth-first search, but more importantly, it uses the shadow prices of the  $y$ -variables to select the branching variable at the node which has been selected for branching.

## VI. COMPUTATIONAL RESULTS

In this section we discuss the choices that were made to construct the instances on which the algorithms have been tested. We continue with computational results for the TQD problem and its variants and evaluate the performance of our algorithms.

### A. Structure of the instances

In order to test the performance of the exact algorithms, two types of instances have been generated: completely random instances and instances with a special structure, inspired by the instances studied by Van de Klundert et al. (2003). All instances have 10, 20 or 50 suppliers and 40 or 100 goods. Furthermore, each supplier has a maximum of 3 or 5 volume intervals. For all instances, the total demand for a good is a random number between 1000 and 10000. For instances with 40 goods, the upperbound-increase from one interval to the next is a random number between 10000 and 50000, while for instances with 100 goods, the upperbound-increase is a random number between 10000 and 100000.

For structured instances, we first determine a base price for each good, randomly picked between 3 and 7. The price for a good in a supplier's first interval is then computed by adding a random number in the interval  $[-2, 2]$  to the base price. Furthermore, for each supplier  $i$  there is a discount rate  $\delta_{ij} \in [0, 0.1]$  for every interval  $j > 1$ , which determines the price  $c_{i,j,k}$  of good  $k$  in interval  $j$  as a function of the price in interval  $j - 1$  as follows:

$$c_{ijk} = (1 - \delta_{ij})c_{i,j-1,k} \quad \forall i, k \text{ and } \forall j > 1 \quad (86)$$

For random instances, the cost of purchasing a good from a supplier in its first interval is a random number

between 2 and 8. The price for this good in each of the next intervals is computed by discounting the price in the previous interval by a percentage picked randomly between 0 and 75%.

The key difference between the random and the structured instances is that for the former instances prices can drop drastically from one interval to the next, whereas for the latter this decrease in price is limited to 10%. Furthermore, for the structured instances, a good that is expensive at one supplier will very likely be expensive at the other suppliers too. For the random instances however, this is not necessarily the case as prices for a good can differ in a wider range between the various suppliers. Finally, the discount percentage one receives when moving from one interval to the next can differ substantially between the goods for the random instances, while it is the same for all the goods for the structured instances.

In the variant with the market share constraints, only global constraints (as in (39)) are included. For the instances with 10 suppliers, 5 suppliers are picked randomly from each of whom between 5 and 20 percent of the total demand needs to be purchased. For instances with 20 suppliers, we pick 10 suppliers and force between 5 and 15 % of the total demand to go to each of them and for the instances with 50 suppliers this becomes 20 suppliers with each 5 to 10 % of the total demand. The more-for-less variant needs no extra modifications, apart from allowing to buy more than what is demanded. For the third variant, the number of winning suppliers is limited to 5 for all instances. If an instance has no solution with only 5 winning suppliers, the interval thresholds are doubled for each supplier until a solution exists.

## B. Results

The results of our experiments are summarized in tables I to IV. The instances are coded with 'S' for structured and 'R' for random instances. The first number indicates the number of suppliers, the second number reflects the number of goods and the third number is the maximal number of intervals per supplier. For each of these types of instances, 10 instances were generated and solved with the three algorithms. This resulted in computation times (in seconds) and a number of nodes searched in the branching tree for each algorithm, averaged per type of instance in the table. All computations were done on a Pentium IV 2 GHz

computer, with 512 Mb RAM.

In Table I, the results for the basic TQD problem are presented. Each algorithm solves all instances in a reasonable amount of time; random instances seem to be harder to solve than the structured ones. The min-cost flow based algorithm clearly performs best in terms of computation time for all instances with 10 or 20 suppliers. However, instances with 50 suppliers prove to be harder to solve with this algorithm. Although the solution time per node is undoubtedly the smallest with the min-cost flow approach, it needs more computing time than the other two exact algorithms. The branch-and-cut approach clearly searches the least amount of nodes, but to achieve this it needs a time-consuming cut generation process. The results show that it pays to generate cuts when the number of suppliers is large.

The results of our experiments with the variant with market share constraints are summarized in Table II. As in the basic case, the random instances require more computation time than the structured ones. Market share constraints are problematic for the branch-and-cut algorithm, whose computation times sometimes even double compared to the basic case. The linear programming based branch-and-bound algorithm deals with these constraints much better, since it manages to solve the instances faster than in the basic case. The min-cost flow algorithm is however by far the fastest algorithm for all instances. Especially for the instances with 50 suppliers, adding market share constraints causes the computations times to slump compared to the basic case. Moreover, less nodes need to be searched, which can be explained by the construction of the branching tree as described in section V.

Table III figures the results for the more-for-less variant. It turns out that in none of the structured instances purchasing extra goods leads to a lower total cost. In the random instances however, it is profitable in more than 85% of the instances to buy more than strictly needed. This is explained by the fact that discounts are substantially larger for the random cases than for the structured instances (see section VI-A).

Once again, the min-cost flow based algorithm performs best on all instances with 10 or 20 suppliers. For instances with 50 suppliers, it is advisable to use the linear programming based branch-and-bound algorithm. Compared to the basic case, the min-cost flow algorithm needs to search slightly more nodes, resulting in more computation time. Apart from the

random instances with 50 suppliers, which seem very difficult for all algorithms, this increase in computation time remains very modest. The linear programming based branch-and-bound algorithm is not affected too much either. The branch-and-cut algorithm however deals poorly with this variant.

Finally, Table IV describes the results for the variant that limits the number of winning suppliers. This constraint proved to be binding for more than 98% of the structured instances but less than 50% of the random instances. For the random instances, the prices drop sharper from one interval to the next, which makes it more interesting to go for the higher intervals. This leads to an optimal solution with less suppliers than for the structured instances. This explains why a constraint limiting the number of winning suppliers less often affects the random instances.

As for the computation times, branch-and-cut seems the best option for the structured instances. For the random instances, the picture is less clear. The instances with 10 suppliers are best solved with the min-cost flow algorithm, although this algorithm is left far behind by the other two for the instances with 20 suppliers. For these instances, branch-and-bound based on linear programming outperforms the other algorithms for instances where suppliers can have up to 5 volume intervals. Branch-and-cut is the fastest approach to solve random instances with 20 suppliers and up to 3 volume intervals per supplier. Notice that no instances with 50 suppliers are mentioned in this table, because the computation times for these problems were impractically high for all algorithms.

## VII. CONCLUSIONS

We presented a procurement problem where suppliers adopt a discount that depends on the total quantity ordered. We argued that different versions of this problem are NP-hard and that it is impossible to find a polynomial-time approximation algorithm with a constant ratio (unless  $P = NP$ ). We described three exact algorithms: one algorithm is based on our result that the problem can be solved by solving a number of min-cost flow problems; the other two algorithms are a branch-and-cut and an linear programming based branch-and-bound algorithm.

The algorithms were tested on fairly large randomly generated instances of the basic problem and three

variants. Our computational results show that all three algorithms came to an exact solution in a reasonable amount of time. However, it also became clear that each algorithm has instances for which it performs best. In general, the min-cost flow based algorithm works best for instances where the number of suppliers does not exceed 20 (which seems to correspond to most practical cases). It works especially well for the variant where we imposed constraints on the market share a supplier is allowed to obtain. The branch-and-cut algorithm outperforms the other algorithms on large instances in terms of suppliers of the basic case and on the structured instances of the variant that requires a limited amount of winning suppliers. Finally, the linear programming based branch-and-bound algorithm is at its best with the large instances of the variant where the buyer is allowed to purchase more than strictly needed.

## ACKNOWLEDGEMENTS

We wish to thank prof. W. Gochet and prof. W. Herroelen for their remarks on an earlier version of this work.

## APPENDIX

*Theorem 1:* The decision version of the TQD problem is strongly NP-complete.

**PROOF.** We define TQD' as the decision version of the TQD problem, where the question is whether it is possible to buy the required goods at a given total purchasing cost  $K$ . Obviously, TQD' is in NP, since given a solution it suffices to check the constraints and the value of the solution, which can easily be done in polynomial time. The reduction is from the 3-dimensional matching (3DM) problem.

The decision version of the 3DM problem is described as follows: given a set  $M \subseteq X \times Y \times Z$  of triples, where each of the sets  $X$ ,  $Y$  and  $Z$  has exactly  $q$  elements, is there a matching in  $M$  that contains  $q$  triples? Every instance of 3DM can be reduced to a TQD' instance in polynomial time. Suppose that the  $3q$  elements of the sets  $X$ ,  $Y$ , and  $Z$  correspond to  $3q$  goods and that each 3-element subset in  $M$  corresponds to a supplier, so  $n = q$  and  $m = 3q$ . Each supplier has 2 intervals. The price of each good in its first interval is 1. This interval has a lower bound of 0 and an upper bound of 2. The second interval has a lower bound of 3 and an upper bound of  $\infty$ . The price of each good in this second interval is also 1, except for the three goods in the 3-element subset corresponding to the supplier, each of which have a price of 0. Each good needs to be purchased exactly once, i.e.,  $d_k = 1 \forall k$ . The question is whether the TQD' problem can be solved with a total purchasing cost of 0.

Further, every yes-instance of 3DM corresponds to a yes-instance of TQD'. A solution of 3DM consists of  $q$  3-element subsets, corresponding to  $q$  suppliers in the TQD' problem. Purchasing from each of these suppliers exactly the 3 goods represented by the 3-element subset enables us to reach every supplier's second interval, where these 3 goods can be bought at price 0. Since every element of  $X \cup Y \cup Z$  occurs exactly once in the solution of 3DM, every good will also be purchased exactly once in the TQD' solution. Therefore, if 3DM has a solution, it can easily be transformed to a solution of TQD'.

Vice versa, every yes-instance of TQD' also corresponds to a yes-instance of 3DM. A solution of the TQD' problem consists of a number of selected suppliers, together providing every good exactly once at a total cost of 0. If a supplier would provide less than 3

goods, the quorum to get in the second interval would not be met, so the cost would not be 0. If the supplier would provide more, the cost would also be strictly positive, because all but these 3 goods still have a price of 1 in the second interval. Providing more than one of the 0-priced goods would violate the demand constraint stating that each good is to be supplied exactly once. Therefore every selected supplier provides precisely 3 goods, namely those that have a price of 0 in the second interval and since  $3q$  goods need to be provided,  $q$  suppliers must be selected. Therefore, for each of the  $q$  suppliers selected in the solution of the TQD' problem, there is a corresponding 3-element set in  $M$ . Moreover, these  $q$  triples define a matching, since every good is bought exactly once. As a consequence, the decision version of the TQD problem is strongly NP-complete.  $\square$

*Theorem 2:* No polynomial-time approximation algorithm with constant worst-case ratio exists for the TQD problem (unless  $P = NP$ ).

**PROOF.** Assume that a  $\rho$ -approximation algorithm for the TQD problem exists. Consider now an instance of 3DM with  $M \subseteq X \times Y \times Z$ , and let us build an instance of the TQD problem as in the proof of Theorem 1 with a price of  $\rho+1$  for any good bought in the first interval, or bought in the second interval when not belonging to one of the three goods of that supplier. Observe that this instance of the TQD problem either has an optimal solution with cost 0 (namely when the 3DM-instance has a matching), or it has an optimal solution with cost at least  $\rho+1$  (when there is no matching in the 3DM instance). Thus, if there is a 3DM-matching the  $\rho$ -approximation algorithm must return a zero-cost solution, which contradicts the NP-hardness of 3DM. Hence such an algorithm cannot exist unless  $P = NP$ .  $\square$

*Theorem 3:* The decision version of the TQD problem with a common discount rate  $\delta$  is strongly NP-complete.

**PROOF.** In order to show that the TQD problem with a common discount rate is NP-complete, we modify the reduction used in Theorem 1 as follows. As in Theorem 1, each supplier has 2 intervals, the first interval ranges from 0 to 2 goods, the second from 3 to an unlimited amount of goods. The prices of all goods in both the first interval are 2, except for the three goods in the 3-element subset corresponding to the supplier, each of which have a price of 1. Since all suppliers use a common discount rate  $\delta$ , the prices in the second

interval are  $(1 - \delta)$  for the three goods in the 3-element subset, and  $2(1 - \delta)$  for the other goods. Each good still needs to be purchased exactly once. The question is now whether this TQD problem can be solved with a total purchasing cost of  $m(1 - \delta)$ . The same reasoning as in Theorem 1 can be applied to verify that every yes-instance of 3DM corresponds to a yes-instance of the TQD problem with common discount rate and vice versa and that indeed the decision version of the TQD problem with a common discount rate is strongly NP-complete.  $\square$

**Theorem 4:** The decision version of the more-for-less variant of TQD problem is strongly NP-complete.

**PROOF.** In the more-for-less setting, the buyer is allowed to purchase more than  $m$  goods in order to reduce the total cost. We can however use the same reduction as in Theorem 3. Indeed, let each supplier have 2 intervals, the first ranging from 0 to 2 goods, the second from 3 to an unlimited amount of goods. Once again, the prices of all goods in both the first and second interval are 1, except for the three goods in the 3-element subset corresponding to the supplier, each of which have a price of  $(1 - \delta)$ . The question remains whether it is possible to solve this TQD problem with a total purchasing cost of  $m(1 - \delta)$ . Clearly this can not be achieved by purchasing more than  $m$  goods, which allows us to conclude that every yes-instance of 3DM corresponds to a yes-instance of the more-for-less variant and vice versa. Hence, the decision version of the more-for-less variant of the TQD problem is strongly NP-complete.  $\square$

## REFERENCES

- [1] L.M. Austin and W.W. Hogan (1976). *Optimizing the procurement of aviation fuels*. Management Science, 22(5), pp. 515-527.
- [2] M. Bichler, J.R. Kalagnanam, H.S. Lee and J. Lee. (2002) *Winner Determination Algorithms for Electronic Auctions: A Framework Design*. In: Proceedings of EC-Web 2002, pp. 37-46.
- [3] S.S. Chauhan, A.V. Eremeev, A.A. Romanova, V.V. Servakh and G.J. Woeginger. (2005) *Approximation of the supply scheduling problem*. Operations Research Letters, 33(3), pp. 249-254.
- [4] Y. Crama, R. Pascual J. and A. Torres (2004). *Optimal procurement decisions in the presence of total quantity discounts and alternative product recipes*. European Journal of Operational Research, 159(2) pp. 364-378.
- [5] A.J. Davenport and J.R. Kalagnanam (2002). *Price negotiations for procurement of direct inputs*. In: B. Dietrich and R.V. Vohra, Mathematics of the internet: e-auction and markets, pp. 27-43.
- [6] Z. Degraeve, E. Labro and F. Roodhooft (2000). *An evaluation of vendor selection models from a total cost of ownership perspective*. European Journal of Operational Research, 125 pp. 34-58.
- [7] M. Eso, S. Ghosh, J.R. Kalagnanam and L. Ladanyi (2001). *Bid evaluation in procurement auctions with piece-wise linear supply curves*. IBM Research Report RC 22219, 2001.
- [8] G. Hohner, J. Rich, E. Ng, G. Reid, A.J. Davenport, J.R. Kalagnanam, H.S. Lee, and C. An (2003). *Combinatorial and Quantity-Discount Procurement Auctions Benefit Mars, Incorporated and Its Suppliers*. Interfaces, 33(1) pp. 23-35.
- [9] P. Katz, A.A. Sadrian and P. Tendick (1994). *Telephone Companies Analyze Price Quotations with Bellcore's PDSS Software*. Interfaces, 24(1) pp. 50-63.
- [10] J.J. van de Klundert, J. Kuipers, F.C.R. Spieksma and M. Winkels (2003). *Telecommunication Carrier Selection under Volume Discounts: a Case Study*. Research Report 0330, Department of Applied Economics, K.U.Leuven, 2003, accepted for Interfaces.
- [11] A. Kothari, D. Parkes and S. Suri (2003). *Approximately strategyproof and tractable multi-unit auctions*. In: Proceedings of the ACM Conference on Electronic Commerce 2003, pp. 166-175.
- [12] C.L. Munson and M.J. Rosenblatt (1998). *Theories and realities of quantity discounts: an explanatory study*. Production and Operations Management, 7(4) pp. 352-369.
- [13] G.L. Nemhauser and L.A. Wolsey (1988). *Integer and Combinatorial Optimization*. Wiley New York (N.Y.).
- [14] A.A. Sadrian and Y.S. Yoon (1994). *A Procurement Decision Support System in Business Volume Discount Environments*. Operations Research, 42(1) pp. 14-23.
- [15] H. Shachnai, O. Shmueli and R. Sayegh (2004). *Approximation Schemes for Deal Splitting and Covering Integer Programs with Multiplicity Constraints*. Manuscript.
- [16] M. Sun (2002). *The transportation problem with exclusionary side constraints and two branch-and-bound algorithms*. European Journal of Operational Research, 140 pp. 629-647.
- [17] J. Xu, L.L. Lu and F. Glover (2000). *The deterministic multi-item dynamic lot size problem with joint business volume discount*. Annals of Operations Research, 96(1) pp. 317-337

<b>Instances</b>	<b>mcf branch&amp;bound</b>		<b>branch&amp;cut</b>		<b>lp branch&amp;bound</b>	
	comp. time	#nodes	comp. time	#nodes	comp. time	#nodes
S-10-40-3	0,01	116,6	0,09	0,3	0,08	29,3
S-10-40-5	0,02	161,1	0,15	10,9	0,11	52,5
S-10-100-3	0,02	69,8	0,12	0,2	0,11	17,2
S-10-100-5	0,14	501,6	0,55	3,2	0,36	74,3
S-20-40-3	0,07	389,2	0,12	0,5	0,16	73,5
S-20-40-5	0,38	1.887,8	0,50	4,7	0,58	207,3
S-20-100-3	0,30	749,6	0,34	1,3	0,57	128,8
S-20-100-5	0,67	1.512,8	1,17	2,1	1,07	155,1
S-50-40-3	5,61	16.671,8	0,51	2,7	1,92	719,0
S-50-40-5	32,93	85.210,4	2,99	16,5	7,81	2.087,2
S-50-100-3	21,81	26.595,3	1,45	2,1	4,67	696,1
S-50-100-5	159,77	168.181,3	10,45	14,7	24,41	2.614,0
R-10-40-3	0,01	54,9	0,09	2,1	0,07	24,3
R-10-40-5	0,07	428,9	0,59	30,5	0,31	160,7
R-10-100-3	0,02	46,9	0,14	2,6	0,10	10,1
R-10-100-5	0,31	845,1	1,50	31,7	0,78	160,0
R-20-40-3	0,14	700,5	0,29	9,5	0,25	121,6
R-20-40-5	0,45	2.155,6	1,81	68,5	1,56	659,9
R-20-100-3	0,59	1.249,6	0,83	8,1	1,05	235,0
R-20-100-5	3,18	3.938,2	6,81	70,1	5,34	882,8
R-50-40-3	10,31	28.975,3	1,67	81,6	6,17	2.411,0
R-50-40-5	18,60	48.876,6	14,18	140,9	18,41	4.303,1
R-50-100-3	97,29	103.885,7	8,84	43,8	38,47	6.289,1
R-50-100-5	241,39	237.953,3	61,71	216,8	122,49	11.451,2

TABLE I  
COMPUTATIONAL RESULTS FOR THE BASIC CASE

<b>Instances</b>	<b>mcf branch&amp;bound</b>		<b>branch&amp;cut</b>		<b>lp branch&amp;bound</b>	
	comp. time	#nodes	comp. time	#nodes	comp. time	#nodes
S-10-40-3	0,01	55,6	0,12	0,7	0,08	25,3
S-10-40-5	0,01	51,3	0,21	2,7	0,12	34,4
S-10-100-3	0,03	102,0	0,18	0,9	0,14	17,6
S-10-100-5	0,07	223,7	0,82	2,9	0,39	58,5
S-20-40-3	0,05	233,6	0,29	0,8	0,20	56,4
S-20-40-5	0,10	453,8	1,68	17,4	0,91	255,8
S-20-100-3	0,13	267,1	0,86	2,5	0,78	136,6
S-20-100-5	0,35	672,4	3,47	9,3	2,08	268,8
S-50-40-3	0,24	622,5	1,18	5,7	1,07	196,1
S-50-40-5	0,31	858,5	9,36	71,9	5,47	790,7
S-50-100-3	1,21	1.185,0	2,86	5,8	3,58	290,4
S-50-100-5	2,85	3.002,7	20,08	63,7	15,00	807,3
R-10-40-3	0,01	67,9	0,15	7,5	0,08	24,0
R-10-40-5	0,04	248,5	0,80	20,4	0,30	125,1
R-10-100-3	0,02	40,7	0,20	0,2	0,14	15,1
R-10-100-5	0,20	546,7	2,46	27,0	1,04	213,2
R-20-40-3	0,12	484,5	0,60	21,6	0,32	132,7
R-20-40-5	0,24	1.062,4	2,91	62,2	1,50	505,7
R-20-100-3	0,26	453,0	1,81	26,5	1,27	253,6
R-20-100-5	5,50	9.671,1	11,95	105,2	8,44	1.226,5
R-50-40-3	0,19	526,5	2,38	25,8	1,15	214,0
R-50-40-5	0,56	1.552,2	19,18	273,7	7,32	2.099,7
R-50-100-3	2,12	2.046,3	7,66	34,1	3,69	287,5
R-50-100-5	15,55	15.900,1	59,75	228,9	27,79	1.731,3

TABLE II  
COMPUTATIONAL RESULTS FOR VARIANT 1 (MARKET SHARE CONSTRAINTS)

<b>Instances</b>	<b>mcf branch&amp;bound</b>		<b>branch&amp;cut</b>		<b>lp branch&amp;bound</b>	
	comp. time	#nodes	comp. time	#nodes	comp. time	#nodes
S-10-40-3	0,02	118,7	0,14	0,4	0,06	27,9
S-10-40-5	0,02	166,8	0,49	23,2	0,11	50,6
S-10-100-3	0,02	69,8	0,29	6,8	0,12	17,5
S-10-100-5	0,16	513,3	2,48	66,5	0,40	75,3
S-20-40-3	0,07	389,9	0,45	5,2	0,17	71,3
S-20-40-5	0,43	1.997,3	4,42	148,7	0,61	219,3
S-20-100-3	0,32	749,9	2,08	43,4	0,63	135,7
S-20-100-5	0,74	1.545,9	11,47	164,2	1,26	185,8
S-50-40-3	5,57	16.724,3	8,90	261,9	1,98	739,7
S-50-40-5	36,90	94.865,4	147,56	2.500,0	9,88	2.491,3
S-50-100-3	22,39	26.625,3	28,42	523,7	5,65	832,3
S-50-100-5	171,14	172.466,7	271,80	3.006,0	32,31	3.365,0
R-10-40-3	0,02	55,5	0,10	0,1	0,06	21,6
R-10-40-5	0,05	428,5	0,67	18,5	0,17	82,9
R-10-100-3	0,01	45,6	0,17	0,2	0,10	9,3
R-10-100-5	0,24	1.068,9	2,49	42,6	0,50	91,5
R-20-40-3	0,13	793,8	0,60	15,9	0,24	117,0
R-20-40-5	0,47	3.167,6	3,55	89,8	0,91	343,9
R-20-100-3	0,53	1.369,0	1,85	17,6	1,00	241,4
R-20-100-5	3,11	10.389,8	25,09	434,8	4,46	801,6
R-50-40-3	15,91	59.066,0	35,16	1.195,8	10,51	4.615,8
R-50-40-5	39,33	171.566,6	169,82	1.511,1	25,54	6.714,3
R-50-100-3	130,42	206.668,9	274,82	6.035,4	79,93	14.059,4
R-50-100-5	446,85	798.002,9	2.036,07	17.577,4	398,31	45.945,3

TABLE III  
COMPUTATIONAL RESULTS FOR VARIANT 2 (MORE FOR LESS)

<b>Instances</b>	<b>mcf branch&amp;bound</b>		<b>branch&amp;cut</b>		<b>lp branch&amp;bound</b>	
	comp.	time	#nodes	comp.	time	#nodes
S-10-40-3	0,26	1.786,3	0,17	3,4	0,29	372,0
S-10-40-5	0,32	2.135,7	0,27	8,1	0,41	421,1
S-10-100-3	0,73	2.017,0	0,35	8,0	1,23	603,4
S-10-100-5	1,43	4.115,9	1,17	16,5	2,65	1.007,4
S-20-40-3	6,70	30.788,1	0,34	2,4	3,00	2.763,0
S-20-40-5	15,48	66.377,0	1,07	27,3	2,39	1.345,2
S-20-100-3	23,83	44.780,9	1,63	17,3	16,76	5.083,6
S-20-100-5	20,02	36.588,2	2,73	12,8	7,15	1.597,2
R-10-40-3	0,03	229,6	0,09	2,1	0,06	25,6
R-10-40-5	0,43	2.697,7	0,57	35,9	0,26	181,9
R-10-100-3	0,10	302,7	0,11	0,0	0,13	26,6
R-10-100-5	0,71	2.040,6	1,33	28,5	0,74	195,9
R-20-40-3	2,06	9.812,6	0,30	8,5	0,37	252,7
R-20-40-5	5,73	25.467,2	1,63	61,2	1,08	578,6
R-20-100-3	7,07	13.881,7	0,91	9,5	2,38	730,9
R-20-100-5	26,47	48.444,9	6,49	66,6	4,19	920,6

TABLE IV  
COMPUTATIONAL RESULTS FOR VARIANT 3 (LIMITED NR. OF WINNING SUPPLIERS)

# Designing Reliable Systems with SREMS++

Angel Juan<sup>\*</sup>, Javier Faulín<sup>†</sup>, Vicente Bargueño<sup>‡</sup> and Anita Goyal<sup>§</sup>

<sup>\*</sup>Polytechnic University of Catalonia (Spain)  
*Department of Applied Mathematics I*  
Email: [angel.alejandro.juan@upc.edu](mailto:angel.alejandro.juan@upc.edu)

<sup>†</sup>Universidad Nacional de Educación a Distancia (Spain)  
*Department of Applied Mathematics I*  
Email: [vbargueno@ind.uned.es](mailto:vbargueno@ind.uned.es)

<sup>†</sup>Public University of Navarra (Spain)  
*Department of Statistics and Operations Research*  
Email: [javier.faulin@unavarra.es](mailto:javier.faulin@unavarra.es)

<sup>§</sup>Management Development Institute (India)  
*Marketing Department*  
Email: [agoval@mdi.ac.in](mailto:agoval@mdi.ac.in)

**Abstract**— Complex systems are everywhere among us: telecommunication networks, computer systems, transporting vehicles, and electrical appliances are well known examples. Designing these as reliable systems is a very important task for managers and engineers, since reliability have a strong relationship to other critical concepts such as quality and security. Furthermore, this task is extremely difficult, due to the fact that analytical methods can become too complicated, inefficient or even inappropriate when dealing with sophisticated systems. In this paper we present the basic ideas behind a simulation-based method, called SREMS, which can be very useful during the design and improvement phases for a wide range of complex systems. SREMS not only provides good estimations of system reliability at different target times (survival function) and system failure-time parameters, but it also allows to identify those components that play a critical role in the system durability, something that can be very useful when improving a system reliability.

**Keywords**— system reliability, simulation methods, C++ programming

## I. RELIABILITY DEFINITION

Reliability is often defined as the probability that a system or device will perform its intended function, under operating conditions, for a specified period of time [23]. On the other hand, availability can be defined as the probability that a system or device, according to some maintenance policy and some operating conditions, performs its intended function at a certain time.

## II. MOTIVATION AND APPLICATIONS

There are many complex systems that we use in our day-to-day life without giving any consideration to their routine maintenance. Personal computers, computer networks, television sets, or DVDs players are some examples of such complex systems. Malfunctioning of these systems can cause a lot of inconveniences, not only in terms of time

and money but also in terms of other hidden costs such as energy and psychic costs [15].

The existence of these consumer costs provides good business opportunities: producers able to deliver more reliable goods than competitors can offer longer warranty periods for their products which, in turn, will add value to them. Positive effects of reliable products on brand or company image should also be considered, since they play an essential role in a competitive market.

Reliability can also be a desirable property of costly systems, such as telecommunication networks or transportation vehicles. Reliability can even be a necessity for systems related to human safety and security, such as buildings, bridges, power plants, airplanes, ships, military weapons, etc.

In all those scenarios, managers and engineers can benefit from efficient methods and software tools that help them to design more reliable systems.

## III. SIMULATION IN RELIABILITY STUDIES

Reliability and availability of time-dependent complex systems is a research area with applications not only to engineering but also to experimental and social sciences [5], [19], [25].

Different analytical approaches can be used in order to calculate the exact reliability of a time-dependent complex system [16]. Unfortunately, when the system is highly complex, it can become extremely difficult or even impossible to obtain its exact reliability at a given target time. Similar problems arose when trying to determine the exact availability at a given target time for systems subject to maintenance policies. Many authors point out that in those situations only simulation techniques, such as Monte Carlo Simulation (MS) and Discrete Event Simulation (DES), can be useful to obtain estimates for reliability and availability parameters (see [2], [4], [6], [10], and [11]).

One of the first good ideas and algorithms on the use of MS techniques to estimate system reliability can be found in [14]. MS techniques have also been proposed to study complex systems availability [26]. In fact, during the last years, several commercial simulators have been developed to study the reliability and availability of complex systems [27]. More recently, several authors have proposed spreadsheets models that make use of MS to find out the reliability of complex systems [9], [13]. As far as we know, SREMS++, the computer version of the method we present here, has some characteristics that cannot be found in those other programs: (i) it helps to identify critical paths and components – those which have a fundamental impact over the system reliability, (ii) it provides statistics about system failure time, (iii) it allows to model systems of virtually any size, and (iv) it is freeware.

#### IV. WHICH SYSTEMS ARE WE CONSIDERING?

The method presented in this paper, SREMS, has been designed to deal with any kind of logical or physical system that meets some general criteria. The hypotheses made by SREMS are common assumptions in reliability studies and they are less restrictive than the ones used when applying analytical methods.

Specifically, when studying the reliability of systems, which are not subject to any maintenance policy (i.e.: when components repair or substitution will not be considered while the system stills working properly), SREMS will make the following main assumptions:

- A1) Two-state systems:** at any given time, the system will be either operational (working properly) or not. Observe that the exact definition of “being operational” is up to the system managers, since it will vary depending of the system and its environmental circumstances
- A2) Minimal paths decomposition:** the system logical structure is known and it can be expressed in the form of minimal paths [16]
- A3) Component failure-times distribution:** for each component, its associated failure-times distribution is perfectly known (i.e.: both the statistical distribution family and exact parameters are known)

- A4) Failure-times independence:** the failure-time associated to one component is independent from the failure-time associated to any other component

Assumptions A2 and A3 guarantee that there is enough information to study the system reliability. Assumption A2 often requires a detailed analysis of logical relationships among components. When dealing with systems with a lot of components and a high degree of relationships among them, determining the system minimal paths structure is not an easy task at all. In those situations, some proposed algorithms –also based on simulation techniques, can be used to find out the minimal paths decomposition [20]. In the context of assumption A3, statistical methods such as accelerated live tests [22] and data fitting techniques [18] are usually required.

Finally, assumption A4 is the most restrictive one and it may require considering some abstraction levels in the system decomposition. For example, if the system were a PC it could be necessary to join several pieces, such as the microprocessor and its associated fan, in just one component. Otherwise, it could not be possible to assume independence among components failure-times.

#### V. STATISTICAL FUNDAMENTALS

SREMS makes use of several mathematical concepts and techniques. Specifically, the method is based on:

- F1) System reliability theory:** system reliability concepts and minimal paths theory [3], [12], [16], [23]
- F2) Simulation techniques:** data fitting, pseudo-random number generation, and variance reduction methods (if necessary) [1], [7], [17], [21], [26]
- F3) Probability and statistical concepts:** probability theory, descriptive statistics and inference techniques [8], [17]

The driving idea of the method is explained below:

Given a fixed instant  $t_0 \geq 0$ , the main target of the simulator is to estimate the system reliability at that time<sup>1</sup>, i.e.:

$$p(t_0) = P(T_s > t_0) = R_s(t_0)$$

<sup>1</sup> In fact, the method will consider several target times at once, since we will be interested in understanding the system survival function.

Considering that the system has two possible states at any given time –it will be either operational or not, it is possible to interpret  $p(t_0)$  as the probability of “success” in a Bernoulli distribution, understanding success as the fact that the system is operational at  $t_0$ , i.e.:

$$Y = \phi(\mathbf{X}(t_0)) \sim Be(p(t_0))$$

where  $\mathbf{X}(t_0)$  is the status vector of the system at  $t_0$  (it describes the actual status of each component at  $t_0$ ), and  $\phi(\mathbf{X}(t_0))$  is the binary status function (value will be 1 if the system is operational at  $t_0$ , and 0 otherwise).

In that situation, if we are able to obtain –using some simulation algorithm<sup>2</sup>,  $m$  random and independent observations,  $Y_1, Y_2, \dots, Y_m$ , from the binary variable  $Y$ , which represents the system status at  $t_0$ , we know that the sum of those variables will be a new one following a binomial distribution of parameters  $m$  (number of proofs) and  $p(t_0)$  (probability of “success” in each proof), i.e.:

$$\sum_{i=1}^m Y_i \sim Bi(m, p(t_0))$$

At this point, it is known that the maximum likelihood estimator for  $p(t_0) = \phi(\mathbf{X}(t_0))$  is given by the sample mean [8]:

$$p' = \bar{y}_i = \frac{\sum_{i=1}^m y_i}{m}$$

This is an unbiased estimator, i.e.:  $E[p'] = p(t_0)$ . Furthermore, the law of the large numbers says that the former estimator will tend to get better as the number of observations gets larger, more concisely:

$$P\left(\lim_{m \rightarrow \infty} p' = p(t_0)\right) = 1$$

Observe that, apart from obtaining a point estimate, it is also possible to obtain confidence intervals for  $p(t_0) = R_s(t_0)$ . In effect, for large values of  $m$ , the central limit theorem says that a confidence interval for  $p(t_0) = R_s(t_0)$  with a  $1 - \alpha$  confidence level would be [8]:

$$p' \pm z\left(\frac{\alpha}{2}\right) \cdot \sqrt{\frac{p(t_0) \cdot (1 - p(t_0))}{m}}$$

where  $z\left(\frac{\alpha}{2}\right)$  is the  $1 - \frac{\alpha}{2}$  percentile in a standard normal distribution (in practice, since  $p(t_0) = R_s(t_0)$  is not known, its value will be substituted by the point estimate  $p'$ ).

The methodology just described here is based on core statistical concepts. Therefore, it is obvious that the key point of SREMS consists in the developing of a simulation-based algorithm, which provides us the  $m$  random and independent observations from the variable  $Y$ . That algorithm, which we have called SREMS\_A1, will be explained in the following section. Observe that the SREMS\_A1 algorithm will not only provide us with the  $m$  needed observations, but also with information about: (a) system critical paths and components, and (b) system failure time.

## VI. ALGORITHM SREMS\_A1

The implementation of SREMS\_A1 as a computer program has a lot of technical details, both of mathematical and programming nature. Therefore, only the general ideas behind the algorithm will be presented here:

Given a system with  $n$  components, SREMS\_A1 performs the following actions:

- S1)** Using information about the distribution of each component failure-times and some high-quality random numbers generator, it assigns a random failure-time,  $T_i$ , to each of the  $n$  components
- S2)** Using results from S1 and information about system structure (minimal paths), it is possible to determine each path failure-time and the failure-time associated to the key-component in each path (i.e.: the most durable component, which will determine the path failure time)
- S3)** Using results from S2, it is possible to determine which are the strongest and weakest paths (that is, the last and the first paths which are likely to go out-of-service)
- S4)** Using results from S2 and S3, it determines the system failure-time,  $T_s$
- S5)** It repeats steps S1 to S4 as many times as it has been indicated by the number of iterations to run ( $m$  times)

<sup>2</sup> This algorithm will be SAEDES\_A1, which will be explained later.

As a result of S5, a random sample will be obtained for each of the following variables: “strongest path”, “weakest path”, “key component in strongest path”, “key component in weakest path”, and “system failure-time”

- S6) Using information from S5, it obtains point and interval estimates for the system failure-time
- S7) For each observed value of the variable “system failure-time” (obtained in S5), it compares this value with each of the target times,  $t_0$ , and if  $T_s > t_0$ , a “success” is assigned to that target time (that is,  $Y(t_0) = 1$ ). Otherwise,  $Y(t_0) = 0$
- S8) Using information from S7, it calculates point and interval estimates for the system reliability at each target time (this parameter is estimated as the number of “successes” associated to that target time divided by the total number of iterations). This will provide us with information about the system survival function. Observe that at both extremes of this function the probability of system failure will be almost 0 or almost 1, respectively. Those extreme situations cause the so called “rare events” problem, which implies that a lot of iterations must be run in order to obtain good estimates for the system reliability at both target times (some variance-reduction techniques could be used to reduce the number of iterations, although we consider that it will not be necessary since we usually will be more interested in the middle target times of the survival function than in the extreme ones)
- S9) Finally, using information from S5, it calculates relative frequencies for each of the following variables: “strongest path”, “weakest path”, “key component in strongest path”, and “key component in weakest path”

**Figure 1** shows a flow diagram that summarizes the SREMS\_A1 algorithm.

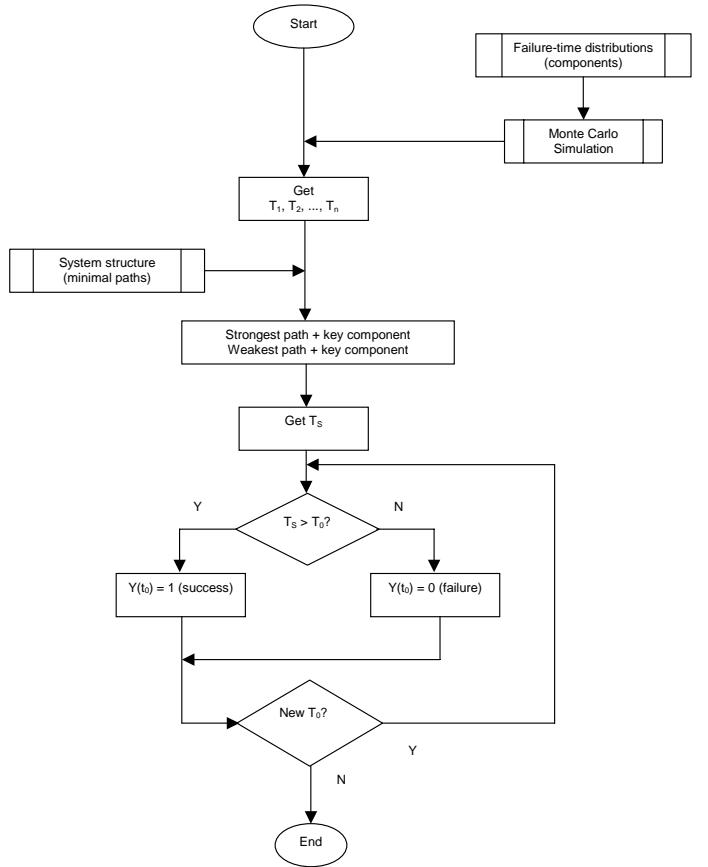


Fig. 1. Flow diagram for SREMS\_A1

## VII. SOFTWARE IMPLEMENTATION

Two important objectives are: (a) to validate the accuracy and effectiveness of the proposed method, and (b) to provide a computer version of the mathematical model that could be used by system managers and engineers. In order to attain those targets, we have developed SREMS++, a C/C++ software implementation of the method.

As **Figure 2** shows, the program, SREMS++, has a modular structure including: (i) a kernel, which takes care of the simulation model, (ii) the library `randomVariates`, which is a random variables generator specially developed for SREMS++ (it includes a well tested and long period pseudo-random numbers generator), and (iii) the `stats` library, which performs all the required statistical operations (descriptive, confidence intervals, etc.).

The program inputs are entered by the user employing three simple `txt` files, and after running the simulation, the program provides a single `txt` file with the results.

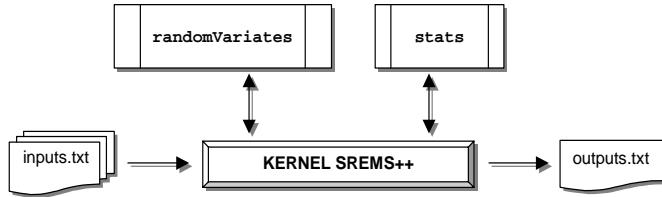


Fig. 2. Modular structure of SREMS++

The three inputs files must contain the following information:

#### **srems\_inputs\_first.txt**

- I1) Number of iterations to run (more iterations implies not only more accurate estimates, but also an increase in computational costs such as simulation length and memory resources)
- I2) Number of components in the system
- I3) Number of target times (times when system reliability has to be determined)
- I4) Time interval (step) between consecutive target times
- I5) Number of paths in the logical structure
- I6) Seed of the simulation

#### **srems\_inputs\_second.txt**

- I7) Failure-times distribution associated to each component
- I8) Length of each path

#### **srems\_inputs\_third.txt**

- I9) System path composition (array of components that make up each path)

On the other hand, once the simulation has been run, the outputs file, **srems\_outputs.txt**, provides the following information about the system being modeled:

- O1) Point and interval estimates for the system reliability at different target times
- O2) Descriptive statistics for the variable “system failure-time”
- O3) Information about key components and paths (i.e.: components and paths that play a critical role in system failure-time and, therefore, in system reliability)

Several tests of the program have been carried out, covering systems with different levels of complexity. Whenever it has been possible, exact reliability values (obtained via analytical methods), have been compared with estimated ones. All tests have given satisfactory results. The following section includes three of the experiments that have been performed to validate the method and to illustrate the program possibilities.

## VIII. CASE STUDY 1: A SIMPLE SYSTEM

A simple system is shown in **Figure 3**. The system has seven components and three minimal paths.

**Table I** contains the failure-time distribution for each of the components.

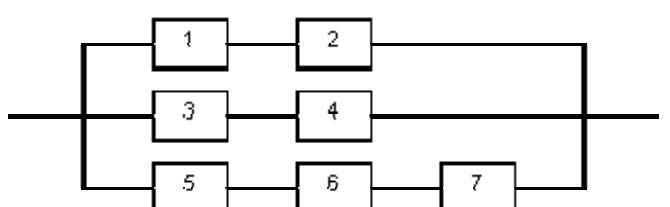


Fig. 3. System for case study 1

TABLE I  
Failure times for components in system 1

Component	Distribution	Parameter 1	Parameter 2
1	Weibull	1.32	2.13
2	Weibull	1.34	2.12
3	Weibull	1.33	2.11
4	Exponential	1.2	--
5	Weibull	1.22	2.01
6	Exponential	1.22	--
7	Weibull	1.20	2.01

In this case, it is not difficult to find out the exact value for  $R_s(t_0)$ , the system reliability at  $t_0$ . If  $P_i$  represents the  $i$ th path, we can use probability theory [24] to get the exact value for  $R_s(t_0)$ :

$$\begin{aligned} R_s(t_0) &= P(T_s > t_0) = P(T_{p_1} > t_0 \cup T_{p_2} > t_0 \cup T_{p_3} > t_0) = \\ &= P(T_{p_1} > t_0) + P(T_{p_2} > t_0) + P(T_{p_3} > t_0) - \\ &\quad - P(T_{p_1} > t_0 \cap T_{p_2} > t_0) - P(T_{p_1} > t_0 \cap T_{p_3} > t_0) - \\ &\quad - P(T_{p_2} > t_0 \cap T_{p_3} > t_0) + P(T_{p_1} > t_0 \cap T_{p_2} > t_0 \cap T_{p_3} > t_0) \end{aligned}$$

and, since each path failure-time is independent from the others, the former expression is equivalent to:

$$\begin{aligned} R_s(t_0) &= R_{p_1}(t_0) + R_{p_2}(t_0) + R_{p_3}(t_0) - \\ &\quad - R_{p_1}(t_0) \cdot R_{p_2}(t_0) - R_{p_1}(t_0) \cdot R_{p_3}(t_0) - \\ &\quad - R_{p_2}(t_0) \cdot R_{p_3}(t_0) + R_{p_1}(t_0) \cdot R_{p_2}(t_0) \cdot R_{p_3}(t_0) \end{aligned}$$

finally, since each minimal path is a series structure containing independent failure-time components, it is clear that:

$$\forall 1 \leq i \leq 3, \quad R_{p_i}(t_0) = \prod_{j=1}^{n_i} R_{ij}(t_0)$$

where  $n_i$  represents the number of components in the  $i$ th path (i.e.:  $n_1 = n_2 = 2$  y  $n_3 = 3$ ) and  $R_{ij}(t_0)$  represents the reliability, at  $t_0$ , of the  $j$ th component on the  $i$ th path.

Using the former expressions and the inputs on each component failure-time distribution, it is easy to determine the reliability, at  $t_0$ , associated to each component, to each path and, eventually, to the whole system. For instance, taking as target times  $t_0 = 0.5, 1.0, 1.5, 2.0$ , the exact reliability of the system at each  $t_0$  is:

$$\begin{array}{ll} R_s(0.5) = 0.9408 & R_s(1.0) = 0.7039 \\ R_s(1.5) = 0.4364 & R_s(2.0) = 0.2414 \end{array}$$

Once the exact reliability for this system has been obtained using probabilistic methods (it is to remember that it will not be possible to obtain this exact value in most of the cases), the next step will be to use SREMS++ to get an estimate value of the system reliability at the different target times (therefore, we will be able to compare the estimated value at each  $t_0$  with the exact one and, eventually, the exact survival function with the estimated one).

Figure 4 shows the results provided by SREMS++ for this system after 500,000 iterations:

	R(t0)	99% CI
1:	0.940474	(0.939612 0.941336)
2:	0.703990	(0.702327 0.705653)
3:	0.435944	(0.434138 0.437750)
4:	0.240954	(0.239396 0.242512)

	strong	weak
1:	0.522930	0.177026
2:	0.293440	0.346384
3:	0.183630	0.476590

	key sp	key wp
1:	0.259888	0.089580
2:	0.263042	0.087446
3:	0.114834	0.089448
4:	0.178606	0.256936
5:	0.051520	0.110686
6:	0.080476	0.251624
7:	0.051634	0.114280

Fig. 4. SREMS++ outputs for system 1

The following analysis can be derived from the output:

- The total time used by the program to complete the 500,000 iterations in a standard PC (AMD Sempron, 2.2Ghz, 512MB RAM) has been of just 3 seconds
- The estimated value for the “mean time to system failure” is 1.5318 (this value cannot be obtained using probabilistic methods)
- The estimated values for the system reliability at the considered target times are:

$$\begin{aligned} R_s(0.5) &= 0.9405 \\ R_s(1.5) &= 0.4360 \end{aligned}$$

$$\begin{aligned} R_s(1.0) &= 0.7040 \\ R_s(2.0) &= 0.2410 \end{aligned}$$

In other words, after a few seconds we have obtained some estimated values that are very close to the real ones (i.e.: the estimated survival function and the real survival function are almost quite similar).

- Path 1 has been the strongest path 52% of times, where path 3 has been the weakest path 48% of times. It makes sense, therefore, to say that a good way to increase system reliability will be to add redundant components to these two paths
- Components 1 and 2 have been the key components associated to the strongest path 26% of times. On the other hand, components 4 and 6 have been the key components 26% and 25% of times associated to the weakest path respectively. Therefore, it makes sense to say that improving reliability of components 1, 2 and 6 (either adding redundancies or increasing components quality) is a good way to improve global system reliability

## IX. CASE STUDY 2: AN INTERMEDIATE SYSTEM

A more complex system containing seven components can be seen in **Figure 5**, where the structure resembles similar to that of some telecommunication or computer network.

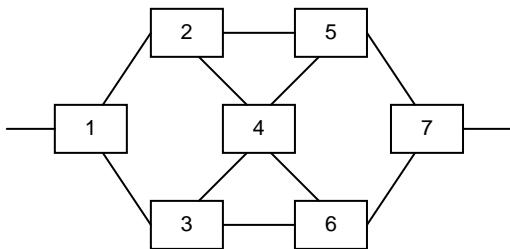


Fig. 5. System for case study 2

The associated failure-times distributions are in **Table II** and **Table III** show the logical (minimal path) structure of the same system.

**TABLE II**  
Failure times for components in system 2

Component	Distribution	Parameter 1	Parameter 2
1	Weibull	1.8	2.8
2	Weibull	1.7	2.7
3	Weibull	1.6	2.6
4	Weibull	1.6	2.5
5	Weibull	1.4	2.4
6	Weibull	1.2	2.2
7	Weibull	1.3	2.3

Observe that this system presents some additional complications -as the fact that several components are repeated in different minimal paths; those complications tend to increase the difficulty of the probabilistic methods. This is yet a relatively simple system and, therefore, it is still being possible to use probability theory [8] to obtain the exact value for its reliability at different target times (i.e.: its survival function). For instance, taking as target times  $t_0 = 0.5, 1.0, 1.5, 2.0$ , the exact reliability of the system at each  $t_0$  is:

$$\begin{aligned} R_s(0.5) &= 0.8156 & R_s(1.0) &= 0.5335 \\ R_s(1.5) &= 0.2799 & R_s(2.0) &= 0.1188 \end{aligned}$$

**TABLE III**  
Minimal path structure for system 2

Path	Composition
1	1 – 2 – 5 – 7
2	1 – 2 – 4 – 6 – 7
3	1 – 3 – 6 – 7
4	1 – 3 – 4 – 5 – 7

On the other hand, SREMS++ has been used to estimate the system reliability and other useful information about the system. Once again, the simulation experiment, consisting in 500,000 iterations, has been carried out using a standard PC (AMD Sempron, 2.2Ghz, 512MB RAM). Results are shown in **Figure 6**.

2\_srems\_outputs - Bloc de notas

Archivo Edición Formato Ver Ayuda

SIMULATION OUTPUTS:

```

elapsed time: 3 sec.
system mean failure time: 1.148337
99% CI for MFT: (1.145826 1.150848)

t0      R(t0)    99% CI
1: 0.815478 (0.814065 0.816891)
2: 0.532328 (0.530510 0.534146)
3: 0.278568 (0.276935 0.280201)
4: 0.118552 (0.117374 0.119730)

path   strong    weak
1: 0.646324 0.536012
2: 0.133642 0.346962
3: 0.168118 0.117026
4: 0.051916 0.000000

comp  key sp   key wp
1: 0.253728 0.084676
2: 0.061876 0.099476
3: 0.054466 0.117026
4: 0.019972 0.124136
5: 0.097142 0.161574
6: 0.078534 0.222826
7: 0.434282 0.190286

```

Fig. 6. SREMS++ outputs for system 2

Again, the computer employed only 3 seconds to run the 500,000 iterations. SREMS++ provided estimated values for the survival function that are very close to the real ones:

$$\begin{aligned} R_S(0.5) &= 0.8155 & R_S(1.0) &= 0.5323 \\ R_S(1.5) &= 0.2786 & R_S(2.0) &= 0.1186 \end{aligned}$$

Furthermore, it also provided the following information: a 99% confidence interval for the system reliability at each target time, an expected value for the system failure-time of 1.1483, and valuable information about paths and components:

- Path 1 has been the strongest one in 65% of the cases, while it has been the weakest one in 54% of the cases (this is not contradictory since components 1 and 7 are repeated in all paths and, therefore, the same path could be, at the same time, the strongest and the weakest one if the system failure is caused by one of those repeated components)
- Component 7 has been the key one in the strongest path in 43% of the cases, while component 6 has been the key one in the weakest path in 22% of the cases. From this information it can be derived that improving the reliability of components 6 and 7 (or duplicating them) should be an efficient way to significantly increase overall system reliability

## X. CASE STUDY 3: A ROUTERS NETWORK

The last system we consider in this paper is a more complex and realistic one (see [Figure 7](#)): a network of 15 routers that connect two servers, A and B, and allow to sent IP packets from the server A to the server B and vice versa.

The associated failure-times distributions are in [Table IV](#). On the other hand, [Table V](#) shows the logical (minimal path) structure of the system.

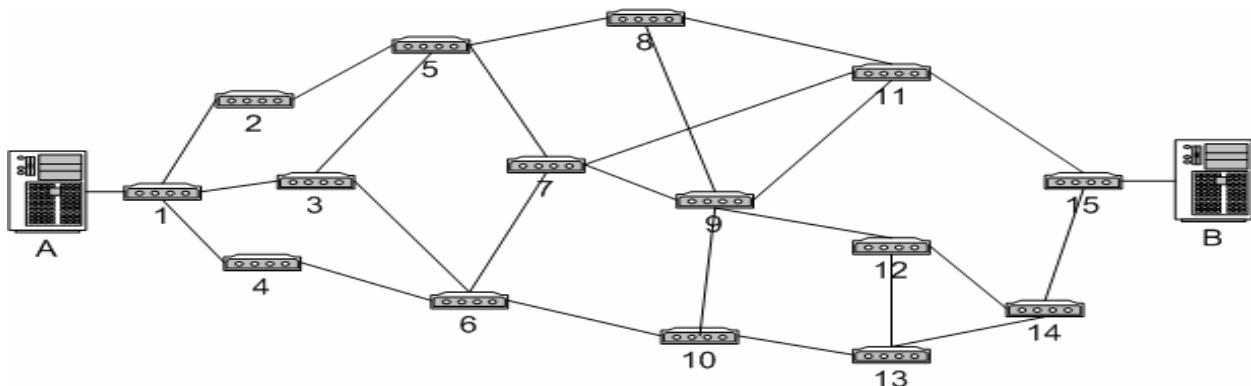


TABLE IV  
Failure times for components in system 3

Component	Distribution	Parameter 1	Parameter 2
1	Weibull	1.6	2.5
2	Weibull	1.7	2.6
3	Weibull	1.5	2.6
4	Weibull	1.6	2.2
5	Weibull	1.3	2.4
6	Weibull	1.2	2.1
7	Weibull	1.2	2.2
8	Weibull	1.9	2.1
9	Weibull	1.3	2.6
10	Weibull	1.5	2.2
11	Weibull	1.6	2.2
12	Weibull	1.2	2.4
13	Weibull	1.2	2.4
14	Weibull	1.1	2.2
15	Weibull	1.6	2.1

It seems obvious that obtaining the survival function of this system using probabilistic or analytical methods, is not an easy task at all. It may take a lot of time and effort and, for some systems, it could even be an impossible task to do. On the other hand, SREMS can obtain, in just a few seconds, good estimations of the system reliability at several target times, together with valuable information about critical components and system failure times.

As **Figure 8** shows, this time the computer employed 8 seconds to run the 500,000 iterations, providing the following estimated values for the system reliability at four different target times:

$$\begin{aligned} R_s(0.5) &= 0.7985 & R_s(1.0) &= 0.4266 \\ R_s(1.5) &= 0.1460 & R_s(2.0) &= 0.0333 \end{aligned}$$

Furthermore, the following notes can also be concluded from the output file (**Figure 9** and **Figure 10**):

- Path 1 has been the strongest one in 48% of the cases, while it has been the weakest one in 29% of the cases.
- Components 1, 15 and 11 are the most critical components of the network, and

should need to be improved or duplicated. Also, special attention should be paid to components 6, 7 and 14.

TABLE V  
Minimal path structure for system 3

Path	Composition
1	1 – 2 – 5 – 8 – 11 – 15
2	1 – 2 – 5 – 8 – 9 – 12 – 14 – 15
3	1 – 2 – 5 – 8 – 9 – 10 – 13 – 14 – 15
4	1 – 2 – 5 – 7 – 11 – 15
5	1 – 2 – 5 – 7 – 9 – 12 – 14 – 15
6	1 – 2 – 5 – 7 – 9 – 10 – 13 – 14 – 15
7	1 – 2 – 5 – 7 – 6 – 10 – 13 – 14 – 15
8	1 – 3 – 5 – 8 – 11 – 15
9	1 – 3 – 5 – 8 – 9 – 12 – 14 – 15
10	1 – 3 – 5 – 8 – 9 – 10 – 13 – 14 – 15
11	1 – 3 – 5 – 7 – 11 – 15
12	1 – 3 – 5 – 7 – 9 – 12 – 14 – 15
13	1 – 3 – 5 – 7 – 9 – 10 – 13 – 14 – 15
14	1 – 3 – 6 – 7 – 11 – 15
15	1 – 3 – 6 – 7 – 9 – 12 – 14 – 15
16	1 – 3 – 6 – 10 – 9 – 11 – 15
17	1 – 3 – 6 – 10 – 9 – 12 – 14 – 15
18	1 – 3 – 6 – 10 – 13 – 14 – 15
19	1 – 4 – 6 – 7 – 11 – 15
20	1 – 4 – 6 – 7 – 9 – 12 – 14 – 15
21	1 – 4 – 6 – 10 – 9 – 11 – 15
22	1 – 4 – 6 – 10 – 9 – 12 – 14 – 15
23	1 – 4 – 6 – 10 – 13 – 14 – 15

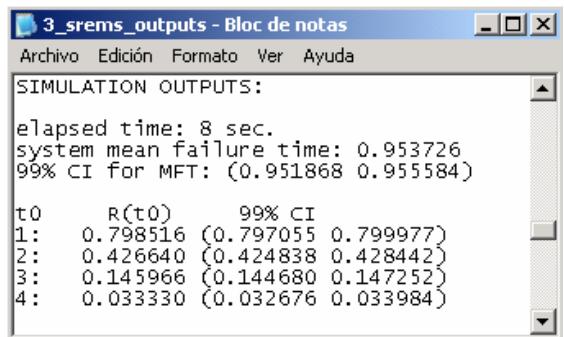


Fig. 8. SREMS++ outputs for system 3 (1 out of 3)

```

path strong weak
1: 0.475436 0.290886
2: 0.058950 0.266892
3: 0.013404 0.146518
4: 0.073074 0.096260
5: 0.011054 0.000000
6: 0.002648 0.000000
7: 0.009892 0.101882
8: 0.065320 0.046380
9: 0.009940 0.000000
10: 0.002542 0.000000
11: 0.018124 0.000000
12: 0.002742 0.000000
13: 0.000730 0.000000
14: 0.101046 0.000000
15: 0.013316 0.000000
16: 0.044500 0.000000
17: 0.007660 0.000000
18: 0.026730 0.000000
19: 0.033754 0.051182
20: 0.004544 0.000000
21: 0.013494 0.000000
22: 0.002604 0.000000
23: 0.008496 0.000000

```

Fig. 9. SREMS++ outputs for system 3 (2 out of 3)

```

comp key sp key wp
1: 0.244250 0.041868
2: 0.013442 0.033100
3: 0.020372 0.046380
4: 0.011556 0.051182
5: 0.072192 0.073138
6: 0.061644 0.101882
7: 0.029428 0.096260
8: 0.028212 0.036084
9: 0.016680 0.065784
10: 0.023842 0.059658
11: 0.102324 0.051380
12: 0.009148 0.086446
13: 0.009542 0.086860
14: 0.035784 0.114662
15: 0.321584 0.055316

```

Fig. 10. SREMS++ outputs for system 3 (3 out of 3)

## XI. CONCLUSIONS

In this paper, a simulation-based method called SREMS has been presented. SREMS can be very helpful for system managers and engineers when determining and improving complex systems reliability. SREMS is able to provide useful information about complex systems reliability (system survival function) and can be applied in most situations where analytical methods are not well suited. Furthermore, it provides complementary information about which components should be improved or duplicated to obtain a significant increment in system reliability. This information will be of significance for providing maintenance services to the buyers of the system.

SREMS has also been implemented as a computer program written in C/C++. It does not matter how many components and how many minimal paths the system contains, the developed program can offer accurate estimates of system reliability and system failure-time parameters.

## REFERENCES

- [1] Banks, J. (ed). 1998. *Simulation: Principles, Methodology, Advances, Applications, and Practice*. New York: John Wiley & Sons.
- [2] Bajenescu, T. I. 1998. Predict the Reliability of Complex Systems by Applying the Monte Carlo Method. In *Proceedings of the 6<sup>th</sup> International Conference on Optimization of Electrical and Electronic Equipments 1998. OPTIM '98*, 789-792. Brasov, Romania.
- [3] Barlow, R., and F. Proschan. 1996. *Mathematical Theory of Reliability*. Philadelphia. Society for Industrial & Applied Mathematics.
- [4] Billinton, R., and P. Wang. 1999. Teaching Distribution Systems Reliability Evaluation Using Monte Carlo Simulation. *IEEE Transactions on Power Systems* 14: 397-403.
- [5] Collet, D. 2003. *Modeling Survival Data in Medical Research*. Boca Raton. Florida: Chapman & Hall/CRC.
- [6] Chisman, J. 1998. Using Discrete Simulation Modeling to study Large-Scale System Reliability/Availability. *Computers and Operations Research* 25: 169-174.
- [7] Chung, C. 2004. *Simulation Modeling Handbook: A Practical Approach*. Boca Raton. Florida: CRC Press.
- [8] Degroot, M. 1988. *Probability and Statistics*. New York: Addison-Wesley.
- [9] Gedam, S.; Beaudet, S. 2000. Monte Carlo Simulation using Excel Spreadsheet for Predicting Reliability of a Complex System. In *Proceedings of the 2000 Annual Reliability and Maintainability Symp.*, 188-193.
- [10] Goel, L., and R. Gupta. 1997. A Windows-based Simulation Tool for Reliability Evaluation of Electricity Generating Capacity. *International Journal of Engineering Education* 13: 347-357.
- [11] Goldfeld, A.; Dubi, A. 1987. Monte Carlo Methods in Reliability Engineering. *Quality and Reliability Engineering Int.*, Vol. 3.
- [12] Hoyland, A., and M. Rausand. 1994. *System Reliability Theory: Models and Statistical Methods*. New York: John Wiley-Interscience.
- [13] Juan, A., and A. Vila. 2002. SREMS: System Reliability using Monte Carlo Simulation with VBA and Excel. *Quality Engineering*, 15: 333-34.

- [14] Kamat, S., and M. Riley. 1975. Determination of Reliability Using Event-Based Monte Carlo Simulation, *IEEE Transactions on Reliability* 24: 73-75.
- [15] Kotler, P. 2003. *Marketing Management*. Delhi: Pearson Education.
- [16] Kovalenko, I.N., N.Y. Kuznetsov, and P.A. Pegg. 1997. *Mathematical Theory of Reliability of Time Dependent Systems with Practical Applications*. New York: John Wiley & Sons, Inc.
- [17] Law, A., and D. Kelton. 2000. *Simulation Modeling & Analysis*. New York: McGraw-Hill.
- [18] Leemis, L. 2003. Input Modeling. In *Proceedings of the Winter Simulation Conference*, ed. D.J. Medeiros, E.F. Watson, J.S. Carson, and M.S. Manivannan, pp. 62-73. Available via <<http://www.informs.cs.org>> [accessed January 4, 2005].
- [19] Levin, M., and T. Kalal. 2003. *Improving Product Reliability: Strategies and Implementation*. New York: John Wiley & Sons.
- [20] Lin, J., and C. Donaghey. 1993. A Monte Carlo Simulation to Determine Minimal Cut Sets and System Reliability. In *Proceedings Annual Reliability and Maintainability Symposium*, 246-249. Atlanta, GA.
- [21] L'Ecuyer, P. 2002. Random Numbers. In the *International Encyclopedia of the Social and Behavioral Sciences*, ed. N.J. Smelser and P.B. Baltes, 12735-12738. Oxford.
- [22] Meeker, W., and L. Escobar. 1998. *Statistical Methods for Reliability Data*. New York: John Wiley & Sons.
- [23] Pham, H. (ed) 2003. *Handbook of Reliability Engineering*. New York: Springer-Verlag.
- [24] Ross, S.M. 2001. *Simulation*. San Diego, California: Academic Press.
- [25] Traub, R. 1994. *Reliability for the Social Sciences: Theory and Applications*. London: Sage Publications.
- [26] Wang, H., and H. Pham. 1997. Survey of Reliability and Availability Evaluation of Complex Networks using Monte Carlo Techniques. *Microelectronics Reliability* 37: 187-209.
- [27] Willis, R. 2000. Comparison of Reliability-Availability Simulators. Available via <[www.enre.umd.edu/rms/simulators.htm](http://www.enre.umd.edu/rms/simulators.htm)> [accessed February 4, 2005]



# Nonconvex optimization using an adapted linesearch

Alberto Olivares\*, Javier M. Moguerza\* and Francisco J. Prieto†

\*Universidad Rey Juan Carlos/Departamento de Informática Estadística y Telemática

C/ Tulipan s/n, 28933 Mostoles (Madrid)

Email: javier.moguerza@urjc.es, alberto.olivares@urjc.es

†Universidad Carlos III de Madrid/Departamento de Estadística

C/ Madrid 126, 28903 Getafe (Madrid)

Email: franciscojavier.prieto@uc3m.es

**Abstract**—This paper describes a new algorithm for the solution of nonconvex unconstrained optimization problems, with the property of converging to points satisfying the second-order optimality conditions. The algorithm is based on a procedure which, from two descent directions (a Newton-type direction and a direction of negative curvature), selects in each iteration the linesearch model best adapted to the properties of these directions. The paper also presents the results of numerical experiments that illustrate its practical efficiency.

**Keywords**—Newton's method, unconstrained optimization, negative curvature.

## I. INTRODUCTION

THE goal of this work is the efficient solution of optimization problems having the form

$$\min_x f(x), \quad (1)$$

where  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  is a twice-differentiable function.

Many algorithms have been proposed in the literature to solve problems of the form (1), but only a few attempt to use the second-order information available in the Hessian matrix of  $f$ . This information may play a very significant role in the design of efficient algorithms. Unconstrained optimization problems are usually solved by applying algorithms based on Newton's method. These methods, when properly implemented, have well-known convergence properties, and in particular they can be shown to be globally convergent to first-order KKT points. There are three broad classes of procedures to ensure these convergence properties: linesearch methods, trust-region methods and filter methods. The method described in this paper belongs to the class of linesearch procedures; in particular, we describe a linesearch that uses second-order information in an efficient manner. This information is introduced through the computation of a negative curvature direction in each iteration. Note that along these negative curvature directions the quadratic model is unbounded below, and this property

offers the potential for a significant reduction in the value of the objective function.

We wish to introduce a methodology for the linesearch that exploits any nonconvexity that the objective function may present locally. To this end, in each iteration we compute a pair of directions  $(s_k, d_k)$ . The first one,  $s_k$ , is a modified Newton direction that ensures fast convergence under convexity assumptions. The second one,  $d_k$ , is a negative curvature direction that allows the algorithm to move in an efficient manner away from local nonconvex regions.

One of the first proposals to take into account second-order information is that of Fiacco [6]. More recently, the use of this information has been studied by Fletcher and Freeman [7], Gill and Murray [8], or Mukai and Polak [21], among others. In a linesearch context the work of McCormick [17] is particularly relevant. In this work, the Armijo rule for the termination of the linesearch is adapted and modified to include negative curvature information. Moré and Sorensen [20] follow a similar approach, and their work is the first one to use explicitly negative curvature directions in the solution of specific instances of unconstrained problems. Another approach is discussed in Goldfarb [10]. This proposal is appropriate in those cases where the current iterate is a negative definite point. More recently, Moguerza and Prieto [18] have extended the methodology of Moré and Sorensen to constrained problems within an interior point framework.

Both in Moré and Sorensen [20] and in Moguerza and Prieto [18], the next iterate is obtained through a backtracking procedure along a second-order curve combining the directions. In Moguerza and Prieto [19], the search is conducted on a curve obtained from the approximate solution of an ODE related to the problem. Using a different approach, Gould et al. [11] compute the directions using a conjugate gradient method: in each iteration the best direction is chosen and a standard linesearch is conducted. Another method based in the selection of directions is suggested by San Matías and

Roma [23]. The relation between our proposal and [23] will be discussed later on in this paper.

The main aim of this work is to identify a criterion such that in each iteration the best search procedure is chosen. Depending on the results from the application of this criterion, the algorithm will apply either a curvilinear search similar to that proposed by Moré and Sorensen [20], combining both search directions, or a standard linesearch similar to that proposed by Gould et al. [11], using just one of the two directions.

The structure of the paper is as follows: In Section 2 we introduce some basic definitions as well as a method to compute the descent directions. Section 3 describes the procedure to select the steplength and discusses its convergence properties. Section 4 introduces the condition to identify the search method to be used in a given iteration. In Section 5 we discuss implementation details and present a general scheme for the algorithm. Section 6 shows the results from the computational experiments. Finally, Section 7 presents some conclusions.

## II. BASIC DEFINITIONS AND SEARCH DIRECTION COMPUTATIONS

For the remainder of the paper we will assume that the following regularity properties hold for problem (1) and the initial point of the algorithm,  $x_0$ .

- A1. The level set  $\mathcal{L}(x_0) = \{x : f(x) \leq f(x_0)\}$  is compact.
- A2. The objective function  $f$  is three times continuously differentiable on an open set that contains  $\mathcal{L}(x_0)$ .

We start by introducing some definitions that will be used in the computation of appropriate search direction pairs  $(s, d)$ .

a) *Definition 1:* Let  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  satisfy conditions A1 and A2.

- a. A point  $x$  is indefinite if  $\nabla^2 f(x)$  has at least one negative eigenvalue.
- b. If  $x$  is indefinite, a vector  $d$  is a direction of negative curvature if  $d^T \nabla^2 f(x)d < 0$ .
- c. At an indefinite point  $x$ , a pair of vectors  $(s, d)$  is said to be a descent pair if  $\nabla f(x)^T s \leq 0$ ,  $\nabla f(x)^T d \leq 0$  and  $d^T \nabla^2 f(x)d < 0$ .

If  $x$  is not indefinite, then  $(s, d)$  is a descent pair if  $\nabla f(x)^T s < 0$ ,  $\nabla f(x)^T d \leq 0$  and  $d^T \nabla^2 f(x)d = 0$ .

Assuming that  $x$  is an indefinite point, a possible choice for a descent pair would be  $s = -\nabla f(x)$  and  $d = \pm u_n$ , where  $u_n$  denotes an eigenvector associated to a negative eigenvalue of  $\nabla^2 f(x)$ . The sign for  $u_n$  is chosen to ensure that  $\nabla f(x)^T d \leq 0$ . If  $x$  is not

indefinite and  $\nabla f(x) \neq 0$ , then second-order information is not relevant. A possible descent pair would be the one defined from  $s = -\nabla f(x)$  and  $d = 0$ . The only case in which there exists no descent pair corresponds to points  $x$  satisfying  $\nabla f(x) = 0$  and  $\nabla^2 f(x)$  positive semidefinite, that is, satisfying the necessary second-order conditions.

From a more practical point of view, in a given point  $x$  a direction  $s$  is assumed to be a sufficient descent direction if there exist constants  $c_1$  and  $c_2$  such that

$$s^T g(x) \leq -c_1 \|g(x)\|^2 \quad (2)$$

and

$$\|s\| \leq c_2 \|g(x)\|, \quad (3)$$

where  $g(x) \equiv \nabla f(x)$ .

Analogously,  $d$  is a sufficient direction of negative curvature at  $x$  if there exists a constant  $c_3$  such that

$$d^T g(x) \leq 0,$$

$$d^T H d \leq \min(0, \theta \lambda_{\min}(H(x)) + \nu(g(x)) \|d\|^2),$$

$$\|d\| \leq c_3,$$

where  $\theta \in (0, 1)$ ,  $\lambda_{\min}(H(x))$  denotes the smallest eigenvalue of the Hessian matrix  $H(x) \equiv \nabla^2 f(x)$ , and the continuous and uniformly bounded function  $\nu(x) \geq 0$  satisfies also  $\nu(x) \rightarrow 0$  whenever  $x \rightarrow 0$ . This specific form of the condition was introduced by Lucidi et al. [16].

We will assume that we have negative curvature directions that satisfy the stronger condition

$$\begin{aligned} d^T g(x) &\leq 0, \\ d^T H d &\leq \min(0, \theta \lambda_{\min}(H(x))), \\ \|d\| &\leq c_3. \end{aligned} \quad (4)$$

Conditions (2) and (3) are the standard ones for Newton-type directions. Condition (4) ensures that  $d$  contains information related to the smallest eigenvalue of the Hessian matrix.

### A. Computing a modified Newton direction

Under assumptions A1 and A2 it is possible to construct a local quadratic model for the objective function from the corresponding Taylor series expansion at all iterates  $x_k \in \mathcal{L}(x_0)$  as:

$$f(x_k + p) \simeq f(x_k) + g(x_k)^T p + \frac{1}{2} p^T H(x_k) p. \quad (5)$$

We introduce the notation

$$\Phi_k(s) \equiv g(x_k)^T s + \frac{1}{2} s^T H(x_k) s, \quad (6)$$

for the right-hand side of (5), omitting the constant term  $f(x_k)$ . As shown for example in Gill et al. [9],  $s_k$  is a stationary point of  $\Phi_k$  if it satisfies the linear system of equations

$$H(x_k)s_k = -g(x_k). \quad (7)$$

The direction  $s_k$ , obtained as a solution of (7), is known as the Newton direction. An efficient method to compute this direction uses the modified Cholesky factorization proposed by Gill and Murray [8]. This modification adapts the Cholesky procedure to the case when the Hessian matrix is indefinite, and negative curvature directions exist. Briefly, given the symmetric matrix  $H(x)$ , the factorization computes a lower triangular matrix  $L$  with unit diagonal, and diagonal matrices  $D = \text{diag}(d_i)$  and  $E = \text{diag}(\epsilon_i)$  having  $d_i \geq 0$  and  $\epsilon_i \geq 0$ , such that  $H(x) + E = LDL^T$  and  $E$  is small. In particular, if  $H(x)$  is sufficiently positive definite, then  $E = 0$ . The modified Newton direction  $s$  can be obtained by solving the system

$$LDL^T s = -g(x).$$

### B. Computing a direction of negative curvature

For an iterate  $x_k \in \mathcal{L}(x_0)$ , a direction of negative curvature  $d_k$  will be useful if it satisfies condition (4). Note that from (4) it follows that

$$d_k^T H(x_k) d_k \rightarrow 0 \text{ and } g(x_k) \rightarrow 0 \Rightarrow \lambda_{H_k} \rightarrow 0, \quad (8)$$

where  $\lambda_{H_k}$  denotes the smallest eigenvalue of  $H(x_k)$  if  $x_k$  is indefinite, and zero otherwise. The motivation underlying (4) is to ensure that the iterates move towards regions of local convexity for  $f$ .

If the complete spectral decomposition of the Hessian matrix  $H(x_k)$  were available, it would be easy to obtain directions  $d_k$  satisfying condition (4). Unfortunately, the determination of a complete system of eigenvectors and eigenvalues for this matrix can be computationally very expensive.

Alternatively, the modified Cholesky factorization of Gill and Murray [8] allows the computation of a negative curvature direction simultaneously with the determination of a modified Newton direction; in this manner, the computational cost to obtain a descent pair is reasonable. If the factorization does not detect any negative curvature information for the current iterate, it is enough to define  $d_k = 0$ . For the details on this procedure to compute directions of negative curvature, and its properties, see Gill et al. [9] and Moré and Sorensen [20].

### III. THE CURVILINEAR SEARCH

In this section we present some convergence results for a search based on the use of descent pairs, as defined in Section II. Many of these results are adapted from those in Moré and Sorensen [20].

When using a search method to ensure global convergence, in each iteration a parametric curve  $x(\alpha)$  is built from the directions computed in that iteration. The goal is to find a value of the parameter  $\alpha > 0$  such that

$$f(x(\alpha)) < f(x). \quad (9)$$

In our case, in each iteration  $k$  we compute a descent pair  $(s_k, d_k)$ , and from it we define a general parametric curve as:

$$\mathcal{C} = \{x_k(\alpha) \equiv x_k + \phi_1(\alpha)s_k + \phi_2(\alpha)d_k, \quad \alpha \geq 0\}, \quad (10)$$

where  $\phi_1(\alpha)$  and  $\phi_2(\alpha)$  are appropriate (nonnegative) weight functions for the linear combination. We build the curve using both descent directions, in order to use simultaneously the information available in them.

We now analyze the properties of this search and reasonable choices for the functions  $\phi_1$  and  $\phi_2$ . We start by studying the problem of finding an appropriate value for  $\alpha$ , ensuring the convergence of the algorithm, from the analysis of the univariate function

$$\psi_k(\alpha) \equiv f(x_k(\alpha)). \quad (11)$$

We need  $\psi_k$  to satisfy some basic properties: i)  $\psi_k$  should be a continuous function of  $\alpha$ , and as a consequence both  $\phi_1$  and  $\phi_2$  should also be continuous functions of  $\alpha$ ; ii) we also have to ensure that there exists a value of  $\alpha > 0$  such that a condition related to (9) and ensuring sufficient descent is satisfied. The following Lemma shows that this is the case as long as either  $\psi'_k(0) < 0$  or  $\psi'_k(0) \leq 0$  and  $\psi''_k(0) < 0$  hold. The Lemma extends the results in Moré and Sorensen [20], as it considers general functions  $\phi_1$  and  $\phi_2$  in (10) with the only assumption that they are three times differentiable.

The Lemma has two parts, the first one presents an existence result that is sufficient for convergence under a backtracking search approach. The second part, including two termination conditions, can be used to prove convergence independently of the search procedure used in the algorithm.

*Lemma 1:* Let  $\psi(\alpha) : \mathbb{R} \rightarrow \mathbb{R}$  defined as in (11) be a three-times differentiable function on an open interval  $I$  containing  $[0, 1]$ , and let  $\gamma_1 \in (0, 1)$ , a given scalar. If either  $\psi'(0) < 0$  or  $\psi'(0) \leq 0$  and  $\psi''(0) < 0$  then there exists a scalar  $\hat{\eta} > 0$  in  $I$  such that

$$\psi(\alpha) \leq \psi(0) + \gamma_1 (\psi'(0)\alpha + \frac{1}{2}\psi''(0)\alpha^2)$$

holds for all  $\alpha \in [0, \hat{\eta}]$ , where  $\hat{\eta}$  is bounded away from zero.

Furthermore, let  $\gamma_2$  be another given scalar such that  $1 > \gamma_2 > \gamma_1 > 0$ . Either  $\psi(1) \leq \psi(0) + \gamma_1 (\psi'(0) + \frac{1}{2}\psi''(0))$  holds or there exists a scalar  $\tilde{\eta} > 0$  satisfying  $\tilde{\eta} \leq \hat{\eta}$ , such that both

$$\psi(\alpha) \leq \psi(0) + \gamma_1 (\psi'(0)\alpha + \frac{1}{2}\psi''(0)\alpha^2), \quad (12)$$

$$\psi'(\alpha) \geq \gamma_2 \min(\psi'(0) + \psi''(0)\alpha, 0) \quad (13)$$

hold simultaneously for all  $\alpha \in [\tilde{\eta}, \hat{\eta}]$ . The value of  $\tilde{\eta}$  is bounded away from zero.

*Proof:* Let

$$\Psi(\alpha) \equiv \psi(\alpha) - \psi(0) - \gamma_1 (\alpha\psi'(0) + \frac{1}{2}\alpha^2\psi''(0)).$$

Note that, from the properties of  $\psi$  and this definition,  $\Psi(0) = 0$ ,  $\Psi'(0) = (1 - \gamma_1)\psi'(0) \leq 0$  and  $\Psi''(0) = (1 - \gamma_1)\psi''(0) < 0$  if  $\Psi'(0) = 0$ . Also,  $\Psi$  is three-times differentiable on  $I$ .

a) Consider the case when  $\psi'(0) < 0$ . From the Taylor series expansion for  $\Psi$  around zero and  $\alpha \in I$  we have

$$\Psi(\alpha) = \Psi'(0)\alpha + \frac{1}{2}\Psi''(\zeta_1)\alpha^2,$$

for some  $\zeta_1 \in [0, \alpha]$ . Let  $\bar{I}$  denote a closed subinterval of  $I$  that contains zero. Then  $\psi$  and its first, second and third derivatives are bounded on  $\bar{I}$ . Let  $K_1$  be a bound for  $\psi''(\alpha)$  on  $\bar{I}$  and  $\beta_1 = -2\psi'(0)/K_1$ ; for all  $\alpha \in (0, \beta_1)$  it holds that  $\Psi(\alpha) < 0$  and it holds that  $\hat{\eta} \geq -2\psi'(0)/K_1$ .

If  $\psi'(0) < 0$  but  $\psi(1) \leq \psi(0) + \gamma_1 (\psi'(0) + \frac{1}{2}\psi''(0))$  does not hold, we have that  $\Psi(1) > 0$  and by continuity there exists a positive value  $\beta_2 < 1$  such that  $\Psi(\beta_2) = 0$  and  $\beta_2$  is the first such value. From the mean-value theorem there exists a positive value  $\beta_3 < \beta_2$ , the smallest positive zero of  $\Psi'$ ,  $\Psi'(\beta_3) = 0$ . Note that (12) holds for all  $\alpha \in [0, \beta_2]$ .

We study two cases: i) If  $\psi'(0) + \beta_3\psi''(0) \leq 0$ , then from  $\gamma_2 > \gamma_1$  it holds that

$$\begin{aligned} 0 &= \psi'(\beta_3) - \gamma_1 (\psi'(0) + \beta_3\psi''(0)) \\ &\leq \psi'(\beta_3) - \gamma_2 (\psi'(0) + \beta_3\psi''(0)) \\ &= \psi'(\beta_3) - \gamma_2 \min(\psi'(0) + \beta_3\psi''(0), 0), \end{aligned}$$

implying that both (12) and (13) hold for  $\alpha = \beta_3$ . ii) If  $\psi'(0) + \beta_3\psi''(0) > 0$ , from  $\Psi'(\beta_3) = 0$  it holds that

$$\begin{aligned} \psi'(\beta_3) &= \gamma_2(\psi'(0) + \beta_3\psi''(0)) \\ &> \min(\psi'(0) + \beta_3\psi''(0), 0) = 0, \end{aligned}$$

and (13) also holds for  $\alpha = \beta_3$ .

From  $\Psi'(\beta_3) = 0$  and Taylor series expansions it follows that

$$\gamma_1 (\psi'(0) + \beta_3\psi''(0)) = \psi'(\beta_3) = \psi'(0) + \beta_3\psi''(\zeta_3), \quad (14)$$

for some  $\zeta_3 \in [0, \beta_3]$ . From  $\beta_3 < 1$  and our assumptions, there exists a value  $K_3$  such that  $|\psi''(\alpha)| \leq K_3$  for any  $\alpha \in [0, \beta_3]$ . Any  $\alpha$  that satisfies (13) will also satisfy  $\alpha \geq \beta_3$  and from (14)

$$\alpha \geq \beta_3 \geq -\frac{(1 - \gamma_1)\psi'(0)}{2K_3}. \quad (15)$$

b) Consider now the case when  $\psi'(0) \leq 0$  and  $\psi''(0) < 0$ . Using again Taylor series expansions for  $\alpha \in [0, 1]$ ,

$$\Psi(\alpha) = \Psi'(0)\alpha + \frac{1}{2}\Psi''(0)\alpha^2 + \frac{1}{6}\Psi'''(\zeta_2)\alpha^3,$$

for some  $\zeta_2 \in [0, \alpha]$ . Let  $K_2$  be a bound for  $\psi'''(\alpha)$  on  $[0, 1]$  and  $\beta_4 = -3(1 - \gamma_1)\psi''(0)/K_2$ . Then, for any  $\alpha \in (0, \beta_4)$  it holds that  $\frac{1}{2}\Psi''(0)\alpha^2 + \frac{1}{6}\Psi'''(\zeta_2)\alpha^3 < 0$  and as  $\Psi'(0) \leq 0$ , (12) holds for all  $\alpha \in (0, \beta_4)$  and we have that

$$\hat{\eta} \geq -3(1 - \gamma_1)\psi''(0)/K_2. \quad (16)$$

If in addition  $\psi(1) > \psi(0) + \gamma (\psi'(0) + \frac{1}{2}\psi''(0))$  holds, we apply an argument similar to the preceding case. As  $\Psi(0) = 0$ ,  $\Psi'(0) \leq 0$ ,  $\Psi''(0) < 0$  and  $\Psi(1) > 0$ , by continuity there exists a positive value  $\beta_5 < 1$  such that  $\Psi(\beta_5) = 0$  and  $\beta_5$  is the first such value. From the mean-value theorem there exists a positive value  $\beta_6 < \beta_5$ , the smallest positive zero of  $\Psi'$ ,  $\Psi'(\beta_6) = 0$ . Note that (12) holds for all  $\alpha \in [0, \beta_5]$ .

But under the conditions on  $\psi'(0)$  and  $\psi''(0)$ , we have  $\psi'(0) + \beta_6\psi''(0) < 0$ , and from  $\gamma_2 > \gamma_1$  it holds that

$$\begin{aligned} 0 &= \psi'(\beta_6) - \gamma_1 (\psi'(0) + \beta_6\psi''(0)) \\ &\leq \psi'(\beta_6) - \gamma_2 (\psi'(0) + \beta_6\psi''(0)) \\ &= \psi'(\beta_6) - \gamma_2 \min(\psi'(0) + \beta_6\psi''(0), 0), \end{aligned}$$

implying again that both (12) and (13) hold for  $\alpha = \beta_6$ .

Again, from  $\Psi'(\beta_6) = 0$  and Taylor series expansions,

$$\begin{aligned} \gamma_1 (\psi'(0) + \beta_6\psi''(0)) \\ = \psi'(\beta_6) = \psi'(0) + \beta_6\psi''(0) + \frac{1}{2}\beta_6^2\psi'''(\zeta_4), \end{aligned}$$

for some  $\zeta_4 \in [0, \beta_6]$ . As  $\beta_6 < 1$ , let  $K_4$  denote a bound for  $\psi'''(\alpha)$  on  $[0, \beta_6]$ . Using  $\psi'(0) \leq 0$  it follows that

$$\begin{aligned} &-(1 - \gamma_1)\beta_6\psi''(0) \\ &\leq -(1 - \gamma_1) (\psi'(0) + \beta_6\psi''(0)) \\ &= \frac{1}{2}\beta_6^2\psi'''(\zeta_4) \leq \frac{1}{2}\beta_6^2K_4. \end{aligned} \quad (17)$$

Any  $\alpha$  that satisfies (13) will also satisfy  $\alpha \geq \beta_6$  and from (18)

$$\alpha \geq \beta_6 \geq -\frac{2(1 - \gamma_1)\psi''(0)}{K_4}. \quad (18)$$

■

The following Corollary assumes a backtracking search to compute  $\alpha$  conducted as described in [9], that is, for a given positive constant  $\delta < 1$  we define the steplength  $\alpha$  as the first value in the sequence  $\{\delta^i\}_{i=0}^\infty$  for which condition (12) is satisfied.

*Corollary 1:* If a backtracking search with parameter  $\delta$  is used, the value of  $\alpha$  computed from the search,  $\hat{\alpha}$ , satisfies

$$\hat{\alpha} \geq \delta\hat{\eta},$$

and is bounded away from zero. If a general search is conducted to compute a value satisfying both (12) and (13) then the computed value  $\tilde{\alpha}$  satisfies

$$\tilde{\alpha} \geq \tilde{\eta}.$$

*Proof:* This result follows for the backtracking search from  $\alpha_k^{i+1} = \delta\alpha_k^i$  and  $\alpha_k^i$  not satisfying (12) and consequently satisfying  $\alpha_k^i > \hat{\eta}$ , and the bound for  $\hat{\eta}$ . For the general search the result follows directly from the bounds for  $\tilde{\eta}$ . ■

The preceding results establish the existence of sufficient descent given certain conditions on the derivatives of  $\psi$ . The satisfaction of these conditions depends on the properties of the functions  $\phi_1$  and  $\phi_2$ , and those of the descent pair  $(s, d)$ . We now analyze our requirements on these functions. From the definition of  $\psi(\alpha)$  it follows that

$$\begin{aligned}\psi'(0) &= \phi'_1(0)\nabla f(x)^T s + \phi'_2(0)\nabla f(x)^T d \\ \psi''(0) &= (\phi'_1(0))^2 s^T \nabla^2 f(x)s + (\phi'_2(0))^2 d^T \nabla^2 f(x)d \\ &\quad + 2\phi'_1(0)\phi'_2(0)s^T \nabla^2 f(x)d + \phi''_1(0)\nabla f(x)^T s \\ &\quad + \phi''_2(0)\nabla f(x)^T d,\end{aligned}$$

and from the properties of the descent pair  $(s, d)$  it follows that

$$\begin{aligned}\phi'_1(0) \geq 0, \phi'_2(0) \geq 0 &\Rightarrow \psi'(0) \geq 0 \\ \phi'_1(0) = 0, \phi'_2(0) \geq 0, \phi''_2(0) \geq 0 &\Rightarrow \psi''(0) \geq 0.\end{aligned}$$

If in addition to these conditions,  $\phi'_2(0) > 0$  and  $\phi''_1(0) > 0$  then  $\psi''(0) < 0$ . We also need the third derivatives to be bounded on  $[0, 1]$ .

We assume in what follows that the functions  $\phi_1$  and  $\phi_2$  satisfy the following condition

*C1.* It holds that

$$\phi'_1(0) = 0, \quad \phi'_2(0) > 0, \quad \phi''_1(0) > 0, \quad \phi''_2(0) \geq 0.$$

Note that under *C1* for any descent pair  $(s, d) \neq 0$  we have

$$\begin{aligned}\psi'(0) &= \phi'_2(0)\nabla f(x)^T d \leq 0 \\ \psi''(0) &= (\phi'_2(0))^2 d^T \nabla^2 f(x)d + \phi''_1(0)\nabla f(x)^T s \\ &\quad + \phi''_2(0)\nabla f(x)^T d < 0.\end{aligned}\tag{19}$$

Also, given  $x_k$  and a descent pair  $(s_k, d_k)$ , we assume that the next iterate  $x_{k+1}$  is defined as

$$x_{k+1} = x_k(\alpha_k),$$

where  $\alpha$  is chosen to satisfy the following condition:

- C2.* The steplength  $\alpha_k$  is defined to be equal to one if  $\psi(1)$  satisfies (12); otherwise,  $\alpha_k$  is computed to satisfy both (12) and (13).

Finally, we need a condition on the descent pair that is sufficiently flexible to allow us to apply an adapted search that makes use of the best directions in each iteration. The condition, replacing (2), (3) and (4) is the following one:

- C3.* Let  $\bar{c}_1, \bar{c}_2, \bar{c}_3$  and  $\bar{\theta}$  be nonnegative constants. For all iterations  $k$ , the descent pair  $(s_k, d_k)$  satisfies the conditions

$$\|s_k\| \leq \bar{c}_1 \|g_k\|, \quad g_k^T s_k \leq 0,$$

$$g_k^T d_k \leq 0, \quad \|d_k\| \leq \bar{c}_2,$$

and defining

$$w_k = \begin{cases} s_k^T g_k / \|g_k\| & \text{if } g_k \neq 0 \\ 0 & \text{otherwise,} \end{cases}$$

$$\min(w_k, d_k^T H_k d_k) \leq \min(-\bar{c}_3 \|g_k\|, \bar{\theta} \lambda_{H_k}).\tag{20}$$

In the following section this condition will be shown to be satisfied by the choice of directions used in the proposed algorithm.

The following result shows that, under these conditions and our initial assumptions the procedure is globally convergent.

*Theorem 1:* Let  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  in (1) satisfy assumptions *A1* and *A2*. Consider the sequence  $\{x_k\}$  generated under condition *C2* using a descent pair  $(s_k, d_k)$  satisfying condition *C3*. Then,

$$\lim_{k \rightarrow \infty} g(x_k) = 0,\tag{21}$$

and if the first-order KKT points of  $f$  in  $\mathcal{L}(x_0)$  satisfy the sufficient second-order condition,

$$\lim_{k \rightarrow \infty} \lambda_{H_k} = 0.\tag{22}$$

*Proof:* From *C2* and (12), all iterates remain in  $\mathcal{L}(x_0)$ . As a consequence of assumptions *A1* and *A2*, the sequence  $\{f(x_k)\}$  is bounded below, and the sequence  $\{x_k\}$  has convergent subsequences.

Consider any of these convergent subsequences, and add condition (12) for all iterations along the sequence up to a given iteration  $r$  in the subsequence, to obtain

$$f(x_r) - f(x_0) \leq \gamma_1 \sum_{k=0}^{r-1} (\psi'_k(0)\alpha_k + \frac{1}{2}\psi''_k(0)\alpha_k^2).$$

Taking limits as  $r \rightarrow \infty$ , using the boundedness of  $f$  and the signs of  $\alpha$ ,  $\psi'$  and  $\psi''$  it follows that

$$\psi'_k(0)\alpha_k \rightarrow 0 \quad \text{and} \quad \psi''_k(0)\alpha_k^2 \rightarrow 0.$$

From condition  $C3$ , if  $s_k = d_k = 0$  for some iteration  $k$ , then  $g_k = 0$  and  $\lambda_{H(x)} = 0$ , implying that  $x_k$  satisfies the second-order necessary conditions. If the descent pair is not equal to zero in any iteration, from  $C1$  we have  $\psi'_k(0) \leq 0$  and  $\psi''_k(0) < 0$  for all  $k$ . For these iterations from Corollary 1, (16) and  $\psi''_k(0) < 0$ , if we conduct a backtracking search we have that

$$\psi''_k(0)\alpha_k^2 \leq \frac{9\delta^2(1-\gamma_1)^2(\psi''_k(0))^3}{K_2^2} \leq 0,$$

and if we conduct a general search, from (18),

$$\psi''_k(0)\alpha_k^2 \leq \frac{4(1-\gamma_1)^2(\psi''_k(0))^3}{K_4^2} \leq 0,$$

and if  $\mathcal{K}_2$  is an infinite subsequence, from  $\psi''_k(0)\alpha_k^2 \rightarrow 0$  and the preceding bounds it must hold along it that  $\psi''_k(0) \rightarrow 0$ . But from (19) this implies

$$d_k^T H(x_k) d_k \rightarrow 0 \quad \text{and} \quad g(x_k)^T s_k \rightarrow 0.$$

If  $g(x_k) = 0$  along an infinite subsequence, from  $C3$  we have  $\lambda_{H_k} \rightarrow 0$ . Otherwise, assume  $g(x_k) \neq 0$  for all  $k$  in the subsequence; we may have that either  $g(x_k)^T s_k / \|g(x_k)\| \rightarrow 0$  along all subsequences, in which case condition  $C3$  implies  $\min(-\bar{c}_3\|g_k\|^2, \bar{\theta}\lambda_{H_k}) \rightarrow 0$  and

$$\|g(x_k)\| \rightarrow 0, \quad \lambda_{H_k} \rightarrow 0,$$

or along a subsequence there exists a positive constant  $K$  such that for all  $k$  in the subsequence  $g(x_k)^T s_k \leq -K\|g(x_k)\| \leq 0$ , but this implies  $\|g(x_k)\| \rightarrow 0$ . From  $\|s_k\| \leq \bar{c}_1\|g_k\|$  we have that  $0 \geq g(x_k)^T s_k / \|g_k\| \geq -\|s_k\| \geq -\bar{c}_1\|g_k\|$  and we must also have  $g(x_k)^T s_k / \|g_k\| \rightarrow 0$ , contradicting our assumption. Thus, in all cases we have

$$\|g(x_k)\| \rightarrow 0, \quad \lambda_{H_k} \rightarrow 0,$$

and the desired result follows from these limits and assumption  $A2$ . ■

The preceding convergence results provide a justification for the algorithm of interest in this paper but, in order to have a practical algorithm, we still need to define in a precise manner both the form of the functions  $\phi_1$  and  $\phi_2$  and that of the descent pair  $(s_k, d_k)$ . Regarding the functions, the simplest choice that satisfies the desired conditions  $C1$  is  $\phi_1(\alpha) \equiv \alpha^2$  and  $\phi_2(\alpha) = \alpha$ ; we will use it in the remainder of the paper. The choice of the directions in the descent pair will be discussed in the following section.

#### IV. THE ADAPTED SEARCH

As we mentioned before, in each iteration the proposed algorithm computes a descent pair  $(s_k, d_k)$ , but before carrying out a search to satisfy  $C2$ , it decides what information in the descent pair will be used in the search. This approach is similar to the one discussed in Gould et al. [11], with the main difference that, while they only consider two alternatives, we consider the three following possibilities:

- To conduct a linesearch based just on the use of the modified Newton direction, using the descent pair  $(s_k, 0)$ .
- To conduct a linesearch based only on the negative curvature direction, using the descent pair  $(0, d_k)$ .
- To carry out a curvilinear search combining both directions, making use of the full descent pair  $(s_k, d_k)$ .

Our choice will be made from a comparison of the descent provided by each direction on a quadratic model of the objective function. We wish to select the alternative that offers the most significant descent information. A similar proposal to this scheme was suggested in San Matías and Roma [23], where the proposed method would select one of these three different possibilities from a comparison of the descent provided on the quadratic model of the objective function. The main differences between this work and the proposal in [23] are the way in which the weights for the directions are chosen and the scaling of the direction of negative curvature.

As previously mentioned, before analyzing the descent information, it is important to consider the scaling of each of the directions. The manner in which they are computed and their properties usually imply a significant difference in their sizes. The Newton direction  $s_k$  ensures good local convergence properties for unit steplengths, while the negative curvature direction has no specific scale attached to it (the quadratic model used to define it would imply an infinite steplength). The recommended procedure is to scale the direction of negative curvature to a size that is not very different from that of the Newton direction, see [18].

Consider now the choice of directions for the search. Let  $Q_k$  denote the local quadratic model for the objective function around  $x_k$ , defined as:

$$Q_k(z) = g_k^T z + \frac{1}{2}z^T H_k z. \quad (23)$$

Gould et al. [11] study the two directions in the descent pair  $(s_k, d_k)$ , and select the one that provides the largest descent ratio, measured against the quadratic model. Thus, the Newton direction  $s_k$  is chosen whenever the

condition

$$\frac{\mathcal{Q}_k(s_k)}{\|s_k\|} \geq \frac{\mathcal{Q}_k(d_k)}{\|d_k\|}, \quad (24)$$

is satisfied. If the Newton direction has not been modified, and the quadratic model is minimized by this direction, it holds that

$$\mathcal{Q}_k(s_k) = \frac{1}{2}g_k^T s_k. \quad (25)$$

If  $d_k$  is replaced in (23) and in condition (24), together with (25), we have the following condition equivalent to (24): Select  $s_k$  whenever

$$\frac{g_k^T s_k}{\|s_k\|} \geq 2\mathcal{Q}_k(d_k) \quad (26)$$

holds for  $d_k$  such that  $\|d_k\| = 1$ .

The inequality (26) provides a criterion to determine which of the two directions in the pair gives the largest potential for descent in the objective function. The proposal in Gould et al. [11] could be generalized to the case when both directions offer a significant potential for descent. If inequality (26) is extended to consider this case, we have the condition

$$\tau_2 \mathcal{Q}_k(d_k) \geq \frac{g_k^T s_k}{\|s_k\|} \geq \tau_1 \mathcal{Q}_k(d_k), \quad (27)$$

where  $\tau_1$  and  $\tau_2$  are prespecified constants satisfying  $0 < \tau_2 < 2 < \tau_1$ . This multiple condition can be used as a criterion to select among the three alternatives introduced at the beginning of this Section, in the following manner:

- If (27) holds, both directions are considered relevant, the descent pair is defined as  $(\bar{s}_k, \bar{d}_k) = (s_k, d_k)$  and the new iterate is obtained from a curvilinear search as:

$$x_{k+1} = x_k + \alpha_k^2 s_k + \alpha_k d_k.$$

- Otherwise, if

$$\frac{g_k^T s_k}{\|s_k\|} < \tau_1 \mathcal{Q}_k(d_k) \quad (28)$$

holds, the algorithm performs a linesearch based only on the Newton direction, the descent pair is defined as  $(\bar{s}_k, \bar{d}_k) = (s_k, 0)$  and the next iterate is obtained from

$$x_{k+1} = x_k + \alpha_k^2 s_k.$$

- Also, if both conditions

$$\tau_2 \mathcal{Q}_k(d_k) < \frac{g_k^T s_k}{\|s_k\|} \quad \text{and} \quad g_k^T d_k \geq \tau_3 d_k^T H_k d_k \quad (29)$$

hold simultaneously for some positive constant  $\tau_3$ , then the procedure conducts a linesearch that considers only the direction of negative curvature, the

descent pair is defined as  $(\bar{s}_k, \bar{d}_k) = (0, d_k)$  and it computes the next iterate as

$$x_{k+1} = x_k + \alpha_k d_k.$$

- Finally, if both conditions

$$\tau_2 \mathcal{Q}_k(d_k) < \frac{g_k^T s_k}{\|s_k\|} \quad \text{and} \quad g_k^T d_k < \tau_3 d_k^T H_k d_k \quad (30)$$

hold simultaneously, then the procedure conducts a search as in the first case, considering both directions as relevant, defining the descent pair as  $(\bar{s}_k, \bar{d}_k) = (s_k, d_k)$  and computing the new iterate from a curvilinear search as:

$$x_{k+1} = x_k + \alpha_k^2 s_k + \alpha_k d_k.$$

The following result establishes that the preceding rules to select the directions in the search satisfy condition  $C3$ , and as a consequence that the algorithm is globally convergent.

*Lemma 2:* Assume that in each iteration  $k$  the iterates  $\{x_k\}$  remain in the set  $\mathcal{L}(x_0)$ , assumptions  $A1$  and  $A2$  hold and we are able to compute a descent pair  $(s_k, d_k)$  satisfying conditions (2), (3), (4) and  $\|d_k\| = 1$ . Then condition  $C3$  holds for the descent pair  $(\bar{s}_k, \bar{d}_k)$  defined as indicated in the preceding paragraphs.

*Proof:* Note that under assumptions  $A1$  and  $A2$  and the condition that the iterates remain in  $\mathcal{L}(x_0)$  there exists a constant  $c_4$  such that  $\|g_k\| \leq c_4$  and from conditions (2), (3) and (4) we have

$$g_k^T s_k \leq 0, \quad \|s_k\| \leq c_2 \|g_k\|, \quad g_k^T d_k \leq 0, \quad \|d_k\| \leq c_3. \quad (31)$$

As a consequence it is enough that we study condition (20).

If (27) holds, the desired result follows from  $\bar{s}_k = s_k$ ,  $\bar{d}_k = d_k$  and conditions (2) and (4) implying

$$\begin{aligned} \bar{s}_k^T g_k &\leq -c_1 \|g_k\|^2 \quad \text{and} \quad \bar{d}_k^T H_k \bar{d}_k \leq \theta \lambda_{H_k} \\ \Rightarrow \min(\bar{s}_k^T g_k / \|g_k\|, \bar{d}_k^T H_k \bar{d}_k) &\leq \min(-c_1 \|g_k\|, \theta \lambda_{H_k}). \end{aligned}$$

If (28) holds, then  $\bar{s}_k = s_k$  and  $\bar{d}_k = 0$ , implying  $\min(\bar{s}_k^T g_k / \|g_k\|, \bar{d}_k^T H_k \bar{d}_k) = s_k^T g_k / \|g_k\|$ . From (28), (31), (4) and  $\tau_1 > 2$ ,

$$\begin{aligned} \frac{s_k^T g_k}{\|s_k\|} &\leq \tau_1 (g_k^T d_k + \frac{1}{2} d_k^T H_k d_k) \\ &\leq \tau_1 \frac{1}{2} d_k^T H_k d_k \leq d_k^T H_k d_k \leq \theta \lambda_{H_k}. \end{aligned} \quad (32)$$

But (2) implies  $-\|s_k\| \|g_k\| \leq s_k^T g_k \leq -c_1 \|g_k\|^2 \Rightarrow \|s_k\| \geq c_1 \|g_k\|$  and using this bound in (32) we obtain

$$\frac{s_k^T g_k}{\|g_k\|} \leq c_1 \frac{s_k^T g_k}{\|s_k\|} \leq c_1 \theta \lambda_{H_k}.$$

From this inequality and  $s_k^T g_k \leq -c_1 \|g_k\|^2$  we have that

$$\begin{aligned} \min(\bar{s}_k^T g_k / \|g_k\|, \bar{d}_k^T H_k \bar{d}_k) &= s_k^T g_k / \|g_k\| \\ &\leq \min(-c_1 \|g_k\|, c_1 \theta \lambda_{H_k}). \end{aligned}$$

Consider now the case when condition (29) holds. Now we have  $\bar{s}_k = 0$  and  $\bar{d}_k = d_k$ , implying  $\min(\bar{s}_k^T g_k / \|g_k\|, \bar{d}_k^T H_k \bar{d}_k) = d_k^T H_k d_k$ . From (29) and (3) we have that

$$\begin{aligned} \frac{s_k^T g_k}{\|g_k\|} &\geq c_2 \frac{s_k^T g_k}{\|s_k\|} \geq c_2 \tau_2 (g_k^T d_k + \frac{1}{2} d_k^T H_k d_k) \\ &\geq c_2 \tau_2 (\tau_3 + \frac{1}{2}) d_k^T H_k d_k. \end{aligned}$$

From (2) we then have

$$d_k^T H_k d_k \leq -\frac{c_1}{c_2 \tau_2 (\tau_3 + \frac{1}{2})} \|g_k\|,$$

and using (4) and the preceding results we obtain

$$\begin{aligned} \min(\bar{s}_k^T g_k / \|g_k\|, \bar{d}_k^T H_k \bar{d}_k) &= d_k^T H_k d_k \\ &\leq \min\left(-\frac{c_1}{c_2 \tau_2 (\tau_3 + \frac{1}{2})} \|g_k\|, \theta \lambda_{H_k}\right). \end{aligned}$$

Finally, if (30) holds, then using the same arguments as in the first case

$$\begin{aligned} \bar{s}_k^T g_k &\leq -c_1 \|g_k\|^2 \text{ and } \bar{d}_k^T H_k \bar{d}_k \leq \theta \lambda_{H_k} \\ \Rightarrow \min(\bar{s}_k^T g_k / \|g_k\|, \bar{d}_k^T H_k \bar{d}_k) &\leq \min(-c_1 \|g_k\|, \theta \lambda_{H_k}). \end{aligned}$$

For the classifying condition (27), determining the type of search to carry out in each iteration, we have taken  $\tau_1 = 10$  and  $\tau_2 = 0.05$ .

The termination condition for the algorithm has been derived from the optimality conditions for problem (1). The procedure terminates whenever

$$\|g(x_k)\| \leq \varepsilon(1 + \|g(x_0)\|),$$

holds, where  $\varepsilon = 10^{-5}$ .

The proposed algorithm follows the scheme presented below:

#### Adapted Curvilinear Search (ACS) Algorithm

Select an initial iterate  $x_0$

Let  $k = 0$

**Repeat**

    Compute  $s_k$  as an approximate solution for the system (7)

    Compute, if it exists, a direction of negative curvature  $d_k$  from the modified Cholesky factorization

    Let  $d_k = 0$  if (33) is not satisfied

    Select the search model from condition (27)

    Compute  $\alpha_k$  to satisfy condition C2

    Update the variables according to the search model used:

        If (27) holds,  $x_{k+1} = x_k + \alpha_k^2 s_k + \alpha_k d_k$

        If (28) holds,  $x_{k+1} = x_k + \alpha_k^2 s_k$

        If (29) holds,  $x_{k+1} = x_k + \alpha_k d_k$

    Let  $k = k + 1$

**until** convergence

## V. ADAPTED CURVILINEAR SEARCH (ACS) ALGORITHM

The implementation of the proposed algorithm involves several decisions concerning practical details; we describe some of them in the following paragraphs.

- In those iterations where the negative curvature direction is used, it is computationally efficient to scale this direction using the information from the Newton direction, see [18].
- The direction of negative curvature is computed from the modified Cholesky factorization. But in some cases, even if the factorization detects the presence of negative curvature in the Hessian matrix, it is not efficient to use it in the search. We consider a negative curvature direction  $d_k$  to be useful if it holds that

$$d_k^T H(x_k) d_k \leq -\varepsilon_d. \quad (33)$$

In our implementation we have taken  $\varepsilon_d = 10^{-7}$ . If this condition is not satisfied in a given iteration, then we set  $d_k = 0$ .

## VI. NUMERICAL RESULTS

We have conducted a certain number of numerical experiments using the proposed **ACS** algorithm on a set of test problems. Both the algorithm and the test problems have been implemented and executed using MATLAB 6.5 under Linux.

### A. The test problems

We have selected the test problems from the collection CUTER (Constrained and Unconstrained Testing Environment, revisited) proposed by Gould et al. [12], and itself an expansion of the original CUTE collection [3]. This set is considered to be a referent for the verification and comparison of nonlinear programming algorithms as the one proposed in this paper. The problems used in the numerical experiments that we have conducted have been chosen to satisfy the following criteria: they must be nonlinear unconstrained problems of dimension between 1 and 500, and having continuous second derivatives available. A total of 70 problems in CUTER satisfy these conditions.

## B. Analysis of the results

Our main motivation for this work has been to analyze the practical impact of using second order information, as well as the study of efficient implementations of algorithms that make use of this information. As a consequence, we have defined the computational experiments to compare the results obtained on the test set using the proposed algorithm to those of three alternative procedures.

The first algorithm, **MN**, is based on a modified Newton method and does not use any second-order information. The second algorithm, **ALS**, carries out an adapted linesearch similar to the one described in Gould et al. [11], where in each iteration the best direction in the descent pair is selected from condition (26), and used in a standard linesearch framework. The third algorithm, **CS**, is based on the proposal by Moré and Sorensen [20]. Finally, a fourth algorithm, **ACS**, uses the adapted curvilinear search described in the preceding sections. In all cases, the initial points used have been those specified as default ones in the CUTEr environment.

Among the 70 problems in the set, the modified Cholesky factorization detected the presence of significant negative curvature in 36 instances. The comparative study centers on these 36 problems, as for the remaining cases the four algorithms provide basically equivalent results. In 9 out of the 36 problems (DECONVU, DJTL, HEART6LS, HIMMELBF, HYDC20LS, MARATOSB, PFIT2LS, PFIT3LS and PFIT3LS) none of the algorithms were able to reach convergence in less than 300 iterations, and these problems were removed from the study.

Tables I, II, III and IV show the results obtained by the four algorithms on the 27 remaining problems. The modified Newton method, **MN**, was unable to solve problem OSBORNEB, while the adapted search algorithm **ALS** failed to solve problem HEART8LS. All algorithms were stopped whenever the iteration count exceeded 250 and the number of function evaluations exceeded 1250. In the tables we have used the following labels:

- *pnam*: Problem name.
- *obj*: Objective function value at the computed solution.
- *KKT*: Norm of KKT conditions at the solution.
- *iter*: Iteration count.
- *fgeval*: Number of function and gradient evaluations.
- *nc*: Number of iterations in which negative curvature is detected.
- *und*: Number of iterations in which only the Newton direction is used.

- *unc*: Number of iterations in which only the negative curvature direction is used.
- *ucs*: Number of iterations in which a curvilinear search, based on both directions, is used.

Problem BARD was the only one in which, though negative curvature was detected using the modified Cholesky factorization, none of the methods made use of it. Also, the adapted linesearch algorithm **ALS** detects negative curvature in one of the iterations for problem BOX3, but it is not used there. It is interesting to note the small number of problems where the norm of the KKT conditions is larger than  $10^{-8}$ : one problem for **MN**, two for **ALS**, two for **CS** and three for **ACS**.

We study first the correlation between the number of curvilinear searches and the performance of the algorithm. Figure 1 displays in the *x*-axis the number of curvilinear searches used within each problem, while the *y*-axis represents the average decrease in the number of iterations when using the **ACS** algorithm instead of the **MN** algorithm for the preceding problems. In all cases the average reduction is negative, that is, the algorithm that uses negative curvature information is more efficient than the modified Newton algorithm. It is interesting to note that an increase in the number of curvilinear searches conducted within the algorithm tends to be associated with a larger reduction in the total number of iterations. For example, the average decrease in the iteration count when the curvilinear search is used three or more times is always larger than 19. At least for the test set used in this experiment, this result seems to imply that the benefits of using negative curvature are not associated to a particular iteration, but rather have an impact in all occasions in which they are used.

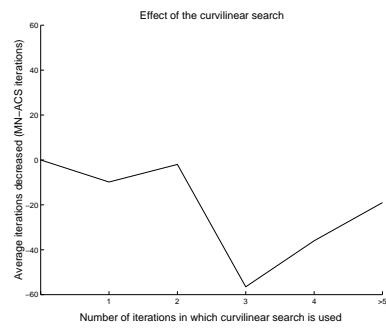


Fig. 1. Correlation between number of curvilinear searches and performance of the algorithm

From Table V, the worst-case performance of the adapted curvilinear search algorithm **ACS**, when compared with the modified Newton algorithm **MN**, corresponds to problem HATFLDE, where the number of iterations required for convergence is increased by 3

(17,65%), while the largest improvement corresponds to problem HUMPS with a decrease in the number of iterations of 36 (47%). The worst case for the **ACS** algorithm when compared to **ALS** implies an increase of 56 iterations (40,57%) in problem PFIT1LS, while the largest improvement corresponds to a decrease of 72 iterations (64,29%) in problem HUMPS. Regarding the curvilinear search algorithm **CS**, the worst performances for **ACS** correspond to problems HAIRY and HIMMELBB, where iteration counts are increased by 4 and 2 (20%) respectively, while the largest improvement, 17 iterations (54,84%), corresponds to problem GULF.

Tables VI and VII present a summary of iteration and function evaluation counts for each one of the algorithms, both including and excluding those problems were the algorithms may have failed. From these tables it is interesting to note that the lowest average number of iterations corresponds to the proposed algorithm **BCA**; the lower numbers of function evaluations for algorithms **MN** and **ALS** is explained by the complexity of a search that uses two directions. This effect can be reduced by using a specialized quadratic search procedure, see for example [9]. Nevertheless, the number of function evaluations for **BCA** is lower than the one for **CS**.

Similar conclusions can be reached from Table VIII. This table provides the proportion of cases in which each algorithm has been the most efficient one (including ties), both regarding iteration and function evaluation counts.

Finally, Tables IX, X and XI show a comparison of the number of cases in which there was an improvement, a worsening or no change regarding iteration counts when comparing the proposed algorithm **ACS** to the other methods. In all cases, the number of improved cases associated to the **ACS** algorithm is significant.

## VII. CONCLUSIONS

We have described an efficient procedure that uses directions of negative curvature for the solution of nonconvex unconstrained problems, selecting the most appropriate search to obtain local solutions for these problems. The algorithm is based on a modified Newton model to compute the search directions, and a new proposal of an adapted curvilinear search to combine these directions. Particular attention has been paid regarding the conditions to impose on directions of negative curvature before they are considered as a part of the search process.

The computational results illustrate the impact of an appropriate use of negative curvature information: in approximately 50% of the problems negative curvature information was used. Due to the low computational cost to obtain these directions, it seems highly advisable

to integrate them within algorithms for unconstrained optimization. Furthermore, the results seem to show that using an adaptive search model increases the efficiency of Newton based algorithms.

An important issue raised by the results in this work is to determine the impact of the quality of the negative curvature direction on the computational efficiency of the overall algorithm, that is, if an improvement of the directions based on iterative methods, for example, would yield even better computational results for the algorithm.

Another interesting and promising task would be to verify the performance of the proposed approach within numerical procedures based on the use of approximate directions such as conjugate gradient or quasi-Newton methods.

## VIII. ACKNOWLEDGEMENTS

This work has been partially financed by the Spanish grants MCYT TIC2003-05892-C05-05 and URJC PPR-2004-05. Special thanks must be given to Julio Holgado and Dominique Orban for their comments and clarifications regarding the installation of CUTER.

## REFERENCES

- [1] L. Armijo, *Minimization of functions having Lipschitz continuos first parcial derivatives*, Pacific Journal of Mathematics 16 (1966) 1-3
- [2] E.G. Boman, *Infeasibility and negative curvature in optimization*, Ph. D. Thesis, Stanford University (1999)
- [3] I. Bongartz, A.R. Conn, N.I.M. Gould and Ph.L. Toint, *CUTE: Constrained and unconstrained testing environment*, Trans. ACM Math. Software 21, (1995)123-160
- [4] J.R. Bunch and B.N. Parlett, *Direct methods for solving symmetric indefinite systems of linear equations*, SIAM Journal on Numerical Analysis, 8 (1971) 639-655
- [5] D.K. Faddeev and V.N. Faddeeva, *Computational Methods for Linear Algebra*, W.H. Freeman and Co. (1963)
- [6] A.V. Fiacco and G.P. McCormick, *Nonlinear programming: sequential unconstrained minimization techniques*, Wiley, New York (1968)
- [7] R. Fletcher and T.L. Freeman, *A modified Newton method for minimization*, Journal of Optimization Theory and Applications 23 (1977) 357-372
- [8] P.E. Gill and W. Murray, *Newton type methods for unconstrained and linearly constrained optimization*, Mathematical Programming 7 (1974) 311-350
- [9] P.E. Gill, W. Murray and M.H. Wright, *Practical Optimization*, Academic Press, London/New York (1981)
- [10] D. Goldfarb, *Curvilinear path steplength algorithms for minimization which use directions of negative curvature*, Mathematical Programming, 18 (1980), 31-40.
- [11] N.I.M. Gould, S. Lucidi, M. Roma and Ph.D.L. Toint, *Exploiting negative curvature directions in linesearch methods for unconstrained optimization*, Optimization Methods and Software, 14 (2000), 75-98
- [12] N.I.M. Gould, D. Orban and Ph.L. Toint, *CUTER (and SifDec), A Constrained and Unconstrained Testing Environment, revisited*, Technical Report TR/PA (2004)

- [13] L.V. Kantorovich and G.P. Akilov, *Functional Analysis*, Pergamon Press (1964)
- [14] W. Kahan, *Numerical linear algebra*, Canad. Math. Bull 9 (1966) 757-801
- [15] A.V. Knyazev and A.L. Skorokhodov, *On exact estimates of the convergence rate of steepest ascent method in symmetric eigenvalue problem*, LAA 154 (1991) 245-257
- [16] S. Lucidi, F. Rochetich and M. Roma, *Curvilinear stabilization techniques for truncated Newton methods in large scale unconstrained optimization*, SIAM Journal on Numerical Analysis, 8 (1998) 916-939
- [17] G. McCormick, *A modification of Armijo's step-size rule for negative curvature*, Mathematical Programming 13 (1977) 111-115
- [18] J.M. Moguerza and F.J. Prieto, *An Augmented-Lagrangian interior-point method using directions of negative curvature*, Mathematical Programming 95 (2003) 573-616
- [19] J.M. Moguerza and F.J. Prieto, *Combining search directions using gradient flows*, Mathematical Programming 96 (2003) 529-559
- [20] J. J Moré and D.C. Sorensen , *On the use of directions of negative curvature in a modified Newton method*, Mathematical Programming 16 (1979) 1-20
- [21] H. Mukai and E. Polak, *A second order algorithm for the general nonlinear programming problem*, Journal of Optimization Theory and Applications, 4 vol. 26 (1978)
- [22] J.M. Ortega and W.C. Rheinboldt, *Iterative solution of nonlinear equations in several variables*, Academos Press, New York (1970)
- [23] S. San Matías and M. Roma, *Un método de búsqueda lineal con direcciones combinadas para optimización irrestringida*, Actas del XXVI Congreso Nacional de Estadística e Investigación Operativa, Úbeda, Spain (2001)

APPENDIX  
TABLES

<b>pnam</b>	<b>obj</b>	<b>KKT</b>	<b>iter</b>	<b>fgeval</b>
ALLINITU	5.7443849103	2.99e-15	8	10
BARD	0.0082148773	3.43e-10	9	9
BEALE	6.08e-26	9.14e-13	9	63
BIGGS6	3.75e-17	1.37e-11	42	69
BOX3	5.40e-19	2.51e-11	8	8
DENSCHND	9.10e-14	8.46e-09	48	70
DENSCHNE	4.79e-18	4.38e-09	11	14
ENGVAL2	1.87e-21	2.98e-09	17	23
EXPFIT	0.2405105939	1.21e-10	8	34
GROWTHLS	1.0040405841	1.75e-11	20	32
GULF	2.75e-19	5.12e-09	22	39
HAIRY	20.00	1.67e-10	25	95
HATFLDD	6.62e-08	1.08e-09	22	38
HATFLDE	5.12e-07	4.76e-14	14	17
HEART8LS	7.87e-20	1.05e-09	241	664
HELIX	2.34e-24	2.44e-11	16	21
HIELOW	874.1654321149	1.54e-10	8	10
HIMMELBB	1.95e-30	1.64e-13	14	18
HUMPS	1.48e-19	1.72e-10	76	488
KOWOSB	3.0780094673e-04	1.56e-12	10	12
LOGHAIRY	0.1823215568	9.42e-09	96	720
OSBORNEA	5.46e-05	4.67e-14	18	25
OSBORNEB	-	-	> 250	> 1250
PFIT1LS	2.48e-22	7.30e-11	189	267
SNAIL	2.29e-32	3.03e-16	104	215
STRATEC	2212.2622909	2.24e-06	17	24
YFITU	6.67e-13	4.99e-12	36	46

TABLE I  
DETAILED RESULTS FOR THE MN ALGORITHM

	<b>MN</b>	<b>ALS</b>	<b>CS</b>	<b>ACS</b>
<b>iter</b>	33.88	30.96	33.48	28.52
<b>fgeval</b>	96.71	96.88	130.54	110.46

TABLE VI  
AVERAGE NUMBERS OF ITERATIONS AND FUNCTION  
EVALUATIONS OVER THE TEST PROBLEMS, EXCLUDING THOSE  
PROBLEMS WHERE AN ALGORITHM MAY HAVE FAILED

<b>pnam</b>	<b>obj</b>	<b>KKT</b>	<b>iter</b>	<b>fgeval</b>	<b>nc</b>	<b>und</b>	<b>unc</b>
ALLINITU	5.7443849103	4.60e-09	8	8	2	6	2
BARD	0.0082148773	3.40e-10	9	9	3	9	0
BEALE	3.04e-23	5.40e-11	10	42	1	9	1
BIGGS6	1.36e-16	2.20e-09	39	87	20	28	11
BOX3	5.40e-19	2.50e-11	8	8	1	8	0
DENSCHND	7.96e-14	7.80e-09	48	49	7	42	6
DENSCHNE	8.17e-21	1.80e-10	12	12	8	9	3
ENGVAL2	1.85e-20	9.40e-09	14	14	1	13	1
EXPFIT	0.2405105939	2.40e-14	11	17	4	7	4
GROWTHLS	1.0040405841	5.10e-10	24	58	7	18	6
GULF	2.87e-25	6.30e-12	26	41	5	22	4
HAIRY	20.00	1.40e-08	24	125	12	12	12
HATFLDD	6.62e-08	3.90e-10	23	28	1	22	1
HATFLDE	5.12e-07	4.40e-10	17	26	4	14	3
HEART8LS	-	-	> 250	> 1250	-	-	-
HELIX	2.29e-21	9.40e-10	14	14	8	8	6
HIELOW	874.1654321149	9.70e-07	9	13	3	7	2
HIMMELBB	1.96e-23	4.30e-10	12	12	10	11	1
HUMPS	7.09e-18	1.20e-09	112	1010	75	37	75
KOWOSB	3.08e-04	1.10e-13	10	14	3	9	1
LOGHAIRY	0.1823215568	2.80e-10	84	300	67	19	65
OSBORNEA	5.46e-05	2.30e-14	22	137	5	20	2
OSBORNEB	0.0875947241	1.00e-12	32	104	15	19	13
PFIT1LS	6.57e-23	3.70e-11	82	98	1	81	1
SNAIL	5.03e-23	1.40e-11	103	185	4	99	4
STRATEC	2212.2622909	4.30e-06	15	18	4	12	3
YFITU	6.67e-13	1.80e-09	38	48	2	37	1

TABLE II  
DETAILED RESULTS FOR THE **ALS** ALGORITHM

	<b>MN</b>	<b>ALS</b>	<b>CS</b>	<b>ACS</b>
<b>iter</b>	49.56	39.11	39.44	34.41
<b>feval</b>	162.88	141.5	153.46	133.31

TABLE VII

AVERAGE NUMBERS OF ITERATIONS AND FUNCTION EVALUATIONS OVER THE TEST PROBLEMS, INCLUDING THOSE PROBLEMS WHERE AN ALGORITHM MAY HAVE FAILED

Number of problems	27
<b>ACS</b>	16
<b>GD</b>	6
Tied	5

TABLE X  
COMPARISON BETWEEN **ACS** AND **ALS**. NUMBER OF PROBLEMS SHOWING BEST PERFORMANCE

	<b>MN</b>	<b>ALS</b>	<b>CS</b>	<b>ACS</b>
<b>% best iter</b>	29.63	33.33	37.04	48.15
<b>% best feval</b>	25.93	48.15	22.22	22.22

TABLE VIII

PERCENTAGE OF PROBLEMS WITH BEST PERFORMANCE

Number of problems	27
<b>ACS</b>	14
<b>MN</b>	7
Tied	6

TABLE IX

COMPARISON BETWEEN **ACS** AND **MN**. NUMBER OF PROBLEMS SHOWING BEST PERFORMANCE

Number of problems	27
<b>ACS</b>	13
<b>CS</b>	6
Tied	8

TABLE XI  
COMPARISON BETWEEN **ACS** AND **CS**. NUMBER OF PROBLEMS SHOWING BEST PERFORMANCE

<b>pnam</b>	<b>obj</b>	<b>KKT</b>	<b>iter</b>	<b>fgeval</b>	<b>nc</b>
ALLINITU	5.7443849103	1.34e-11	9	11	2
BARD	0.0082148773	9.68e-11	11	14	2
BEALE	6.10e-26	9.20e-13	9	88	1
BIGGS6	7.83e-17	1.47e-09	71	112	14
BOX3	8.50e-19	3.45e-10	8	8	1
DENSCHND	7.76e-14	7.69e-09	48	68	5
DENSCHNE	8.17e-21	1.81e-10	9	9	3
ENGVAL2	4.95e-21	4.84e-09	14	16	1
EXPFIT	0.2405105939	3.51e-13	10	40	1
GROWTHLS	1.0040405841	7.81e-10	27	44	5
GULF	2.18e-30	4.74e-15	31	60	7
HAIRY	20.00	7.42e-08	16	80	9
HATFLDD	6.62e-08	7.11e-10	23	30	1
HATFLDE	5.12e-07	4.22e-13	26	44	5
HEART8LS	4.19e-29	3.59e-13	212	818	201
HELIX	2.06e-36	9.52e-18	19	36	5
HIELOW	874.1654321149	3.02e-08	6	9	1
HIMMELBB	1.04e-29	4.78e-13	10	11	7
HUMPS	6.44e-26	1.14e-13	78	1119	55
KOWOSB	3.08e-04	7.79e-14	11	15	3
LOGHAIRY	0.1823215568	1.49e-12	66	797	48
OSBORNEA	5.46e-05	4.58e-13	20	30	4
OSBORNEB	0.0401377363	4.85e-15	16	39	6
PFIT1LS	1.38e-20	4.88e-10	159	222	3
SNAIL	3.16e-22	3.55e-11	106	247	4
STRATEC	2212.2622909	3.78e-11	16	23	3
YFITU	6.67e-13	2.58e-10	34	47	3

TABLE III  
DETAILED RESULTS FOR THE **CS** ALGORITHM

<b>pnam</b>	<b>obj</b>	<b>KKT</b>	<b>iter</b>	<b>fgeval</b>	<b>nc</b>	<b>und</b>	<b>unc</b>	<b>ucs</b>
ALLINITU	5.7443849103	1.30e-11	9	11	2	7	0	2
BARD	0.0082148773	3.40e-10	9	9	3	9	0	0
BEALE	6.08e-26	9.10e-13	9	88	1	8	1	0
BIGGS6	3.75e-17	3.40e-14	43	90	19	25	3	15
BOX3	8.50e-19	3.50e-10	8	8	1	7	0	1
DENSCHND	7.60e-14	7.60e-09	47	70	6	41	4	2
DENSCHNE	8.03e-20	5.70e-10	11	11	6	9	0	2
ENGVAL2	4.93e-28	1.60e-12	14	16	1	13	1	0
EXPFIT	0.2405105939	3.50e-13	10	40	1	9	0	1
GROWTHLS	1.0040405841	6.00e-12	21	99	7	14	4	3
GULF	6.72e-18	1.00e-09	14	27	6	8	4	2
HAIRY	20.00	7.80e-08	20	158	10	12	2	6
HATFLDD	6.62e-08	1.10e-12	22	32	4	19	3	0
HATFLDE	5.12-07	1.00e-12	17	33	4	14	3	0
HEART8LS	5.55e-29	4.30e-13	197	761	183	44	37	116
HELIX	8.14e-38	1.90e-18	18	36	7	12	3	3
HIELOW	874.1654321149	3.00e-08	6	9	1	5	0	1
HIMMELBB	1.10e-30	7.80e-14	12	13	9	10	0	2
HUMPS	1.08e-22	4.60e-12	40	848	22	26	10	4
KOWOSB	3.08e-04	2.80e-09	6	11	2	5	0	1
LOGHAIRY	0.1823215568	3.80e-14	68	557	54	27	10	31
OSBORNEA	5.46e-05	5.90e-13	14	19	3	13	0	1
OSBORNEB	0.0401377363	3.30e-11	19	54	7	4	12	3
PFIT1LS	3.08e-20	5.80e-10	138	195	3	135	2	1
SNAIL	3.16e-22	3.60e-11	106	247	4	102	1	3
STRATEC	2212.2622909	4.50e-06	17	24	3	14	1	2
YFITU	6.67e-13	5.50e-12	34	47	3	31	1	2

TABLE IV  
DETAILED RESULTS FOR THE **ACS** ALGORITHM

<b>pnam</b>	<b>iter</b>				<b>fgeval</b>			
	<b>MN</b>	<b>ALS</b>	<b>CS</b>	<b>ACS</b>	<b>MN</b>	<b>ALS</b>	<b>CS</b>	<b>ACS</b>
ALLINITU	8	8	9	9	10	8	11	11
BARD	9	9	11	9	9	9	14	9
BEALE	9	10	9	9	63	42	88	88
BIGGS6	42	39	71	43	69	87	112	90
BOX3	8	8	8	8	8	8	8	8
DENSCHND	48	48	48	47	70	49	68	70
DENSCHNE	11	12	9	11	14	12	9	11
ENVAL2	17	14	14	14	23	14	16	16
EXPFIT	8	11	10	10	34	17	40	40
GROWTHLS	20	24	27	21	32	58	44	99
GULF	22	26	31	14	39	41	60	27
HAIRY	25	24	16	20	95	125	80	158
HATFLDD	22	23	23	22	38	28	30	32
HATFLDE	14	17	26	17	17	26	44	33
HEART8LS	241	-	212	197	664	-	818	761
HELIX	16	14	19	18	21	14	36	36
HIELOW	8	9	6	6	10	13	9	9
HIMMELBB	14	12	10	12	18	12	11	13
HUMPS	76	112	78	40	488	1010	1119	848
KOWOSB	10	10	11	6	12	14	15	11
LOGHAIRY	96	84	66	68	720	300	797	557
OSBORNEA	18	22	20	14	25	137	30	19
OSBORNEB	-	32	16	19	-	104	39	54
PFIT1LS	189	82	159	138	267	98	222	195
SNAIL	104	103	106	106	215	185	247	247
STRATEC	17	15	16	17	24	18	23	24
YFITU	36	38	34	34	46	48	47	47

TABLE V  
OVERALL COMPARISON OF ITERATION AND FUNCTION EVALUATION COUNTS

# A multi-criteria and fuzzy logic based approach for the relative assessment of the fire hazards of chemical substances and installations

Apostolos N. Paralikas<sup>\*,†,§</sup>, Argyrios I. Lygeros<sup>\*</sup>

<sup>\*</sup> School of Chemical Engineering, National Technical University of Athens,  
15780 Zografos Campus, Athens, Greece.

<sup>†</sup>National Operations & Emergency Response Centre, Greek Fire Brigade,  
1 Rizariou & Mikras Asias str., Halardi, 15233, Athens, Greece.

<sup>§</sup>Corresponding author. Tel.: +30 210 6549911, fax: +30 210 6828382,  
Email: [aparalik@mail.ntua.gr](mailto:aparalik@mail.ntua.gr)

**Abstract—** Hazardous materials classification and ranking is a multi-criteria problem; no single property can be used for assessing the fire hazards of chemical substances or materials. This paper explores the use of a multi-criteria decision-making technique, Analytic Hierarchy Process, in better incorporating the different properties or parameters, in hazardous materials assessment.

Based on this approach, a methodology has been developed and is introduced, for the rapid assessment and relative ranking of the fire hazards of chemical substances. Fuzzy logic was also used, for handling both linguistic variables and uncertainties present in hazard assessment. The proposed approach has been applied in developing two hazard ranking indices: ‘Substance Fire Hazard Index’, , and ‘Consequences Index’, presented here.

Finally some conclusions on the efficiency of using the multi-criteria approach in hazard indices development are also discussed.

**Keywords—** Fuzzy sets; AHP; Hazard assessment; ranking indices; Multi-criteria decision making.

## I. INTRODUCTION

A major challenge that chemical industry and regulatory authorities are facing nowadays is the comprehensive management of the risks resulting from industrial activities and hazardous materials use, production, transportation or disposal.

First step in risk management is hazards identification and assessment, which requires systematic and methodical study and analysis of the hazards. For this purpose, a wide range of hazard identification and assessment techniques has been developed; among them are relative assessment and ranking techniques [1, 2, 3]. The main advantage of this kind of techniques is that they do not require the commitment of many resources, in manpower and time. The most known and widespread among this category of techniques are Dow’s ‘Fire & Explosion Index’

[4] and ICI ‘Mond’ index [5], aiming at the rapid hazard assessment at installations that use hazardous substances. Other indices have also been proposed for ranking toxic [6, 7], ecotoxic [8] and reactive substances [9]. Recent developments in this field include, among other, a range of indices and similar tools [10-15]. Tiixtier *et al* [16] have recently presented a collection and comparative analysis of 62 hazard and risk analysis methodologies, including indices, relative assessment and ranking techniques.

## II. A MULTI-CRITERIA PROBLEM

Is it widely acknowledged, both in chemical process safety and fire safety literature, that there is no single fire hazard property or parameter that could be used for the assessment of the fire hazards of materials or substances. More specifically, it has been stated that: ‘No single fire hazard property, such as flash point or ignition temperature, should be used to describe or appraise the fire hazard or fire risk of a material, product, assembly, or system under actual fire conditions’ [17]. Moreover, ‘there is no single parameter which defines flammability, but some which are relevant are: a) flash point, b) flammability limits, c) auto-ignition temperature, d) ignition energy, and e) burning velocity’ [18]. ‘The fire hazard properties may be used as elements of a fire risk assessment only when such assessment takes into account all of the factors that are pertinent to the evaluation of the fire hazard of a given situation.’ [17]

There is a wide variety of fire hazard properties that could be taken into account in such an assessment. Different methods, tools, codes, legislation requirements, guidelines, etc, [19, 20], use varying sets of properties to access the fire hazards of chemical substances. Such properties included, for example, are the parameters outlined above [18], or those described in ref. [17],

namely: flash point, ignition temperature, flammable (explosive) limits, specific gravity (relative density), vapour density, boiling point, melting point, water solubility.

The scope of this work is to develop a new methodology for developing safety-related indices, aimed at the relative ranking and comparative assessment of hazardous substances, installations, units, or processes. The proposed methodology handles the issue of relative assessment and ranking as a multi-criteria decision-making problem; therefore aims in incorporating the different decision criteria in the assessment. For this purpose, a multi-criteria analysis technique, Analytic Hierarchy Process (AHP) [21], has been employed. Furthermore, the development of the proposed methodology was also based on fuzzy logic concepts.

Based on the proposed methodology two indices, the ‘Substance Fire Hazard Index’, *SFHI*, and the ‘Consequences Index’, *CI*, have been developed and are presented, in order to demonstrate the application of the methodology. *SFHI* is proposed as a tool for the relative ranking and comparative assessment of hazardous substances, according to their fire hazard properties. It is focused on estimating the fire hazards of the substances related to accidents that could take place at installations that use, process, produce, or store hazardous substances. The calculation of the proposed index is based on a total of 16 hazardous properties. The ‘Consequences Index’, *CI*, is introduced as a tool for the ranking of industrial facilities, units or processes that use hazardous substances, according to the magnitude of the possible consequences at the installation, as well as the natural and human environment around it, from a possible accident that could take place.

The proposed methodology could also be used for the development of similar indices, based on any organization’s need and views. Also the proposed indices could be modified by any user to include their own priorities, or decision environment. It should be emphasized that the principal aim of this work is to present a new methodological approach in dealing with chemical accidents hazards, namely the multi-criteria approach, not to introduce a new index or two. Each body, institution, authority, etc using or adopting this approach could in the first place adapt the proposed methodology and indices in a way to better suit its needs, or the decision environment within it is working.

This paper is organized as follows: in section III principles of fuzzy sets theory are presented; in section IV Analytic Hierarchy Process is briefly described; in section V the proposed methodology is introduced. In Sections VI and VII the developed indices are outlined. Following in Section VIII, a demonstration example of the proposed ‘Consequences Index’, *CI*, is presented. Finally in Section IX, some conclusions are presented.

### III. FUZZY SETS CONCEPTS

Fuzzy Sets theory was introduced by L.Zadeh in 1965 to deal with imprecision, uncertainty and vagueness that are inherent in many ‘real world’ problems [22]. There have been many successful applications of fuzzy sets and logic since, including chemical process safety related issues.

Central point in fuzzy sets theory is the notion of membership. In classic sets, an element may or may not belong to a given set. In fuzzy sets, an element may belong to a set up to some degree, the membership degree, which takes values between 0 and 1. Fuzzy numbers are fuzzy sets with membership functions represented mathematically, allowing the performance of arithmetic operations. Common shapes, used in this work, are triangular and trapezoid fuzzy numbers.

Fuzzy numbers can be used effectively to describe Linguistic Variables, such as ‘very tall’, ‘tall’; and furthermore, for handling qualitative data (e.g. ‘slightly soluble’, etc.). In existing hazard classification systems, levels, classes or categories of the various hazardous properties of chemical substances are usually determined with the use of intervals defined by ‘crisp’ boundaries. For example, according to NFPA 704 [23], a substance (e.g. *gasoline*) with flash point -43°C has Fire Hazard Rating,  $N_F=4$ . The same rating applies to a substance with flash point 13 °C (e.g. *ethanol*), while for a substance with flash point 32 °C (e.g. *xylene*),  $N_F=3$  and *kerosene*, with flash point 37,8 °C, has  $N_F=2$ .

As it will be described later on, fuzzy linguistic variables have been used in the development of the proposed index, for describing the various levels, classes or categories of each hazardous property, and also in the development of utility functions for assigning penalty values to each hazardous property.

### IV. AHP: AN OUTLINE

The use of decision-making tools, among

them Analytic Hierarchy Process (AHP), has been suggested in literature [24-25] for the assessment of chemical safety related issues. AHP, developed by T.L. Saaty in late 70s [21], is designed to deal with complex decision-making problems involving multiple criteria, in a wide range of application fields [26]. It can be used for ranking decision alternatives, using a set of parameters that are taken into account in the assessment. The assessment and ranking of the fire hazards of chemical substances, or facilities that use produce or store hazardous substances, process units, etc, can be viewed as such a complex problem, with multiple parameters involved; various substances, facilities, processes or units can be viewed as decision alternatives. AHP allows for intangible and quantitative factors to be successfully involved in the assessment process. It has already been employed in fire safety assessment of buildings and structures [27-30], as well as in safety related issues [31-32]. The use of the multi-criteria approach in chemical risks management is a relatively new application field [33-34], although J.J. Buckley [35] had used the example of chemicals ranking in introducing Fuzzy Hierarchical Analysis, the fuzzy sets extension of AHP.

As it has been mentioned already, no single fire hazard property can be used to describe or appraise the fire hazards or risks of materials; therefore AHP is a suitable multi-criteria method for incorporating different parameters in the assessment. Two major steps can be distinguished in the procedure: First, structuring the problem under consideration in a hierarchical form. This involves ‘decomposition’ of the problem into components, namely the identification of the parameters that are considered relevant, as well as the organization of these parameters in groups and subgroups, which are then linked in a hierarchical manner. This results in forming the Hierarchy, the hierarchical structure that is representative of the analysis of the specific problem. The lowest level of the Hierarchy is consisted by the alternatives, or, in the ‘ratings’ mode, by the parameters that are employed in the assessment. This mode is suitable in cases of large number of alternatives or for the development of a general index. In this case, the assessment is performed in a spreadsheet manner, where a Weight Factor is assigned to each parameter and for the different alternatives relevant Penalty Factors representing the magnitude of each parameter are assigned.

Second step is the assignment of weights to each parameter of the problem. This is done

through pair-wise comparative judgments among all parameters that belong to the same group or sub-group of the decision hierarchy. The parameters are compared in respect to their importance, likelihood or preference, depending on the nature of the problem under consideration. The pair-wise comparisons are performed by an expert, or group of experts, capturing their knowledge, expertise or understanding, which are incorporated in the final results. To compare parameter  $i^{th}$  with parameter  $j^{th}$ , the decision-maker assigns a linguistic value  $a_{ij}$ , which corresponds to a numeric value, an integer in the range 1-9. The meaning of each value on the scale is presented in Table I.

TABLE I  
THE PAIR-WISE COMPARISONS SCHEME IN AHP

$a_{ij}=1$	The two parameters are <b>equally important</b> ( <i>likely / preferred, etc...</i> )
3	parameter $i$ is <b>weakly more</b> .... than $j$ .
5	parameter $i$ is <b>strongly more</b> .... than $j$ .
7	parameter $i$ is <b>very strongly more</b> .... than $j$ .
9	parameter $i$ is <b>absolutely more</b> .... than $j$ .
2,4,6,8	interval values between adjacent choices.

Pairwise comparisons of all elements within each group or subgroup form an  $n \times n$  matrix,  $A$ . Rows and columns of the pair-wise comparisons matrix are the  $n$  elements of the respective group or subgroup. The local priorities vector on the group’s or sub-group’s elements is elicited from the eigenvector that corresponds to the maximum eigenvalue of matrix  $A$ . The synthesis of the local priorities of all levels results to the weight vector of the priorities of all parameters taken into account.

A final step, assigning Penalty Factors representing the value of the respective parameter under consideration, is also included when the assessment is performed using the ‘ranking’ mode and not through the pair-wise comparisons among the alternatives. The procedure adopted in the proposed methodology for assigning Penalty Factors is described later on.

## V. THE PROPOSED METHODOLOGY

The development of the proposed methodology for the relative assessment and ranking of hazardous substances and facilities that use, produce or store hazardous substances was based on the above-mentioned approach. The respective steps in the development of the proposed indices are outlined as following:

- a) Determination of the criteria - parameters taken into account in the index calculation.
- b) Assignment of a Weight factor,  $W_j$ , to each criterion / parameter, and
- c) Development of utility functions (or value functions) for the calculation of the Performance measure (or Penalty factor),  $P_j^S$ , attributed to the  $j$  criterion / parameter for the  $S^{th}$  substance/facility.

The combination of the Weights for each parameter and the respective Penalties provides the total value of the relative ranking for the  $S^{th}$  element under consideration:

$$I_S = \sum_j (W_j * P_j^S) \quad (1)$$

where:  $W_j$ : the Weight factor of the  $j$  Parameter, and

$P_j^S$ : the Penalty attributed to the  $j^{th}$  Parameter.

In the following sections the above-mentioned steps are presented in more detail.

## VI. THE ‘SUBSTANCE FIRE HAZARD INDEX’

The proposed ‘Substance Fire Hazard Index’, *SFHI*, is introduced as a tool for the relative ranking and comparative assessment of hazardous substances, according to their fire hazard properties. Its calculation is based on a total of 16 properties, which include fire hazards, physical properties, special hazards and burning properties of chemical substances. *SFHI* is focused on estimating the fire hazards that are related to major accidents that could take place at

installations that use, process, produce, or store hazardous substances.

The stages followed in the development of the proposed index, are presented in more detail:

### A. Fire hazard properties taken into account

First stage in the development of the proposed *SFHI*, was the determination of the substance properties to be taken into account in the development and calculation of the index. For this purpose, all relevant properties related to the behavior of chemical substances under conditions of an accident were recorded. A wide range of relevant sources has been consulted, including chemical substances classification and labelling systems [36]; hazardous chemicals legislation [37, 38]; codes, guides and guidelines [17], [19, 20], [39, 40]; risk evaluation models [41] and fire safety literature [42-46].

The properties to be incorporated in calculation of the proposed Substance Fire Hazard Index, 16 in total, were identified, selected, and then classified in groups and subgroups, as presented in Figure 1.

### B. Determination of Weighting Factors, $W_j$ .

Second stage in the development of the proposed index was the determination of the weights for each parameter/property taken into account in the index calculation. For each Parameter,  $I_j$ , a Weight Factor,  $W_j$ , has to be ascribed. For their determination, Analytic Hierarchy Process has been employed. First step in the implementation of AHP, as it has already been described, is the development of the Hierarchy, the hierarchical structure, which

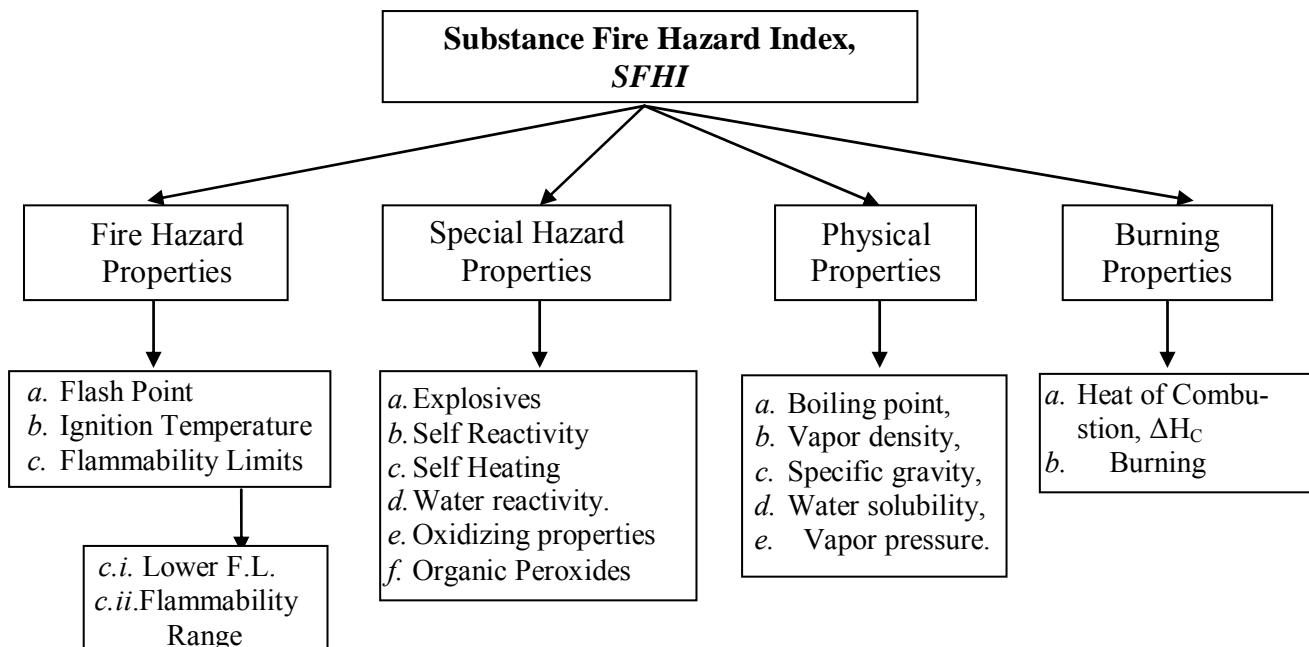


Fig. 1 Hierarchical structure of the Hazard Properties taken into account in SFHI calculation.

represents the problem under consideration. The Hierarchical structure developed and used is shown in Figure 1.

Then the user of the proposed index will have to perform the pair-wise comparisons among all elements of each subgroup, at all levels of the hierarchy, in order to elicit the ‘local’ weights of the parameters, according to his/her own perception and understanding of the problem.

The volume of the pair-wise comparisons produces the ‘local’ weights or priorities, which are then synthesized to produce the final Weights,  $W_j$ , of the parameters. For performing these calculations and also for developing the hierarchy, a software program, ‘Expert Choice 2000’ [47], has been used.

### C. Assignment of penalties, $P_j^S$

Third stage in the development of the proposed index was the elaboration of the procedure for assigning Penalty Factors,  $P_j^S$ , to the parameters (hazardous properties),  $I_j$ , taken into account in the calculation of the index. The development of this procedure was based on fuzzy logic, as well as on AHP. These Penalty Factors are representative of the value  $T_j^S$ , of the respective hazardous property  $I_j$ , for the  $S^{th}$  substance or material under consideration. For the assignment of Penalty Factors a ‘Utility Function’ (or value function) has been developed for each one of the parameters taken into account. The 3-steps procedure for the development of the Utility Functions is outlined as following:

*1) Development of Linguistic Variables for each parameter-property:* The various properties taken into account in the calculation of the proposed index have been described as a set of Linguistic terms, representing the various levels, classes or categories of the hazardous properties of chemical substances. These classes are usually determined by using intervals that are defined by ‘crisp’ boundaries; examples are provided in Table II. The linguistic terms that compose each Linguistic Variable were then represented as triangular or trapezoid fuzzy numbers. In order to develop the Linguistic terms for each Variable, all major classification systems for each property were examined and their levels, classes or categories, as well their crisp boundaries were recorded. Different classification systems may use different number of classes, or these classes may correspond to different values or value intervals; even different definitions of hazardous properties are being used. Based on these recordings, the Linguistic terms of each Variable were determined, and then represented as fuzzy numbers. For example, for the property ‘Flash Point’ (in °C), the following classification is used in two major classification systems, GHS [36] and NFPA 30 [39], as shown in Table II:

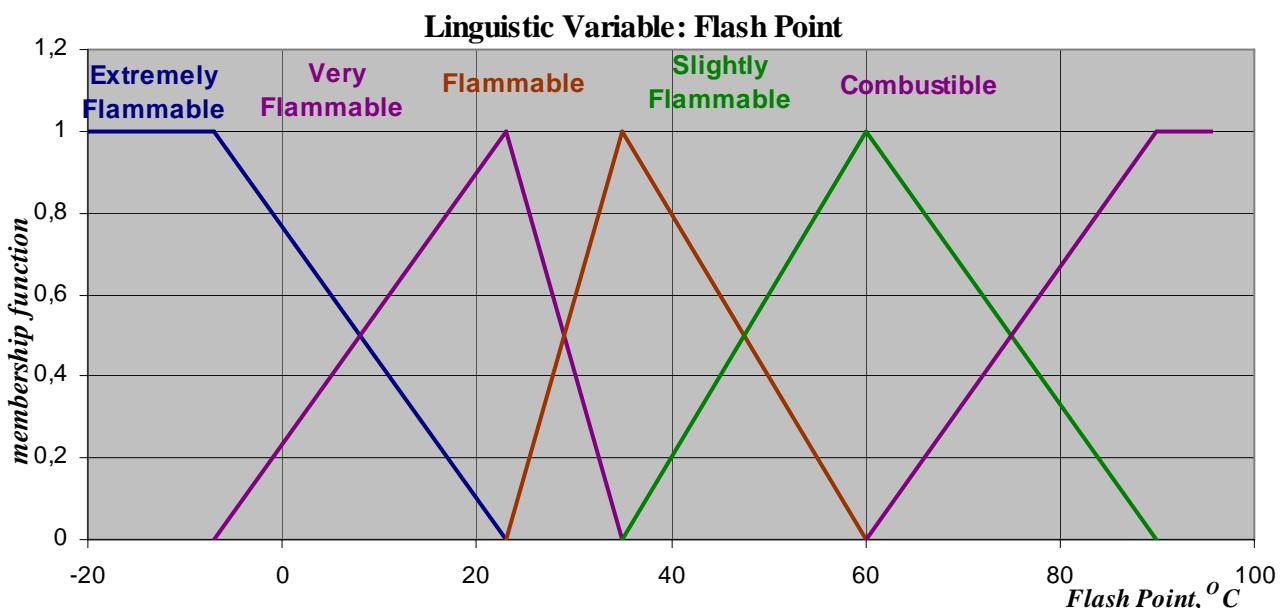


Figure 2: Fuzzy Numbers describing the Linguistic Variable ‘Flash Point’ Categories.

TABLE II  
'FLAMMABILITY' CLASSES IN NFPA 30 AND GHS

<b>GHS</b>	
<u>Class</u>	<u>Limits</u>
1	B.P. = <35 °C.
2	B.P.>35 °C & F.P. <23 °C.
3	B.P>35 °C & 23 °C < F.P.<37,8 °C
4	60 °C < F.P..< 93 °C

<b>NFPA 30</b>		
<u>Classes</u>	<u>Limits</u>	
I A	Flammable	F.P.<22,8 °C & BP<37,8 °C
I B	Liquid	F.P.<22,8 °C & B.P.>37, °C
I C		22,8 °C < F.P. < 37,8 °C
II	Combustible	37,8 °C < F.P. < 60 °C
III A	Liquid	60 °C < F.P. < 93 °C
III B		F.P. > 93 °C

Based on the above-mentioned classification systems, codes, etc, five Linguistic terms describing the Linguistic Value 'Flash Point' were developed for use in the proposed index. These terms where then described as triangular or trapezoid fuzzy numbers,  $fn_k$ , as presented in Table III and pictured in Figure 2.

The physical meaning of these linguistic terms is that a specific Flash Point value of a given substance may belong not only to a sole class, but also to more than one (e.g. 'flammable' and 'very flammable'), with different degrees of membership to each. Introducing the value  $T_j^S$  of the property  $I_j$  for substance  $S$  in Figure 2 produces the membership degree,  $m_{jk}^k$ , of the specific value to each one of the fuzzy linguistic terms,  $k$ . The membership degrees belong to the interval  $[0,1]$  ( $0 \leq m_{jk}^k \leq 1$ ), and are used for the determination of the Penalty Factor that corresponds to the specific value  $T_j^S$ , as it will be

described in the next paragraphs.

2) *Assignment of a weight factor to each Linguistic term:* Next step in the development of the procedure for calculating the Penalty Factors  $P_j^S$ , is the assignment of a weight factor,  $w_k$ , to each linguistic term,  $k$ , of each Linguistic Variable. For this purpose AHP was also employed, through pairwise comparisons among all linguistic terms, based on their relative importance. The weights,  $w_k$ , assigned to the linguistic terms,  $k$ , of the Linguistic Variable 'Flash Point', are presented in Table III.

3) *Calculation of the Penalty Factor:* The Penalty Factor,  $P_j^S$ , is then calculated through the combination of the membership degrees,  $m_{jk}^k$ , of the property value,  $T_j^S$ , for the  $S^{th}$  substance, to each linguistic term, and the weight factors,  $w_k$ , of each linguistic term, using the following relation:

$$P_j^S = \frac{\sum_k (m_{jk}^S * w_k^j)}{\sum_k m_{jk}^S} \quad (2)$$

where  $w_k^j$ : the weight factor of the  $k$  linguistic term

$m_{jk}^S$ : membership degree of the property value,  $T_j^S$ , for the  $S^{th}$  substance,  
 $k$ : the  $k^{th}$  linguistic term of the  $j^{th}$  property.

This procedure, called 'defuzzification', results to the transformation of the membership degrees to a 'normal', or crisp, number. The technique employed is 'center-of-maximum' method, one of the simplest, with minimum complexity in calculations.

After calculating all penalty factors that correspond to each property value,  $T_j^S$ , belonging to the set of values of the given property, the 'Utility Function' curve can be generated.

Penalty factors belong to the interval [0,1]. The Utility Function for the Linguistic Variable ‘Flash Point’ is presented in the Figure 3.

The above-mentioned procedure has been repeated for all the hazardous properties taken into account in the calculation of the proposed index. In cases of hazardous properties that do not have a continuous set of values, but use discrete levels or classes (e.g. reacting with water: ‘violently’, ‘reacting’, ‘slowly’, ‘not reacting’), no Utility Function can be developed. In such cases, a standard Penalty Factor was preassigned to each hazardous property level, using AHP.

#### D. CALCULATION OF THE INDEX

The ‘Substance Fire Hazard Index’ for the  $S$  substance is calculated from the following equation:

$$SFHI_S = \sum_j (W_j * P_j^S) \quad (3)$$

where:  $W_j$ : the Weight factor of the  $j$  Property, and  
 $P_j^S$ : the Penalty factor, attributed to the  $j^{th}$  Property for the  $S^{th}$  substance.

## VII. THE ‘CONSEQUENCES INDEX’

The proposed ‘Consequences Index’ is introduced as a tool for ranking industrial installations, units or processes that use, produce

or store flammable and toxic substances, based on accident consequence analysis. Units and installations are classified according to their inherent ‘Consequences Potential’, defined as the total of the consequences to Human Health, Environment and Property that is possible to be caused by an accident at the installation.

The calculation of the Index is based on the 21 Consequences Categories (CCs) that have been identified. These include Consequence Categories incorporated in other similar indices and ranking tools, as well as legislative requirements and the associated technical guidance documents. For each CC,  $j$ , a Weight Factor,  $W_j$ , has been assigned, using AHP. A Penalty Factor,  $P_{ij}$ , is attributed to each  $CC_j$ , for each  $i^{th}$  installation under consideration, representing the extent of the expected or possible damages from an accident at the installation. The assignment of these values is based either on calculation tools, or on estimation. For the calculation of the Penalty Factors, a ‘Utility Function’ has been developed.

The development of the Consequences Index was based on the same procedure outlined in the previous section. The different parameters that are taken into account on the index calculation, 21 Consequences Categories, were then organized in groups and subgroups, forming the decision Hierarchy, as it is displayed in Figure 4. There are three groups of parameters: Human Health, Economic and Environmental Consequences.

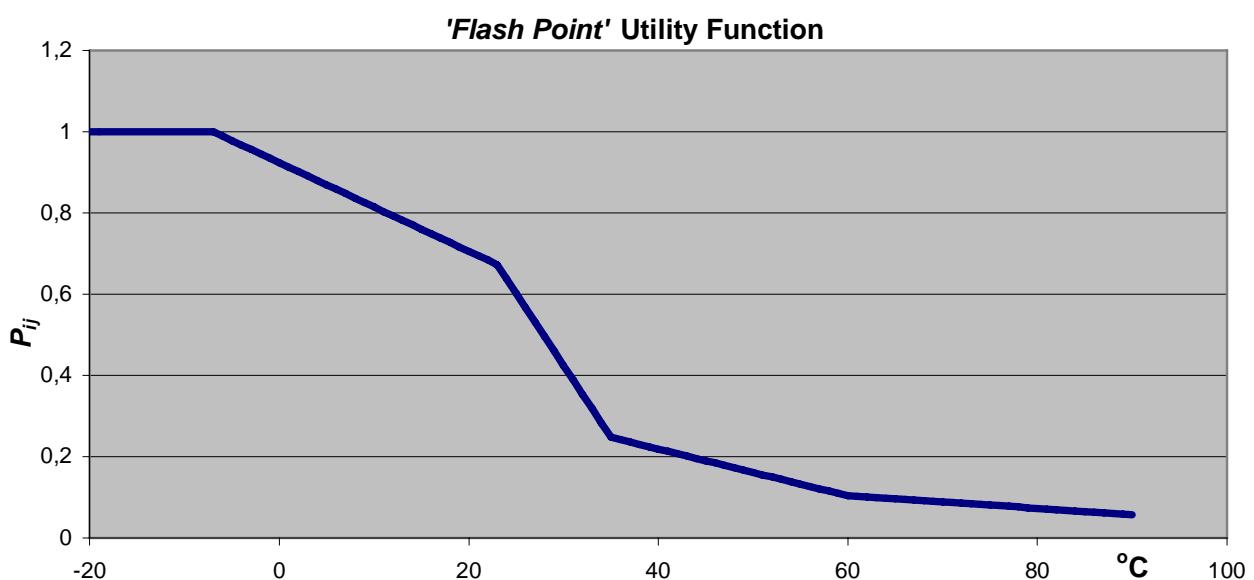


Figure 3: Utility Function for the Linguistic Variable ‘Flash Point’

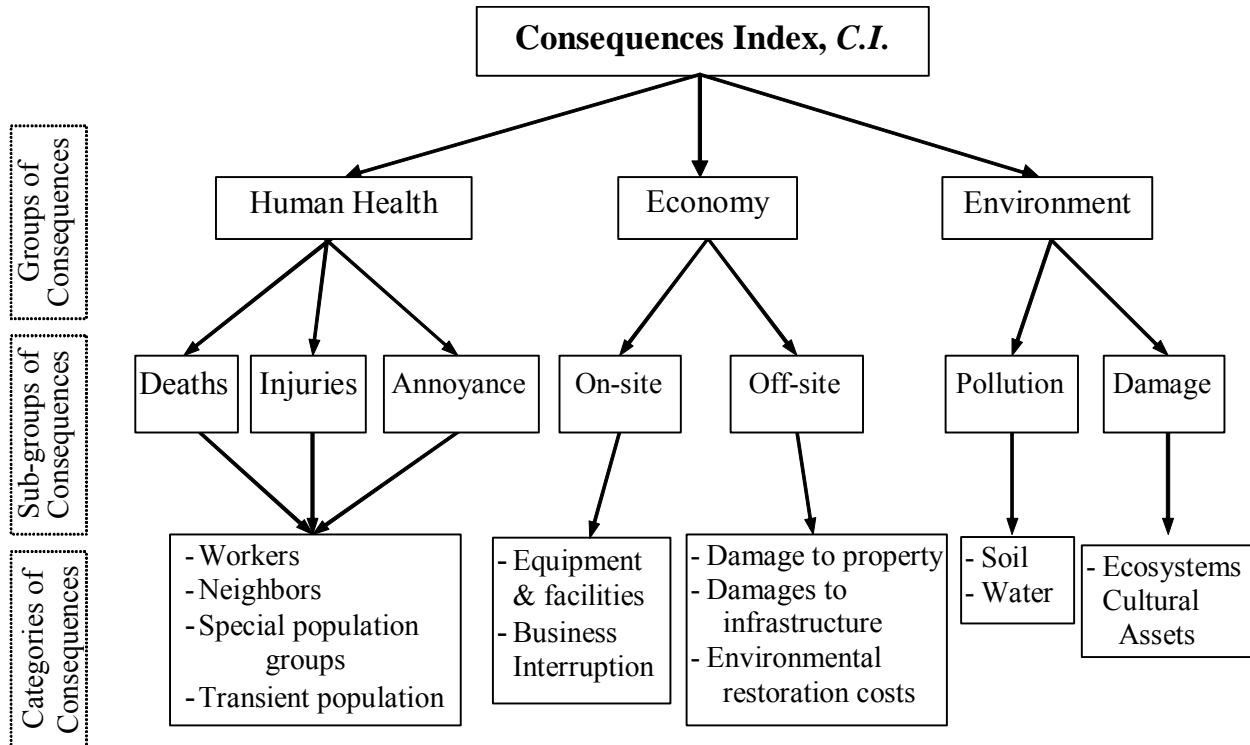


Figure 4. Hierarchical structure of the Consequences Categories.

The first group represents numbers of people, in four different population categories. These numbers are calculated using three different exposure radii, or Hazard Distances, corresponding to different levels of expected Human Health consequences. The calculation is performed by using relevant consequences assessment tools, like the Greek ‘Major Technological Accidents Response Plan’ (SATAME [48]) in Greece, or CATS (Consequences Assessment Tool Set), introduced in Greek Fire Brigade Operations Center within the framework of Athens 2004 Olympics Security Action Plan. Both are using a Geographical Information System (GIS) platform for storing and handling demographic data and other information, e.g. on special installations, vulnerable populations, etc.

Furthermore, a unique Utility Function has been developed for all 21 CCs, since all of them are of the same nature and can be described by the same linguistic terms. The procedure for utility function development is the same as it has been already outlined in the previous section. Relevant industrial accidents scaling and classification schemes have been taken into account.

#### VIII. A DEMONSTRATION EXAMPLE

The proposed C.I. has been applied and tested in the case of a typical storage facility [34]. The tested scenarios involved 4 different toxic chemical substances with varying characteristics and behavior: Ammonia-AM, Chlorine-CL, Ethyleneoxide-EO and Propyleneoxide-PO. Two different types of storage have been considered: pressurized-P and refrigerated-R. Finally, two types of toxic release have been examined for each case: rapture of the sphere and of a 10" pipeline; summing up to a total of 16 accident scenarios. The technical details of the calculation parameters are outside the scope of this presentation and are not repeated here. The weights,  $W_j$ , used in the C.I., presented in Table IV, were assigned by the authors.

TABLE IV  
CONSEQUENCE CATEGORIES ( $CC_j$ ), & THEIR WEIGHTS,  
 $W_j$

$CC_j$	Description	$W_j$
<b>A Consequences to Human Health</b>		
<b>A.1. Number of people inside the ‘deaths’ radius</b>		
A.1.a	Workers of the installation	0,024
A.1.b	Residents around the installation	0,046
A.1.c	Special population categories (schools, hospitals, jails, etc)	0,11
A.1.d	Transient population (in recreation, shopping & sports areas, etc)	0,363
<b>A.2. Number of people inside the ‘injuries’ radius</b>		
A.2.a	Workers of the installation	0,006
A.2.b	Residents around the installation	0,012
A.2.c	Special population categories	0,029
A.2.d	Transient population	0,095
<b>A3. Number of people inside the ‘annoyance’ radius</b>		
A.3.a	Workers of the installation	0,002
A.3.b	Residents around the installation	0,004
A.3.c	Special population categories	0,009
A.3.d	Transient population	0,031
<b>B Economic Consequences</b>		
<b>B.1 On-site</b>		
B.1.a	Damages to equipment, facilities, etc	0,007
B.1.b	Business Interruption	0,02
<b>B.2 Off-site</b>		
B.2.a	Damage to property (houses, other facilities, etc)	0,006
B.2.b	Damages to infrastructure	0,013
B.2.c	Cost of Environmental restoration	0,035
<b>C Environmental Consequences</b>		
<b>C.1 Pollution</b>		
C.1.a	Soil	0,028
C.1.b	Water (lakes, rivers, shores, aquifers)	0,113
<b>C.2 Damages</b>		
C.2.a	ecosystems, biotopes, protected areas, riverbanks, seashores	0,038
C.2.b	Cultural Assets (historical sites, cemeteries, churches, etc)	0,009

The results and rankings obtained have been compared with those calculated for each scenario using a similar established index, Dow’s ‘Chemical Exposure Index’ (CEI, [6]). Both *C.I.*

and *CEI* rankings obtained for each scenario are presented in Table V.

TABLE V  
C.I. & CEI RANKINGS & THEIR DEVIATIONS.

Installation/ scenario	C.I.		CEI		deviation
	index	rank	index	rank	
1 AM-P-sphere	0,7280	1	957	3	2
2 AM-P-10”	0,6255	5	217	7	2
3 AM-R-sphere	0,1755	11	114	11	0
4 AM-R-10”	0,0143	16	105	12	-4
5 CL-P-sphere	0,7280	1	1000	1	0
6 CL-P-10”	0,7280	1	1000	1	0
7 CL-R-sphere	0,5875	6	925	4	-2
8 CL-R-10”	0,7280	1	853	5	4
9 EO-P-sphere	0,3517	8	207	8	0
10 EO-P-10”	0,4169	7	233	6	-1
11 EO-R-sphere	0,1785	10	171	9	-1
12 EO-R-10”	0,2165	9	158	10	1
13 PO-P-sphere	0,0745	12	76,5	13	1
14 PO-P-10”	0,0485	13	71	14	1
15 PO-R-sphere	0,0351	14	60	15	1
16 PO-R-10”	0,0320	15	56	16	1

Most of the results are within a margin of 1 or 2 positions shift, with only a couple of exceptions, one because of the large number of scenarios taking the same position (1<sup>st</sup>) in *C.I.*

The referenced *CEI* uses the 3 Hazard Distances approach for the determination of the possible accident consequences. Therefore, in order for the results to be comparable, the *C.I.* was calculated using only Human Health group of consequences (12 out of 21 Consequences Categories), assuming that both Economic and Environmental consequences equal to zero. It should be noted that the *C.I.* can be extended to include economic and environmental consequences as well, with no similar methodology existing in the field of chemical accidents assessment, to compare results with.

Furthermore, for each scenario or case, further analysis can be conducted using the proposed *C.I.*, to indicate specific parameters, where possible action taken could pay a lot in return in reducing the index ranking, and consequently the hazards to people, property or the environment. For example,

moving a school, a nursery or a hospital in or out of the hazard distance, could have an impact on the *C.I.*, while in the case of *CEI* only the distance itself is included in the calculations.

#### IX. CONCLUSIONS-DISCUSSION

A methodological approach for the development of hazard classification indices was introduced and presented in this work. The development of the proposed methodology was based on multi-criteria decision-making, as well as on fuzzy logic. Based on the proposed methodology, two new indices, the *Substance Fire Hazard Index*, *SFHI*, which is focused on the major-accident hazards of the substances, as the *Consequences Index*, for the rapid ranking of industrial facilities that use, produce, process or store hazardous substances have been developed and presented. Aim of the proposed indices is the rapid assessment and relative ranking of fire hazards and risks of chemical substances or materials and the ranking of facilities based on accident consequences potential. A number of similar indices are under development, for the relative assessment of toxic and ecotoxic substances, based on different sets of properties. Furthermore, the multi-criteria approach could be also used for handling trade-offs between different substances with both fire and toxic or ecotoxic characteristics.

The proposed methodology could also be used for the development of similar indices, based on any organization's need and views. Also the proposed indices could be modified by any possible user to include their own priorities, or decision environment. It should be emphasized that the principal aim of this work is to present a new methodological approach in dealing with chemical accidents hazards, namely the multi-criteria approach. Each body, institution, authority, etc using or adopting this approach could in the first place adapt the proposed methodology and indices in a way to better suit its needs, or the decision environment within it is working.

For the development of the proposed methodology, the issue of hazard classification was viewed as a multi-criteria decision making problem. Therefore, a multi-criteria decision-making technique, Analytic Hierarchy Process, has been employed. As it has already been stated, AHP is capable for dealing with complex problems, involving multiples criteria of different nature, by analyzing their parameters in a

hierarchical manner. One of its disadvantages, on the other hand, is its inability to deal effectively with problems that cannot be represented by a strict hierarchical structure, namely when there are interconnections or interdependencies among parameters or elements of different subgroups of the same or different levels. This could also be the case in fire hazards classifications and assessment, where various properties could be considered as connecting, or dependant to each other. Such an example of interdependencies between 'Flash Point' and 'Boiling Point' is demonstrated in Table III.

For dealing with this issue, an extension of AHP, the Analytic Network Process – ANP [49], has been introduced. ANP allows for feedback among different elements to be taken into account in the ranking of the alternatives. Therefore, the Hierarchical structure is transposed to a Network structure, resulting in the formation of a 'Supermatrix'. Nevertheless, the number of alternatives cannot exceed a threshold (7-9), because of the size of the 'Supermatrix' that is formed. This limitation is making ANP not suitable as a basis for the development of a generic index aiming in the classification or rapid assessment of a big, or unlimited, number of chemical substances or installations, although its employment for this purpose has been tested.

In order to deal with this issue, further use of fuzzy logic is being examined. More specifically, the use of a 'fuzzy control system' is being examined, to account for any predefined interdependencies among properties, taking into account the values of those properties during the pair wise assessment process.

Possible applications of the proposed *Substance Fire Hazard Index*, *SFHI*, could include:

- Tool for the rapid assessment of substances, based on their instinctive properties.
- Tool for the relative assessment of substances for the selection of a less hazardous one.
- Support tool for the substitution of a hazardous substance with a less hazardous one.
- Tool for 'risk communication' regarding the magnitude of the inherent hazard of a substance.

Possible application fields of the 'Consequences Index' could include:

- Tool for the assessment of existing installations through their relative ranking, and for focusing on installations with the bigger disaster potential.

➤ Tool for the assessment of proposed new installations and for the rapid ranking of alternative sites.

➤ Tool for the assessment of progress made at an existing installation on the reduction of its “consequences potential” (or the opposite).

➤ Tool for “Risk communication” regarding the magnitude of the potential impacts of an accident.

Possible users the introduced methodology or the presented indices could include, among others:

➤ Public authorities responsible for industrial accidents preparedness, prevention and response. E.U. ‘Sevezo’ Directive, for example, requires that Safety Reports are submitted by certain installations and facilities, and that accident scenarios are included in these reports. Competent national authorities could use such ranking tools to prioritise facilities for inspection, as required by the directive.

➤ Existing E.U. occupational safety legislation requires that, in cases where hazardous chemicals are being used, screening should be done for any available safer alternative. Ranking tools could be useful for fast screening in such cases.

➤ Furthermore, many different agencies or companies, non-governmental organizations, etc, have developed similar tools, as it has already been mentioned, based on their own needs. The multi-criteria approach could provide a more suitable framework for a ‘holistic’ assessment of the inherent hazards of chemical substances and installations.

## REFERENCES

- [1] Centre of Chemical Process Safety, *Guidelines for Hazard Evaluation Procedures*, 2<sup>nd</sup> ed. New York: AIChE/CCPS, 1992.
- [2] J.M. Watts, “Fire Risk Ranking”, in: *Handbook of Fire Protection Engineering*. Quincy, MA: Society Fire Protection Engineering / National Fire Protection Association, 1995
- [3] F. Crawley and B. Tyler, *Hazard Identification Methods*. Rugby, UK: European Process Safety Center / IChemE, 2003.
- [4] Dow, *Dow’s Fire & Explosion Index Hazard Classification Guide*, 7<sup>th</sup> ed. New York: AIChE, 1994
- [5] ICI, *The Mond Index*, 2<sup>nd</sup> ed., Imperial Chemical Industries, Cheshire, UK: Imperial Chemical Industries, 1993.
- [6] Dow, *Dow’s Chemical Exposure Index Guide*. New York: AIChE, 1994.
- [7] B.J. Tyler, A.R. Thomas, P. Doran, and T. Greig, “A Toxicity Hazard Index”, *Chem. Health. Saf.*, 3: 19-25, 1996.
- [8] A. Scott, “Environment-accident index: validation of a model”, *J. Haz. Mat.*, 61: 305-12, 1998.
- [9] D.R. Stuhl, *Fundamentals of Fire and Explosion*. New York: AIChE, 1976.
- [10] F.I. Kahn and S.A. Abbasi, “Multivariate Hazard Identification and Ranking System”, *Proc. Saf. Progr.*, 17: 3, 157-70, 1998.
- [11] F.I. Khan, T. Husain, and S.A. Abbasi, “Safety Weighted Hazard Index (SWeHI): A New, User-friendly Tool for Swift yet Comprehensive Hazard Identification and Safety Evaluation in Chemical Process Industries”, *Proc. Saf. Env. Prot.*, 79: 2: 65-80, 2001.
- [12] F.I. Kahn, and S.A. Abbasi, “Accident Hazard Index: A Multi-attribute Method for Process Industry Hazard Rating”, *Proc. Saf. Env. Prot.*, 75: 4: 217-224, 1997.
- [13] S. Shah, U.Fischer, and K.Hungerbühler, “A Hierarchical Approach for the Evaluation of Chemical Process Aspects From the Perspective of Inherent Safety”, *Proc. Saf. Env. Prot.*, 81: 430-443, 2003.
- [14] M. Gentile, W.J. Rogers, and M.S. Mannan, “Development of a Fuzzy Logic-based Inherent Safety Index”, *Proc. Saf. Env. Prot.*, 81: 444-456, 2003.
- [15] M.Y. Gunasekera, and D.W. Edwards, “Estimating the Environmental Impact of Catastrophic Chemical Releases to the Atmosphere. An Index Method for Ranking Alternative Chemical Process Routes”, *Proc. Saf. Env. Prot.*, 81: 463-474, 2003.
- [16] J. Tixier, G. Dusserre, O. Salvi, and D. Gaston, “Review of 62 risk analysis methodologies of industrial plants”, *J. Loss Prev. Proc. Ind.*, 15: 291–303, 2002.
- [17] NFPA 325, *Guide to Fire Hazard Properties of Flammable Liquids, Gases and Volatile Solids*, 1994 ed., Quincy, MA: National Fire Protection Association, par.1-3.1, 1994.
- [18] F.P. Lees, *Loss Prevention in the Process Industries*. Oxford: Butterworth-Heinemann, p.16/25, 1996.
- [19] NFPA 49, *Hazardous Chemicals Data*, 1994 ed., Quincy, MA: National Fire Protection Association, 1994.
- [20] Centre of Chemical Process Safety, *Guidelines for Engineering Design for Process Safety*. New York: AIChE / CCPS, 1996.
- [21] T.L Saaty, *The Analytic Hierarchy Process*. New York: McGraw Hill, 1980.
- [22] R.R. Yager, S. Ovchinnikov, R.M. Tong and H.T. Nguyen, (Editors), *L.A. Zadeh: Selected Papers on Fuzzy Sets and Applications*. New York: John Wiley & Sons, 1987.
- [23] NFPA 704, *Standard System for the Identification of the Hazards of Materials for Emergency Response*, 1996 ed., Quincy, MA: National Fire Protection Association, 1996.
- [24] Center of Chemical Process Safety, *Inherent Safer Chemical Processes: A Life Cycle Approach*, New York: AIChE / CCPS, 1996
- [25] D.C. Hendershot, “Inherently safer chemical process design”, *J. Loss Prev. Proc. Ind.*, 10: 151-157, 1997.
- [26] D.R. Anderson, D.J. Sweeney, and T.A. Williams, *An introduction to management science: quantitative approaches to decision making*. 6<sup>th</sup> ed., St. Paul, Minn.: West Publishing, 1991.
- [27] J. Shields and G. Silcock, “An application of the Hierarchical Approach to Fire Safety”, *Fire Saf J.*, 11: 235-242, 1986.

- [28] F.J. Doddand and H.A. Donegan, "Prioritisation Methodologies in Fire Safety Evaluation", *Fire Technology*, pp.232-249, 2<sup>nd</sup> Quarter 1994.
- [29] F.J. Dodd and H.A. Donegan, "Some considerations in the combination and use of expert opinions in Fire Safety Evaluation", *Fire Saf J.*, 22: 315-327, 1994.
- [30] C.M. Zhao, S.M. Lo, J.A. Lu, and Z. Fang, "A simulation approach for ranking of fire safety attributes of existing buildings". *Fire Saf. J.*, 39: 557-579, 2004.
- [31] S.H. Yeo and K.G Neo, "Inclusion of environmental performance for decision making of welding processes", *J. of Materials Processing Techn.*, 82: pp.78-88 1998.
- [32] E. Cagno, F. Caro, M. Mancini, and F. Ruggeri F., "Using AHP in determining the prior distributions on gas pipeline failures in a robust Bayesian approach", *Reliab. Engn. Syst. Saf.*, 67: 275-284, 2000.
- [33] F.I. Khan, R. Sadiq, and M.M. Haddara, "Risk-based Inspection and Maintenance (RBIM): Multi-attribute Decision-making With Aggregative Risk Analysis", *Proc. Saf. Env. Prot.*, 82 (B6), pp. 398-411, 2004.
- [34] A.N. Paralikas and A.I. Lygeros, "A Multi-Criteria and Fuzzy Logic Based Methodology for the Relative Ranking of the Fire Hazards of Chemical Substances and Installations", *Proc. Saf. Env. Prot.*, 83 (B2), pp. 122-134, 2005.
- [35] J.J. Buckley, "Fuzzy Hierarchical Analysis", *Fuzzy Sets and Systems*, 17: 233-247, 1985.
- [36] ILO, *Globally Harmonized System for the Classification and Labeling of Chemicals-GHS*. Geneva: International Labor Organization, 2001. [Online]. Available: [http://www.ilo.org/public/english/protection/safework/g\\_hs/ghsfinal/index.htm](http://www.ilo.org/public/english/protection/safework/g_hs/ghsfinal/index.htm)
- [37] DOT, *Hazardous Materials Regulations, Title 49 CFR*. Washington D.C.: U.S. Department of Transportation, 2001.
- [38] European Commission, *Council Directive 96/82/EC "on the control of major-accident hazards"*, (OJ No L 10 of 14 January 1997). [Online]. Available: <http://europa.eu.int/comm/environment/seveso/#2>
- [39] NFPA 30, *Flammable and Combustible Liquids Code*, 1996 ed., Quincy, MA: National Fire Protection Association, 1996.
- [40] Federal Emergency Management Agency, U.S. Department of Transportation, and U.S. Environmental Protection Agency, *Handbook of chemical hazard analysis procedures*. Washington D.C.: Federal Emergency Management Agency Publications Office, 1998.
- [41] A. Suarez and C. Kirchsteiger, *A Qualitative Model to Evaluate the Risk Potential of Major Hazardous Industrial Plant*, Ispra: European Commission, JRC, Major Accident Hazards Bureau, 1998.
- [42] T. Ludwig, *Industrial Fire Prevention and Protection*, N.York: Van Nostrand Reinhold, 1991.
- [43] D. Tuhtar, *Fire and Explosion Protection: A Systems Approach*, Chichester: Horwood E, 1989.
- [44] D. Drysdale, *Introduction to Fire Dynamics*. 2<sup>nd</sup> ed., Chichester: John Wiley and Sons, 1999.
- [45] D. Crowl and J. Louvar, *Chemical Process Safety-Fundamentals with Applications*. N.Jersey: Prentice-Hall, 1990.
- [46] C. Kirchsteiger, "Absolute and relative ranking approaches for comparing and communicating industrial accidents", *J. Haz. Mat.*, 59: 31-54, 1998.
- [47] Expert Choice Inc, *Expert Choice 2000: Quick start guide & tutorials*. Pittsburgh, PA: Expert Choice Inc, 2000.
- [48] N.C. Markatos, C.T. Kyranoudis, K. Zografos, and I. Ziomas, "An operational center for managing major chemical industrial accidents", in *Seveso 2000 Conf.*, pp.282-291, Athens, Nov. 10-12, 1999. <http://mahbsrv.jrc.it/Proceedings/Greece-Nov-1999/I2-KYRANOUDIS-z.pdf>.
- [49] T.L. Saaty, *The Analytic Network Process: Decision Making With Dependence And Feedback*. Pittsburgh, PA: RWS Publications, 1996.

# Supply Chain Games

Federico Perea \*

\*University of Seville /Dept. Statistics and OR  
 Faculty of Mathematics. c/Tarfia sn 41012 Seville  
 Email: perea@us.es

**Abstract**—The Supply Chain Problem we propose in this paper arises when, over a graph, a group of nodes offers certain commodity, other nodes require it and a third group of nodes does not need this material nor offer it but is strategically relevant to the distribution plan. The transport of one unit of material to a demand node generates a fixed profit, and the shipping of the material through the arcs has an associated cost. In this work we study the possible cooperation between the nodes of the network, prove that such a cooperative situation is totally balanced and show the relation between these games and other well-known games.

**Keywords**—Cooperative games, networks, core, balancedness.

## I. INTRODUCTION

OPTIMIZATION problems over graphs are extensively used in real applications to model situations like production planning, communication, scheduling, transportation or assignment among others. In such problems it is normally assumed that the resources used in the model are under the control of a individual or a group of individuals having identical interests. In this paper we deal with a situation in which the resources are owned by agents with conflicting objectives, which may consider cooperating with each other in order to get a better global solution. Situations like that may arise, for instance, when there is group of warehouses having a certain product and several shops where that product can produce benefits. The transportation of the material from one point to another generates costs. The warehouses and the shops have to decide how to distribute the material in order to obtain the highest profit. It directly follows that game theory can be employed to analyze such a situation and find fair allocations of the joint profit that the group of agents can make.

A game is a decision process in which a group of agents, called players, converge and act, independently or collectively, under certain rules in order to obtain a result, called payoff. When studying games, one may assume that either all players will cooperate with each other or that the game will be played noncooperatively. Several

models of cooperation on graphs have been studied in the literature, see [3], [5], [6], [9] or [13]. In this paper we study the possible cooperation over a graph when the particular problem we propose, called Supply Chain Problem, is given.

The paper is structured as follows. In section II we introduce graphs and cooperative games, for a more detailed description see [14] and [8]. In section III we present our model of Supply Chain Problem (*SChP* for short) and its formulation as a linear program. Section IV is devoted to introduce the class of cooperative games arising from *SChP*, called Supply Chain Games (*SChG* for short), and show some of its properties, including balancedness. In section V we propose a core allocation that can be computed in polynomial time. Section VI revises some known games on networks and their relation with *SChG*.

## II. PRELIMINARIES

Let  $N$  be a nonempty finite set, which is interpreted as the set of players. We consider a set of ordered pairs of distinct members of  $N$ ,  $A \subset N \times N$ . We refer to these ordered pairs as *arcs*, and we denote the arc from  $i$  to  $j$  as  $(i, j)$ . (So  $(i, j) \neq (j, i)$ ). Then, we say that  $G = (N, A)$  is a *directed graph*.

Given the set of players  $N = \{1, \dots, n\}$ , a coalition of  $N$  is any  $S \subset N$ . The set of all those possible coalitions of  $N$  is denoted by  $2^N$ .

For a game with set of players  $N = \{1, \dots, n\}$ , we define its *characteristic function* as

$$v : 2^N \rightarrow \mathbb{R}.$$

For every  $S \subset N$ ,  $v(S)$  can be interpreted as the maximum profit that the coalition  $S$  can make by acting on its own, without taking into account what the other players  $N \setminus S$  can do. So,  $v(N)$  is the best payoff that the coalition formed by all the players can make. This coalition,  $N$ , is called *the grand coalition*. We define  $v(\emptyset) = 0$ .

Thus, we have that a cooperative game can be represented by its player set  $N = \{1, 2, \dots, n\}$  and its

characteristic function

$$\begin{aligned} v : 2^N &\longrightarrow \mathbb{R} \\ S &\longrightarrow v(S). \end{aligned}$$

Depending on the properties of the characteristic function, players may want to join together or not. Some properties that characteristic functions are desirable to satisfy are the following.

**Definition II.1 (0-normality)** The game  $(N, v)$  is said to be 0-normalized if

$$v(\{i\}) = 0 \quad \forall i \in N.$$

**Definition II.2 (Superaditivity)** We say that the game  $(N, v)$  is superadditive if

$$\forall S, T \subset N : S \cap T = \emptyset \Rightarrow v(S) + v(T) \leq v(S \cup T).$$

**Definition II.3 (Monotonicity)** The game  $(N, v)$  is monotonic if

$$\forall S \subset T \subset N, \quad v(S) \leq v(T).$$

The most important problem we face when dealing with cooperative games is how to divide the total benefit among the players, that is, how to allocate  $v(N)$ . We define an allocation for the game  $(N, v)$  as a vector  $x \in \mathbb{R}^n$ , where its  $i^{th}$  coordinate represents the payoff that player  $i$  receives after the allocation  $x$ . An acceptable property for allocations to satisfy is the *collective rationality principle*, which assures that every coalition  $S$  of  $N$  receives at least what they would obtain by acting without the help of the other players  $N \setminus S$ . Allocations satisfying that principle are called *core* allocations.

**Definition II.4** Let  $(N, v)$  be a cooperative TU-Game. The core of  $(N, v)$ , denoted  $C(N, v)$ , is the set

$$\{x \in \mathbb{R}^n : x(S) \geq v(S) \quad \forall S \subset N, \quad \sum_{i=1}^n x_i = v(N)\},$$

where  $x(S) = \sum_{i \in S} x_i$ , that is, the payoff that coalition  $S$  receives from allocation  $x$ .

We remark that  $x \in C(N, v)$  iff no coalition can improve upon  $x$ . Thus, each member of the core consists of a highly stable payoff distribution.

Unfortunately not all TU-Games have core allocations. The concept of balancedness provides us with a theorem that characterizes those games that have non-empty core.

**Definition II.5** Given is  $(N, v)$  a cooperative game. Let  $\Psi = \{S_1, S_2, \dots, S_r\}$  be a collection of coalitions of  $N$ . We say that  $\Psi$  is a balanced collection if there exist some coefficients  $\gamma_1, \gamma_2, \dots, \gamma_n$ , with  $\gamma_l \geq 0 \quad \forall l = 1, 2, \dots, r$ , such that

$$\sum_{l:i \in S_l} \gamma_l = 1 \quad \forall i \in N.$$

The coefficients  $\gamma_l$  are called balancing weights.

Bondareva [2] and Shapley [11] independently identified the class of games that have non-empty core as the class of balanced games.

**Theorem II.1 (Bondareva and Shapley)** The core of the game  $(N, v)$  is non-empty iff for every balanced collection  $\{S_1, \dots, S_k\}$  with balancing weights  $\lambda_1, \dots, \lambda_k$  the inequality

$$\sum_{j=1}^k \lambda_j v(S_j) \leq v(N)$$

holds.

### III. SUPPLY CHAIN PROBLEM

The classical transportation problem arises when an optimal distribution plan to transport a good on a bipartite network (the set of nodes  $N$  is divided into two disjoint groups  $P, Q$ ) must be determined. The nodes of  $P$  offer that good and the nodes of  $Q$  require the same good. The set of arcs is  $A = \{(i, j) : i \in P, j \in Q\}$ , in other words, there is an arc joining each node that offers material with each node demanding it. In this problem we assume that the transportation of one unit of material from a supply node  $i$  to a demand node  $j$  gives rise to a profit equal to  $b_{ij}$ , the goal being to maximize the total profit generated when covering the demand.

The problem we now deal with is a generalization of the transportation problem, as we will see later. We assume that we have a directed network  $G = (N, A)$ , where  $N$  and  $A$  are the set of nodes and the set of arcs of the graph respectively. Each node  $i \in N$  has a scalar number  $b_i \in \mathbb{R}$  associated with it. If this number is positive, node  $i$  is said to be a *supply* node (node  $i$  can offer  $b_i$  units of the material to be transported), if it is negative we say that node  $i$  is a *demand* node (node  $i$  requires  $-b_i$  units) and if it is null we say that node  $i$  is a *transfer* node. We denote each arc of  $A$  by the ordered pair formed by its initial node and its final node, that is, the arc  $(i, j)$  joins vertices  $i$  and  $j$  in this way. Each arc  $(i, j)$  has a scalar number  $c_{ij} \in \mathbb{R}$  associated with it, which is interpreted as the necessary cost of the shipping

of one unit of material through the arc  $(i, j)$ . We also assume that each unit of covered demand generates a profit of  $K$  units. So, the problem consists of finding a feasible distribution plan that maximizes the total benefit.

**Example III.1** Let us consider the transportation network as depicted in figure 1. The numbers in the nodes represent the amount of material, or demand depending on the sign of the number, that each node has. The number on the arcs are the unitary costs of shipping one unit of material through them. If we suppose that the benefit generated after covering one unit of demand is equal to 15 monetary units,  $K = 15$ , we have a Supply Chain Problem as defined in this section. So, the goal is to maximize the general profit.

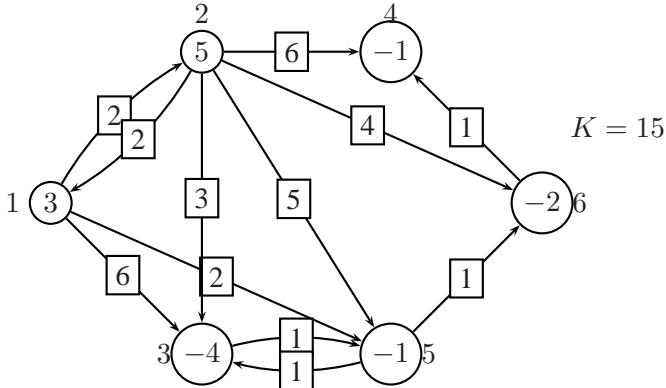


Fig. 1. Transportation Network

Note that in this model it is not necessary to cover all the demand nor to launch all the available offer. The demand of a node  $j$  will be covered if and only if there is a profitable path from the supply nodes to  $j$  and there is some available material.

#### A. Formulation

In this section we define  $SChP$  in a mathematical way. Given a directed graph  $G = (N, A)$ , a vector  $b \in \mathbb{R}^n$  and a matrix  $C \in \mathbb{R}^{n \times n}$  where  $c_{ij}$  represents the unitary cost of shipping one unit through arc  $(i, j) \in A$ , we define the following sets:

$$P := \{i \in N : b_i > 0\}, \quad Q := \{i \in N : b_i < 0\},$$

$$R := \{i \in N : b_i = 0\}.$$

We shall call these sets *supply* set, *demand* set and *transfer* set respectively. It is clear that

$$P \cup Q \cup R = N, \quad P \cap Q = P \cap R = Q \cap R = \emptyset,$$

that is,  $P, Q$  and  $R$  form a partition of  $N$ .

So, given the set of nodes  $N$ , where  $|N| = n$ , the set of arcs  $A \subset N \times N$ , a matrix  $C \in \mathbb{R}^{n \times n}$ , a vector  $b \in \mathbb{R}^n$  and a unitary benefit  $K \in \mathbb{R}$ , we define a Supply Chain Problem as the 5-tuple

$$(N, A, C, b, K).$$

**Example III.2** In the transportation network described in example III.1, the corresponding  $SChP$  is defined as

$$(N, A, C, b, K)$$

where

$$N = \{1, 2, 3, 4, 5, 6\},$$

$$A = \{(1, 2), (1, 3), (1, 5), (2, 1), (2, 3), (2, 4), (2, 5), (2, 6), (3, 5), (5, 3), (5, 6), (6, 4)\},$$

$$C = \begin{pmatrix} - & 2 & 6 & - & 2 & - \\ 2 & - & 3 & 6 & 5 & 4 \\ - & - & - & - & 1 & - \\ - & - & - & - & - & - \\ - & - & 1 & - & - & 1 \\ - & - & - & 1 & - & - \end{pmatrix} \quad b = \begin{pmatrix} 3 \\ 5 \\ -4 \\ -1 \\ -1 \\ -2 \end{pmatrix}$$

and  $K = 15$ .

In the rest of the section we formulate Supply Chain Problems as linear programs. Let  $(N, A, C, b, K)$  be a Supply Chain Problem. Let us consider  $x_{ij}$  as the amount of shipped-through-arc- $(i, j)$  material,  $\forall (i, j) \in A$ . A feasible distribution plan must satisfy several conditions:

- Supply nodes cannot produce new material; that is, the amount of material leaving from certain supply node, *outgoing* material, minus the amount of material that goes to it, *incoming* material, must be less than or equal to what the node can offer. Thus we have that:

$$\sum_{j \in N: (i,j) \in A} x_{ij} - \sum_{k \in N: (k,i) \in A} x_{ki} \leq b_i \quad \forall i \in P. \quad (1)$$

- The incoming material must be equal to the outgoing material in every transfer node, that is, the transfer nodes can not neither create material nor keep material. Mathematically we have that:

$$\sum_{j \in N: (i,j) \in A} x_{ij} - \sum_{k \in N: (k,i) \in A} x_{ki} = 0 \quad \forall i \in R. \quad (2)$$

- In our model, it is also necessary that in a feasible distribution plan every demand node can not receive more than what they request: the amount of incoming flow minus the amount of outgoing flow must

be less than or equal to its demand. In terms of  $x_{ij}$  one has that:

$$\sum_{k \in N: (k,i) \in A} x_{ki} - \sum_{j \in N: (i,j) \in A} x_{ij} \leq -b_i \quad \forall i \in Q. \quad (3)$$

- Besides, the flow must be non-negative,

$$x_{ij} \geq 0 \quad \forall (i,j) \in A. \quad (4)$$

One can observe that the inequalities (2) can be separated into:

$$\sum_{j \in N: (i,j) \in A} x_{ij} - \sum_{k \in N: (k,i) \in A} x_{ki} \leq 0 = b_i \quad \forall i \in R,$$

$$\sum_{j \in N: (i,j) \in A} x_{ij} - \sum_{k \in N: (k,i) \in A} x_{ki} \geq 0 = b_i \quad \forall i \in R,$$

this last equation being equivalent to

$$\sum_{k \in N: (k,i) \in A} x_{ki} - \sum_{j \in N: (i,j) \in A} x_{ij} \leq 0 = -b_i \quad \forall i \in R.$$

Thus, the feasibility constraints (1), (2) and (3) can be expressed as:

$$\sum_{j \in N: (i,j) \in A} x_{ij} - \sum_{k \in N: (k,i) \in A} x_{ki} \leq b_i \quad \forall i \in P \cup R. \quad (5)$$

$$\sum_{k \in N: (k,i) \in A} x_{ki} - \sum_{j \in N: (i,j) \in A} x_{ij} \leq -b_i \quad \forall i \in Q \cup R. \quad (6)$$

Regarding the objective function, let us calculate it step by step:

- We want to maximize the total benefit, that is, the sum of the demands covered in each node of  $Q$ . Let us consider  $i \in Q$ . The total demand of material that is covered in node  $i$  after the distribution plan  $(x_{ij})_{(i,j) \in A}$  is:

$$\sum_{k \in N: (k,i) \in A} x_{ki} - \sum_{j \in N: (i,j) \in A} x_{ij},$$

that is, the amount of incoming material to  $i$  minus the amount of outgoing material from  $i$ . Thus, we have to maximize the total demand covered by the flow, which is:

$$\sum_{i \in Q} \left( \sum_{k \in N: (k,i) \in A} x_{ki} - \sum_{j \in N: (i,j) \in A} x_{ij} \right). \quad (7)$$

The reader should note that we have to multiply the above expression times the unitary benefit  $K$  in order to obtain the expression of the exact profit.

Equation (7) can be simplified as follows:

$$\begin{aligned} \sum_{i \in Q} \left( \sum_{k \in N: (k,i) \in A} x_{ki} - \sum_{j \in N: (i,j) \in A} x_{ij} \right) &= \\ \sum_{i \in Q} \sum_{k \in N: (k,i) \in A} x_{ki} - \sum_{i \in Q} \sum_{j \in N: (i,j) \in A} x_{ij} &= \\ \sum_{i \in Q} \left( \sum_{k \in P \cup R: (k,i) \in A} x_{ki} + \sum_{k \in Q: (k,i) \in A} x_{ki} \right) - \\ \sum_{i \in Q} \left( \sum_{j \in P \cup R: (i,j) \in A} x_{ij} + \sum_{j \in Q: (i,j) \in A} x_{ij} \right) &= \\ \sum_{i \in Q} \sum_{k \in P \cup R: (k,i) \in A} x_{ki} - \sum_{j \in P \cup R: (i,j) \in A} x_{ij}. \end{aligned}$$

- The transportation through each arc gives rise to a cost, so we must minimize

$$\sum_{(i,j) \in A} c_{ij} x_{ij}.$$

Therefore, our objective function is to maximize

$$\begin{aligned} K \sum_{i \in Q} \left( \sum_{k \in P \cup R: (k,i) \in A} x_{ki} - \sum_{j \in P \cup R: (i,j) \in A} x_{ij} \right) - \\ \sum_{(i,j) \in A} c_{ij} x_{ij}. \end{aligned} \quad (8)$$

To summarize, given a Supply Chain Problem

$$(N, A, C, b, K)$$

where  $N = \{1, \dots, n\}$ ,  $A \subset N \times N$ ,  $C \in \mathbb{R}^{n \times n}$ ,  $b \in \mathbb{R}^n$  and  $K \in \mathbb{R}$ , an optimal distribution plan is given by one optimal solution to the linear program consisting of maximizing (8) under the constraints (5), (6) and (4).

**Example III.3** The formulation of the example III.1 is

$$\begin{aligned} \max \quad & 15(x_{13} + x_{15} + x_{23} + x_{24} + x_{25} + x_{26}) - \\ & \sum_{(i,j) \in A} c_{ij} x_{ij} \\ \text{s.t.:} \quad & x_{12} + x_{13} + x_{15} - x_{21} \leq 3 \\ & x_{21} + x_{23} + x_{24} + x_{25} + x_{26} - x_{12} \leq 5 \\ & x_{13} + x_{23} + x_{53} - x_{35} \leq 4 \\ & x_{24} + x_{64} \leq 1 \\ & x_{15} + x_{25} + x_{35} - x_{53} - x_{56} \leq 1 \\ & x_{26} + x_{56} - x_{64} \leq 2 \\ & x_{ij} \geq 0 \quad \forall (i,j) \in A \end{aligned}$$

and one can see that an optimal solution to this linear program is given by the distribution plan:

$$x_{15} = 3, x_{26} = 1, x_{23} = 4, x_{64} = 1, x_{56} = 2,$$

and  $x_{ij} = 0$  in any other case.

In the previous example all the demand was covered, but this is not a general characteristic of *SChP*. There

exist  $SChP$  where an optimal distribution plan does not cover all the demand, or does not launch all the material that supply nodes own, or both cases at the same time, as we can see in the following example.

**Example III.4** One can check that the distribution plan  $x_{12} = 2, x_{23} = 0, x_{13} = 0$  is optimal in the Supply Chain Problem described in figure 2 and, nevertheless, the supply node 1 keeps one surplus unit and the demand node 3 does not have its demand covered. This is due to the fact that the necessary cost of sending one unit of material from node 1 to node 3 is higher than the benefit that this unit would generate in node 3.

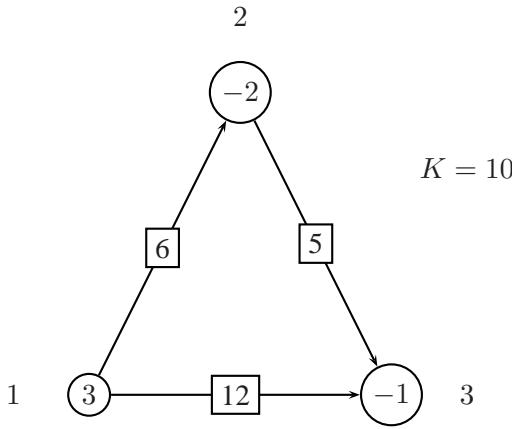


Fig. 2. Transportation Network

#### IV. SUPPLY CHAIN GAMES

After having defined Supply Chain Problems, we are now interested in studying the possible cooperation that can arise within the set of nodes. So, given a Supply Chain Problem  $(N, A, C, b, K)$  we define the corresponding Supply Chain Game as follows.

The set of players is  $N = \{1, \dots, n\}$ , every node is owned by one player and each player is associated with the only node that it owns. Then, to avoid a more complicated notation, we denote the player that owns node  $i$  by  $i$ . Thus, we have supply players, demand players and transfer players, depending on the kind of node they own, that is, we have the set of supply players  $P = \{i \in N : b_i > 0\}$ , the set of demand players  $Q = \{i \in N : b_i < 0\}$  and the set of transfer players  $R = \{i \in N : b_i = 0\}$ .

Now we have to define  $v$ , the characteristic function of the game,

$$\begin{aligned} v : 2^N &\rightarrow \mathbb{R} \\ S &\rightarrow v(S) \end{aligned}$$

That is to say, we need to find the maximum profit that each group of players can make by acting on their own, without taking into account what the other players do.

For each  $S \subset N$  we have a transportation subnetwork  $(S, A_S, C_S, b_S, K)$ , where  $A_S = (S \times S) \cap A$ , and  $C_S$  and  $b_S$  are the restrictions of  $C$  and  $b$  to  $S$  respectively. In the same way, we have the supply, demand and transfer sets of  $S$ , defined as:

$$P_S = P \cap S, Q_S = Q \cap S, R_S = R \cap S.$$

The above subnetwork has an optimal distribution plan that gives us the maximum profit that the coalition  $S$  can make, and is the solution to the linear program which consists of maximizing the function

$$K \sum_{i \in Q_S} \left( \sum_{k \in P_S \cup R_S : (k,i) \in A_S} x_{ki} - \sum_{j \in P_S \cup R_S : (i,j) \in A_S} x_{ij} \right) - \sum_{(i,j) \in A_S} c_{ij} x_{ij} \equiv f_S(x)$$

subject to the constraints

$$\sum_{j \in S : (i,j) \in A_S} x_{ij} - \sum_{k \in S : (k,i) \in A_S} x_{ki} \leq b_i \quad \forall i \in P_S \cup R_S,$$

$$\sum_{k \in S : (k,i) \in A_S} x_{ki} - \sum_{j \in S : (i,j) \in A_S} x_{ij} \leq -b_i \quad \forall i \in Q_S \cup R_S$$

and

$$x_{ij} \geq 0 \quad \forall (i,j) \in A_S.$$

This problem is denoted by  $Pr(S) \forall S \subset N$ .

Then, we define the cooperative game with transferable utility associated with the Supply Chain Problem  $(N, A, C, b, K)$  as follows.

**Definition IV.1** Let  $(N, A, C, b, K)$  be a Supply Chain Problem. The associated Supply Chain Game is the cooperative game with transferable utility  $(N, v)$ , where  $v(S)$  is the optimal value of problem  $Pr(S)$  for every nonempty coalition  $S \subset N$ , and  $v(\emptyset) = 0$ .

**Example IV.1** Let us consider the Supply Chain Game arising from the Supply Chain Problem on figure 3. One can check that the characteristic function of this game is

$S$	$\{1, 3\}$	$\{2, 3\}$	$\{1, 2, 3\}$
$v(S)$	10	7	12

and  $v(S) = 0$  in any other case.

The first consequence that one can get from the definition of Supply Chain Games is that they are well defined. This is due to the fact that, for each coalition  $S$ , the

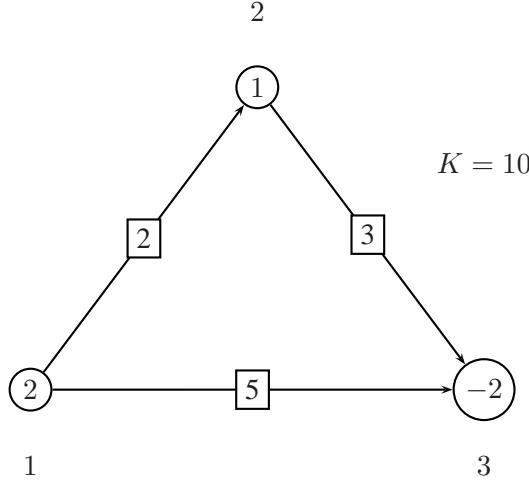


Fig. 3. Transportation Network

distribution plan consisting of every player doing nothing ( $x_{ij} = 0 \forall \text{ arc } (i, j)$ ) is feasible and gives rise to a value of the objective function equal to 0.

#### A. Properties of Supply Chain Games

In this section we explore some interesting properties of Supply Chain Games. The first ones are summarized in the following proposition.

**Proposition IV.1** Let  $(N, v)$  be a Supply Chain Game. One has that:

- 1)  $(N, v)$  is 0-normalized.
- 2)  $(N, v)$  is superaditive.
- 3)  $(N, v)$  is monotonic.

*Proof:*

- 1) Let  $(N, v)$  be a *SChG*. Given  $i \in N$ , it is clear that  $A_{\{i\}}$ , the arcs that  $i$  owns, is the empty set. Thus, the objective function of the problem to be solved in order to calculate  $v(\{i\})$ ,  $f_{\{i\}}$ , is the function equal to zero. Then we conclude that  $v(\{i\}) = 0$ .
- 2) Let  $(N, v)$  be a *SChG*. Consider  $S, T \subset N$  such that  $S \cap T = \emptyset$ . Let us see that  $v(S) + v(T) \leq v(S \cup T)$ .

Let  $x^S$  and  $x^T$  be two optimal distribution plans for the coalitions  $S$  and  $T$  respectively, that is,  $f_S(x^S) = v(S)$  and  $f_T(x^T) = v(T)$ .

We define

$$x^* = (x_{ij}^*) : x_{ij}^* := \begin{cases} x_{ij}^S & \text{if } (i, j) \in A_S \\ x_{ij}^T & \text{if } (i, j) \in A_T \\ 0 & \text{otherwise} \end{cases}$$

for every  $(i, j) \in A_{S \cup T}$ .

By definition

$$A_{S \cup T} = A \cap ((S \cup T) \times (S \cup T)).$$

Let us see that  $x^*$  is feasible for  $Pr(S \cup T)$ . To do so, we have to prove that

$$\sum_{j \in S \cup T : (i, j) \in A} x_{ij}^* - \sum_{k \in S \cup T : (k, i) \in A} x_{ki}^* \leq b_i, \quad (9)$$

$\forall i \in P_{S \cup T} \cup R_{S \cup T}$ . We also need to prove that

$$\sum_{k \in S \cup T : (k, i) \in A} x_{ki}^* - \sum_{j \in S \cup T : (i, j) \in A} x_{ij}^* \leq -b_i, \quad (10)$$

$\forall i \in Q_{S \cup T} \cup R_{S \cup T}$ , and

$$x_{ij}^* \geq 0 \quad \forall (i, j) \in A_{S \cup T}. \quad (11)$$

We prove (9), the proof of (10) being analogous.

•

$$\begin{aligned} \sum_{j \in S \cup T : (i, j) \in A} x_{ij}^* - \sum_{k \in S \cup T : (k, i) \in A} x_{ki}^* = \\ \sum_{j \in S : (i, j) \in A} x_{ij}^* - \sum_{k \in S : (k, i) \in A} x_{ki}^* + \\ + \sum_{j \in T : (i, j) \in A} x_{ij}^* - \sum_{k \in T : (k, i) \in A} x_{ki}^*. \end{aligned} \quad (12)$$

We have that  $P_S \cup P_T \cup R_S \cup R_T$  is a partition of  $P_{S \cup T} \cup R_{S \cup T}$ . Let us suppose that  $i \in P_S$ , the proof for the other cases is analogous, then

$$x_{ij}^* = \begin{cases} x_{ij}^S & \text{if } j \in S \\ 0 & \text{otherwise} \end{cases}$$

Thus (12) becomes

$$\sum_{j \in S : (i, j) \in A} x_{ij}^S - \sum_{k \in S : (k, i) \in A} x_{ki}^S \leq b_i, \quad (13)$$

since  $x^S$  is feasible for  $Pr(S)$ .

• Clearly  $x_{ij}^* \geq 0 \forall (i, j) \in A_{S \cup T}$ .

Thus,  $x^*$  is feasible for  $Pr(S \cup T)$ . By the optimality of  $v(S \cup T)$  in  $Pr(S \cup T)$  one has that  $f_{S \cup T}(x^*) \leq v(S \cup T)$ .

Besides, it is easy to check that  $f_{S \cup T}(x^*) = f_S(x^S) + f_T(x^T)$ . So we conclude that

$$v(S) + v(T) = f_S(x^S) + f_T(x^T) = f_{S \cup T}(x^*) \leq v(S \cup T). \quad (14)$$

- 3) By joining the nonnegativity property of  $v$  with the fact that *SChG* are superaditive, we conclude that *SChG* are monotonic.

■

The following step is to prove that  $SChG$  have non-empty core.

**Theorem IV.1** Supply Chain Games have non-empty core.

*Proof:* Let  $(N, A, C, b, K)$  be a Supply Chain Problem and  $(N, v)$  its associated Supply Chain Game. We firstly prove that  $(N, A, C, b, K)$  is balanced.

Let  $\{S_1, \dots, S_r\}$  be a balanced collection and  $\{y_1, \dots, y_r\}$  its balancing weights, that is,  $\sum_{j: i \in S_j} y_j = 1 \forall i \in N$ . In order to prove the balancedness of  $(N, v)$  we have to prove that  $\sum_{l=1}^r y_l v(S_l) \leq v(N)$ .

Let  $x^l$  be an optimal distribution plan for the coalition  $S^l$ ,  $l = 1, \dots, r$ . We have that

$$v(S_l) = f_{S_l}(x^l).$$

Let us consider

$$\delta^l(i, j) = \begin{cases} y_l & \text{if } (i, j) \in A_{S_l} \\ 0 & \text{otherwise} \end{cases}$$

$$\forall l = 1, \dots, r.$$

Let us define  $x^*$ , a distribution plan for the complete network, as follows:

$$x_{ij}^* = \sum_{l=1}^r \delta^l(i, j) x_{ij}^l.$$

We shall see that  $x^*$  is feasible for  $Pr(N)$ , that is to say, that  $x^*$  is a feasible distribution plan for the whole network. To do that we need to check:

1)  $x_{ij}^* \geq 0$ . Clearly by its definition.

2) Let  $i \in P \cup R$ . We have to prove that

$$\begin{aligned} \sum_{j \in N: (i, j) \in A} x_{ij}^* - \sum_{k \in N: (k, i) \in A} x_{ki}^* &\leq b_i. \\ \sum_{j \in N: (i, j) \in A} x_{ij}^* - \sum_{k \in N: (k, i) \in A} x_{ki}^* &= \\ \sum_{j \in N: (i, j) \in A} \sum_{l=1}^r \delta^l(i, j) x_{ij}^l &- \\ \sum_{k \in N: (k, i) \in A} \sum_{l=1}^r \delta^l(k, i) x_{ki}^l &= \\ \sum_{l=1}^r \left( \sum_{j \in N: (i, j) \in A} \delta^l(i, j) x_{ij}^l \right. &- \\ \left. \sum_{k \in N: (k, i) \in A} \delta^l(k, i) x_{ki}^l \right) &= \\ \sum_{l: i \in S_l} \left( \sum_{j \in S_l: (i, j) \in A_{S_l}} y_l x_{ij}^l \right. &- \\ \left. \sum_{k \in S_l: (k, i) \in A_{S_l}} y_l x_{ki}^l \right) &= \end{aligned}$$

$\sum_{l: i \in S_l} y_l \left( \sum_{j \in S_l: (i, j) \in A_{S_l}} x_{ij}^l - \sum_{k \in S_l: (k, i) \in A_{S_l}} x_{ki}^l \right) \leq$   
 $\sum_{l: i \in S_l} y_l b_i = b_i \sum_{l: i \in S_l} y_l = b_i$ . And we conclude  
that the inequality holds.

3) Analogously to 2) we prove that

$$\sum_{k \in N: (k, i) \in A} x_{ki}^* - \sum_{j \in N: (i, j) \in A} x_{ij}^* \leq -b_i \quad \forall i \in Q \cup R$$

From 1), 2) and 3) we conclude that the distribution plan  $x^*$  is feasible for the linear program  $Pr(N)$ , which defines the value of  $v(N)$ .

On the other hand,

$$\begin{aligned} \sum_{l=1}^r y_l v(S_l) &= \sum_{l=1}^r y_l \left( K \sum_{i \in Q_{S_l}} \sum_{k \in P_{S_l} \cup R_{S_l}: (k, i) \in A_{S_l}} x_{ki}^l - \right. \\ &\quad \left. \sum_{j \in P_{S_l} \cup R_{S_l}: (i, j) \in A_{S_l}} x_{ij}^l \right) - \sum_{(i, j) \in A_{S_L}} c_{ij} x_{ij}^l = \\ &K \sum_{l=1}^r \sum_{i \in Q_{S_l}} \sum_{k \in P_{S_l} \cup R_{S_l}: (k, i) \in A_{S_l}} y_l x_{ki}^l - \\ &K \sum_{l=1}^r \sum_{i \in Q_{S_l}} \sum_{j \in P_{S_l} \cup R_{S_l}: (i, j) \in A_{S_l}} y_l x_{ij}^l - \\ &\quad \sum_{l=1}^r \sum_{(i, j) \in A_{S_l}} y_l c_{ij} x_{ij}^l = \\ &K \sum_{l=1}^r \sum_{i \in Q} \sum_{k \in P \cup R: (k, i) \in A} \delta^l(k, i) x_{ki}^l - \\ &K \sum_{l=1}^r \sum_{i \in Q} \sum_{j \in P \cup R: (i, j) \in A} \delta^l(i, j) x_{ij}^l - \\ &\quad \sum_{l=1}^r \sum_{(i, j) \in A} \delta^l(i, j) c_{ij} x_{ij}^l = \\ &K \sum_{i \in Q} \sum_{k \in P \cup R: (k, i) \in A} \sum_{l=1}^r \delta^l(k, i) x_{ki}^l - \\ &K \sum_{i \in Q} \sum_{j \in P \cup R: (i, j) \in A} \sum_{l=1}^r \delta^l(i, j) x_{ij}^l - \\ &\quad \sum_{(i, j) \in A} \sum_{l=1}^r \delta^l(i, j) c_{ij} x_{ij}^l = K \sum_{i \in Q} \sum_{k \in P \cup R: (k, i) \in A} x_{ki}^* - \\ &K \sum_{i \in Q} \sum_{j \in P \cup R: (i, j) \in A} x_{ij}^* - \sum_{(i, j) \in A} c_{ij} x_{ij}^* = \\ &K \sum_{i \in Q} \sum_{k \in P \cup R: (k, i) \in A} x_{ki}^* - \sum_{j \in P \cup R: (i, j) \in A} x_{ij}^* - \\ &\quad \sum_{(i, j) \in A} c_{ij} x_{ij}^* = f_N(x^*) = \underbrace{\leq}_{x^* \text{ is feasible for } Pr(N)} v(N). \end{aligned}$$

We have proven that  $\sum_{l=1}^r y_l v(S_l) \leq v(N)$  for each balanced collection  $\{S_1, \dots, S_r\}$  with balancing weights

$(y_1, \dots, y_r)$ . Thus, we can assure that the game  $(N, v)$  is balanced.

Since every subgame of a Supply Chain Game is also a Supply Chain Game, we can say that  $(N, v)$  is totally balanced and, applying the Bondareva-Shapley theorem, see theorem II.1, we have that the core of  $(N, v)$  is non empty. ■

## V. DUALITY: AN ALLOCATION IN THE CORE

In this section we give an allocation in the core of any Supply Chain Game. In  $SChG$  the characteristic function is given by the optimal value of the linear problem  $Pr(S)$ . One can check that its dual problem, called from now on  $Dr(S) \forall S \subset N$ , is the following linear program:

$$\begin{aligned} \min \quad & g_S(u) \equiv \sum_{i \in P_S} b_i u_i - \sum_{j \in Q_S} b_j u_j \\ \text{s.t.:} \quad & u_i - u_j \geq -c_{ij} \quad \forall \{i, j\} \in (P_S \cup R_S)^2 \\ & u_i + u_j \geq -c_{ij} + K \quad \forall \{i, j\} \in (P_S \cup R_S) \times Q_S \\ & -u_i - u_j \geq -c_{ij} - K \quad \forall \{i, j\} \in Q_S \times (P_S \cup R_S) \\ & -u_i + u_j \geq -c_{ij} \quad \forall \{i, j\} \in Q_S^2 \\ & u_i \geq 0 \quad \forall i \in P_S \cup Q_S \\ & u_i \in \mathbb{R} \quad \forall i \in R_S \end{aligned} \quad (15)$$

Obviously, the above constraints are restricted to those pairs  $\{i, j\}$  such that  $(i, j) \in A_S$ .

Consider  $u^*$  an optimal solution to  $Dr(N)$  and  $t^*$  an optimal solution to  $Pr(N)$ . By the duality theorem, see [1], one has that

$$\sum_{i \in P} b_i u_i^* - \sum_{j \in Q} b_j u_j^* = g_N(u^*) = f_N(t^*) = v(N).$$

Let us define the following allocation:

$$x = (x_i)_{i \in N}, \quad x_i = u_i^* |b_i| \quad \forall i \in N,$$

where  $|\cdot|$  is the absolute value function. We shall see that  $x$  is an allocation in the core of the game.

1)  $x(N) = \sum_{i=1}^n x_i = \sum_{i=1}^n u_i^* |b_i| = \sum_{i \in P} b_i - \sum_{j \in Q} b_j = v(N)$ . So, the efficiency property is satisfied.

2) We now have to prove that  $x(S) \geq v(S)$  for all  $S \in 2^N$ . Let  $S$  be a coalition in  $N$ .

$$x(S) = \sum_{i \in S} x_i = \sum_{i \in P_S} u_i^* b_i - \sum_{j \in Q_S} u_j^* b_j = g_S(u^*). \quad (16)$$

Let  $x^S \in \mathbb{R}^{|S|} : x_i^S = u_i^* b_i \forall i \in S$ . Clearly the restriction of  $u^*$  to  $S$  is feasible for  $Dr(S)$ , which is the same problem as  $Dr(N)$  but without some of its constraints. Let  $u^S$  be an optimal solution to

$Dr(S)$ . We have that equation (16) is less than or equal to

$$g_S(u^S) = v(S). \quad (17)$$

Thus, we have proven that the allocation  $x$  satisfies the collective rationality principle.

From 1) and 2) we conclude that  $x \in C(N, v)$ .

The set consisting of all the allocations that can be obtained from optimal solutions to  $Dr(N)$  following the above process is the well-known Owen Set, see [7].

Since solving  $Dr(N)$  can be done in polynomial time (see [12]), the reader may note that the given procedure provides a core allocation in polynomial time.

**Example V.1** Let us consider the Supply Chain Game given by the Supply Chain Problem described in figure 4.

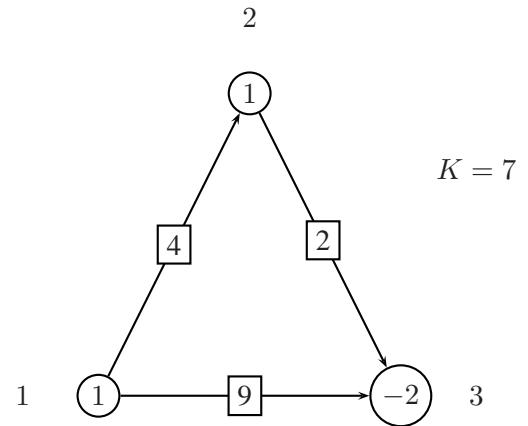


Fig. 4. Transportation Network

The characteristic function of this game is:

S	{2,3}	{1,2,3}
v(S)	5	6

and  $v(S) = 0$  for any other coalition. One can check that the dual linear program associated to this game is:

$$\begin{aligned} \max \quad & u_1 + u_2 + 2u_3 \\ \text{s.t.:} \quad & u_1 - u_2 \geq -4 \\ & u_1 + u_3 \geq -2 \\ & u_2 + u_3 \geq 5 \\ & u_1, u_2 \geq 0 \end{aligned}$$

One optimal solution to this problem is

$$u^* = (1, 5, 0)$$

and the corresponding Owen allocation is

$$(1, 5, 0),$$

which is an allocation in the core of the game.

## VI. SUPPLY CHAIN GAMES AND THE STATE OF THE ART

Now we explore the relationship between  $SChG$  and some other games that have been studied in the literature.

### A. Flow Games and Supply Chain Games

One interesting conclusion from the balancedness of Supply Chain Games is that they are flow games. This is due to the fact that every totally balanced game is a flow game itself, see [4].

### B. Transportation Games and Supply Chain Games

Now we are going to see that the class of Transportation Games is included in the class of Supply Chain Games. Let  $(N, v)$  be a Transportation Game defined from the Transportation Problem  $(P, Q, B, p, q)$ , see [10]. If we consider  $R = \emptyset$ ,  $C = -B$ ,  $K = 0$ ,  $b = ((p_i)_{i \in P}, (-q_j)_{j \in Q})$ ,  $A = P \times Q$ ,  $N = P \cup Q$ , then the Supply Chain Game  $(N', v')$  defined from the Supply Chain Problem  $(N, A, C, b, K)$  is equivalent to the transportation game  $(N, v)$  defined before. Let us prove it.

- $N' = P \cup Q \cup R = P \cup Q = N$ .
- The value of  $v'(S)$  is given when we solve  $Pr(S)$ . If we take

$$K = 0, c_{ij} = -b_{ij}, A = P \times Q$$

then the objective function  $f_S(x)$  is equal to

$$\max \sum_{i \in P} \sum_{j \in Q} b_{ij} x_{ij},$$

and the constraints of  $Pr(S)$  become

$$\sum_{j \in Q} x_{ij} \leq p_i \quad \forall i \in P$$

and

$$\sum_{k \in P} x_{ki} \leq q_i \quad \forall j \in Q.$$

Moreover, due to the fact that  $A = P \times Q$ , we get that the problem we have to solve in order to find the value of  $v'(S)$  is

$$\begin{aligned} \max & \sum_{i \in P} \sum_{j \in Q} b_{ij} x_{ij} \\ \text{s.t.: } & \sum_{j \in Q} x_{ij} \leq p_i \quad \forall i \in P \\ & \sum_{k \in P} x_{ki} \leq q_i \quad \forall j \in Q \\ & x_{ij} \geq 0 \quad \forall i \in P, \forall j \in Q \end{aligned}$$

which is an equivalent problem to the one we have to solve in order to calculate  $v(S)$ . So, we conclude that  $v'(S) = v(S)$  for all  $S \subset N$ .

**Theorem VI.1** Transportation Games are Supply Chain Games.

*Proof:* The above discussion. ■

The reciprocity of the previous theorem is not true, that is, there are Supply Chain Games that are not Transportation Games. Let us see an example.

**Example VI.1** Let us consider the  $SChG$  as depicted in figure 5.

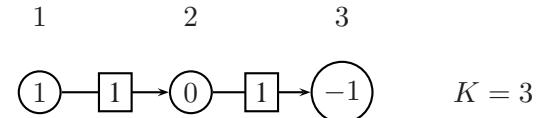


Fig. 5. Supply Chain Game

One can see that  $v(N) = 1$ , and  $v(S) = 0 \forall S \neq N$ . It is obvious that in a Transportation Game with  $v(N) > 0$ , there exists a pair  $\{i, j\}$ ,  $i \in P$ ,  $j \in Q$  such that  $v(\{i, j\}) > 0$ . This does not happen in this example, so we conclude that  $SChG \not\subseteq TG$ .

Since Assignment Games are a particular case of Transportation Games, we conclude that Assignment Games are Supply Chain Games as well.

### C. Shortest Path Games and Supply Chain Games

In this section we study the relationship between Supply Chain Games and Shortest Path Games ( $SPG$  for short) as presented in [3].

Since  $SChG$  have been proven to be monotonic, and in [3] it is shown that the class of  $SPG$  coincides with the class of monotonic games ( $MO$ ), we conclude that  $SChG \subset SPG$ .

The converse implication is not true, as shown in the following example:

**Example VI.2** Consider the shortest path game  $\sigma$  given by  $N = \{1, 2\}$ ,  $g = 6$  and  $\Sigma$  and  $o$  as depicted in figure 6.

The characteristic function  $v$  for this game is

$$v(\{1\}) = 3, v(\{2\}) = 2, v(\{1, 2\}) = 4.$$

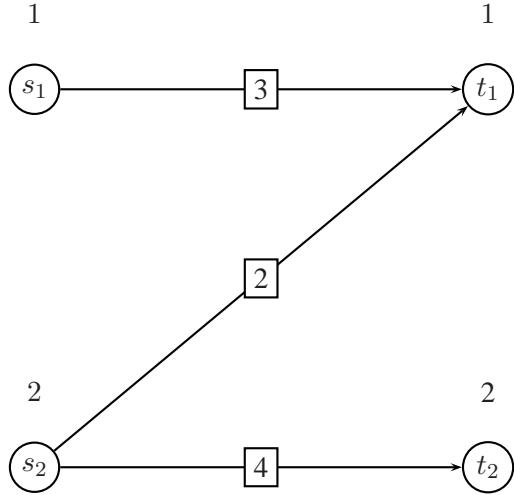


Fig. 6. Shortest Path Game

Clearly  $C(N, v) = \emptyset$ , so  $(N, v)$  cannot be a Supply Chain Game, since  $SChG$  are balanced.

Nevertheless, we prove that a subclass of  $SPG$  is included in the class of  $SChG$ .

**Proposition VI.1** Consider Shortest Path Game  $\sigma = (\Sigma, N, o, g)$  where  $\Sigma = (N, A, L, s, t)$  and  $o(i) = i \forall i \in N$ . If  $|s| = 1$  or  $|t| = 1$  then  $\sigma$  is a Supply Chain Game.

*Proof:* It is clear that, if  $|s| = 1$ , the objective for any profitable coalition is to send the only unit of commodity from  $s$  to the nearest  $j \in t$ . Analogously, if  $|t| = 1$  then the optimal transportation plan would be to find the closest  $i \in s$  to  $t$  and ship the unit that  $i$  offers to  $t$ . In both cases, from the definition of  $SChG$ , it is the same objective as in  $SChG$ , and the result follows. ■

In [3] some conditions for  $SPG$  to have non-empty core are given. From proposition VI.1 we have the following result.

**Theorem VI.2** If  $|s| = 1$  or  $|t| = 1$ , then the corresponding  $SPG$  is balanced.

*Proof:* It is clear from the fact that those  $SPG$  are  $SChG$ , which are balanced. ■

## REFERENCES

- [1] Bazaraa, M.S., Jarvis, J.J., Sherali, H.D. "Linear programming and network flows." *John Wiley & Sons* (1990).
- [2] Bondareva, O. *Certain Applications of the methods of linear programming to the theory of cooperative games*. (In Russian). Problemy Kibernetiky, 10:119-139 (1963).
- [3] Fragnelli, V., García-Jurado, I., Méndez-Naya, L. *On Shortest Path Games*. Mathematical Methods of Operations Research 52:251-264. Springer-Verlag (2000).
- [4] Kalai, E. & Zemel, E. *Totally Balanced Games and Games of Flow*, Mathematics of Operations Research 7(vol 3):476-478 (1982).
- [5] Markakis, E., Amin, S. *On the core of the multicommodity flow game*, Decision Support Systems 39:3-10, (2005).
- [6] Myerson, R.B. *Graphs and cooperation in games*, Mathematics of Operations Research Vol.2 No.3 (1977).
- [7] Owen, G. *On the Core of Linear Production Games*, Mathematical Programming 9:358-370 (1975).
- [8] Owen, G. "Game Theory." *Academic Press, San Diego* (1995).
- [9] Puerto, J., García-Jurado, I., Fernández, F.R. *On the core of a class of location games*, Mathematical Methods of Operations Research 54:373-385, (2001).
- [10] Sánchez, J. *El problema del transporte. Una aproximación desde la Teoría de Juegos*. PhD Thesis. Murcia (1998).
- [11] Shapley, L. *On balanced sets and cores*. Naval Research Logistics Quarterly 14:453-460 (1967).
- [12] Vazirani, Vijay V. "Approximation Algorithms," *Springer-Verlag*, (2001).
- [13] Voorneveld, M., Grahn, S. *Cost allocation in shortest path games*. Mathematical Methods of Operations Research 56:323-340 (2002).
- [14] Wilson, R.J., Watkins, J.J. "GRAPHS: An Introductory Approach" *Library of Congress Cataloging-in-Publication Data* Singapore (1990).

## ACKNOWLEDGEMENTS

The author would like to thank the anonymous referees for their valuable comments and advise.

# Hybrid Supply Chain Modelling – Combining LP-Models and Discrete-Event Simulation

Margaretha Preusser<sup>\*</sup>, Christian Almeder<sup>\*</sup>, Richard F. Hartl<sup>\*</sup>, and Markus Klug<sup>†</sup>

<sup>\*</sup>University of Vienna

Department of Business Studies

Brünnerstr. 72, 1210 Vienna, Austria

Email: {margaretha.preusser, christian.almeder, richard.hartl}@univie.ac.at

<sup>†</sup>ARC Seibersdorf research

2444 Seibersdorf, Austria

Email: markus.klug@arcs.ac.at

**Abstract**—In this paper we present a new approach to support the operational decisions for supply chain networks by combining LP-modelling and discrete-event simulation. The network structure is assumed as given, but the transportation, production and inventory amounts are optimized. We develop an LP model of a general supply chain network including production, inventory and transportation using different transportation modes. Furthermore we extend it to a multi-objective problem and present a solution method for that extension. The LP is connected with a discrete-event simulation model representation of the supply chain including its full complexity and nonlinearities. Alternating the LP model is solved and using its solution simulation runs are performed to gain new estimates for the linearized parameters of the LP. This method allows improving the supply chain by approximating an optimal solution.

**Keywords**—Supply chain management, LP-Models, Discrete-event simulation, Multi-objective optimization.

## I. INTRODUCTION

THE development in the economy during the last decades shows a clear globalization trend. Firms act globally and large networks of manufacturing and distribution facilities are built to meet these trends. Hence, supply chain networks increase in size, i.e., the number of facilities within the network, as well as in material flow. So managing supply chains becomes a more important but also a more sophisticated task.

In this paper we want to investigate a general supply chain network with different facilities (suppliers, manufacturers, distributors) and different transportation modes connecting those facilities. The aim is to reduce costs by simultaneously optimizing the production/transportation schedule and reducing inventory levels. If we compare our problem to the tasks in the supply chain matrix (cf. [18]), the problem is a combination of several tasks in the matrix: production planning, distribution planning and transport planning. Aspects of the integration of transport and production planning within supply chains have been investigated in several papers (e.g. [1], [4]), whereas Meyr [11] and Schneeweiss [16]) present

combined planning approaches for different levels of decision making. For simplicity, we assume that there is a central decision maker in the supply chain with perfect information.

Traditionally there are two approaches for analyzing and improving supply chain networks and the material flow through them: (1) modelling the supply chain as an LP or MIP and applying either exact methods or heuristics to find “good” solutions; (2) using a simulation based approach to analyze the complex behaviour of a supply chain and try improving the current situation based on a trial-and-error principle.

The first approach is well known and is often discussed in the literature. Pankaj and Fisher [13] showed that based on an MIP model the coordination of production and distribution can reduce the operating cost substantially. Erengüç, Simpson, and Vakharia [4] summarized the developments on LP modelling of the production and distribution planning in a supply chain, whereas Owen and Daskin [12] gave an overview on facility location problems. In general the problems solved with LPs and MIPs are usually very restricted in order to keep them solvable. For example, Tsiakis, Shah, and Pantelides [22] did some work on designing supply chain networks, where manufacturing and customer zones are already fixed, and only warehouses and distribution centres are of unknown locations. Dogan and Goetschalckx [2] showed that larger design problems can be solved using decomposition. Other papers concentrate on the transportation aspect as Vidal and Goetschalckx [23].

In the field of supply chain simulation Kleijnen [8] gives a short overview of simulation tools and techniques used for supply chains. He distinguishes between four different approaches: spreadsheet simulation, system dynamics, discrete-event dynamic systems simulation, and business games. Clearly, discrete-event simulation is the most powerful tool to consider complex stochastic systems. For discrete-event simulation numerous software packages are available, both very specialized ones for a specific part of the supply chain and general ones with a high functionality in modelling and visualization of supply chains (cf. [7], [9]). Most of today’s simulators include

possibilities to do a black-box parameter optimization of a simulation model. But Swisher et al. [19] and Fu [6] show in their papers, that so far implemented methods are very rudimentary. In research much better optimization methods for a stochastic environment are known, but yet not included in the software packages.

The point we are focusing on is to investigate the possibilities of combining optimization and simulation in the field of network flow optimization. The idea is to iteratively exchange data between a simulation model and the optimization model, in order to be able to consider nonlinearities within the model and combine the advantages of discrete-event simulation and optimization via linear programming.

There are a few papers on optimization of network flows in the context of supply chain management simulation. Yaged [24] discusses in his paper a static network model which includes nonlinearities. He tries to optimize the flow by solving a linearized version of the network and improve the flow in the network. Paraschis [14] discusses several different possibilities to linearize such networks and Fleischmann [5] presents several applications of network flow models, which are solved through linearization. But all three papers do not include any stochastic elements. Lee and Kim [10] show a real combination of simulation and optimization for the case of a production-distribution system. They use simulation to check the result of the simpler optimization model in a more realistic environment and to update the parameters for the optimization. After several iterations they end up with a solution of the optimization model which is also within the constraints of the stochastic simulation model. Truong and Azadivar [21] developed an environment for solving supply chain design problems, where they combine simulation with genetic algorithms and mixed-integer programs. But they remain on a strategic level with the questions of facility location and partner selection.

In our paper we want to present a general LP-Model for supply chain networks including also a multi-objective version. Furthermore we demonstrate how simulation and LP/MIP optimization can be combined on the operational level. We do not use the optimization on top of the simulation (optimizing parameters), but we include simulation and optimization in an iterative process in order to gain the advantages of optimization (exact solution) and simulation (nonlinearities, complex structure) [15].

The supply chain is represented as a discrete-event model (D-E model) and a linearized version of it is modelled as an LP. Due to the structure of the LP model, it is possible to consider only specific subsystems of the supply chain. At first we perform several simulation runs including all stochastics in order to get average values of the parameters (e.g. unit

transportation costs) which are then fed into the LP model. Now we can solve the LP. The result is transformed into decision rules and is used in the discrete-event model. Then we start again with further simulation experiments (see Figure 1).

The paper is organized as follows: In Section II we describe the underlying LP-Model. The multi-objective solution method is described in Section III. Section IV contains a small example for combining the LP model with a simulation model. We conclude the paper and give an outlook for further research in Section V.

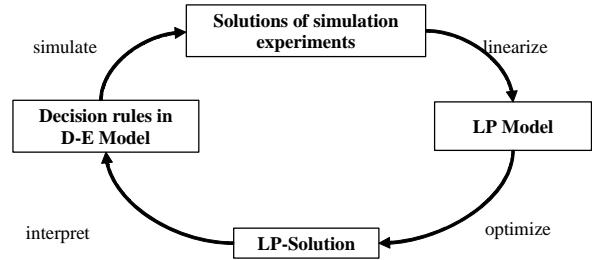


Fig. 1. Interaction between Simulation and Optimization

## II. THE MODEL

In order to facilitate the understanding of the following sections we now want to give a brief description of the underlying LP model. A detailed description of the whole model has already been presented in [15].

The basis for our supply chain model is a predefined network, i.e., the locations of all actors and the distances between them are given. Within the network we differentiate between three types of participants: (i) suppliers providing raw materials; (ii) customers who demand certain products at a specific time; (iii) intermediate nodes where production, stocking, and transshipment takes place. Additionally we consider different transportation modes and different products. We distinguish between 4 different categories of decision variables to account for transportation, production, transshipment, and inventory decisions. Therefore it is necessary to determine for every period the amount of each product, that has to be transported from one location to another and which of the transportation modes should be used. The intermediate nodes either tranship the incoming products or use them as raw materials for manufacturing other goods. At every location we may keep products on stock, intermediate nodes actually have two inventories (inbound and outbound). The decisions concerning the optimal inventory levels are included in the model.

The supply chain consists of  $J$  locations, which can be separated into three subsets, i.e the suppliers  $J^S$ , the intermediates  $J^0$  and the customers  $J^C$ . Furthermore the model includes a set of products  $P$  and a set of transportation modes  $V$ .  $T$  indicates the number

of periods. The model implies three types of decision variables:  ${}^v x_{ij}^p(t)$ ,  $m_i^p(t)$  and  $u_i^p(t)$ . Each of them can be assigned to one subproblem, i.e. transportation, production and transshipment. The problem is formulated as follows (for a detailed explanation of the variables and the notation see the Appendix):

$$\begin{aligned} \min & \sum_{ij \in J} \sum_{p \in P} \sum_{t=1..T} \sum_{v \in V} {}^v c_{ij}^p \cdot {}^v x_{ij}^p(t) + \sum_{i \in J^0} \sum_{p \in P} \sum_{t=1..T} w_i^p \cdot m_i^p(t) \\ & + \sum_{i \in J^0} \sum_{p \in P} \sum_{t=1..T} z_i^p \cdot u_i^p(t) + \sum_{i \in J^0} \sum_{p \in P} \sum_{t=1..T} {}^{in} h_i^p \cdot {}^{in} l_i^p(t) \\ & + \sum_{i \in J^0 \cup J^s} \sum_{p \in P} \sum_{t=1..T} {}^{out} h_i^p \cdot {}^{out} l_i^p(t) + \sum_{i \in J^c} \sum_{p \in P} \sum_{t=1..T} \rho_i^p \cdot (-{}^{in} l_i^p(t)) \end{aligned} \quad (1)$$

$$\sum_{p \in P} {}^v g^p \cdot {}^v x_{ij}^p(t) \leq {}^v C_{ij}(t) \quad (2)$$

$${}^v x_{ij}^p(t) \leq {}^v Cap_{ij}^p(t) \quad (3)$$

$$\sum_{p \in P} a_i^p \cdot m_i^p(t) \leq {}^{prod} C_i(t) \quad (4)$$

$$m_i^p(t) \leq {}^{prod} Cap_i^p(t) \quad (5)$$

$$\sum_{p \in P} d_i^p \cdot u_i^p(t) \leq {}^{ta} C_i^p(t) \quad (6)$$

$$u_i^p(t) \leq {}^{ta} Cap_i^p(t) \quad (7)$$

$${}^{out} l_i^p(t) \geq 0 \quad \forall i \in J^s \quad (8)$$

$${}^{in} l_i^p(t) \geq 0 \quad \forall i \in J^0 \quad (9)$$

$${}^{out} l_i^p(t) \geq 0 \quad \forall i \in J^0 \quad (10)$$

$${}^{in} l_i^p(t) \leq 0 \quad \forall i \in J^c \quad (11)$$

$$\sum_{p \in P} q_i^p \cdot {}^{in} l_i^p(t) \leq {}^{in} L_i(t) \quad \forall i \in J^0 \quad (12)$$

$${}^{in} l_i^p(t) \leq {}^{inv} Cap_i^p(t) \quad \forall i \in J^0 \quad (13)$$

$$\sum_{p \in P} q_i^p \cdot {}^{out} l_i^p(t) \leq {}^{out} L_i(t) \quad \forall i \in J^0 \quad (14)$$

$${}^{out} l_i^p(t) \leq {}^{invout} Cap_i^p(t) \quad \forall i \in J^0 \quad (15)$$

$${}^{in} f_j^p(t) = \sum_{\substack{i \in J \\ {}^v \tau_{ij} < t}} \sum_{v \in V} {}^v x_{ij}^p(t - {}^v \tau_{ij}) \quad \forall j \in J^0 \cup J^c \quad (16)$$

$${}^{out} f_j^p(t) = \sum_{i \in J} \sum_{v \in V} {}^v x_{ji}^p(t) \quad \forall j \in J^s \cup J^0 \quad (17)$$

$${}^{out} l_i^p(t) = {}^{out} l_i^p(t-1) - {}^{out} f_i^p(t) + S_i^p(t) \quad \forall i \in J^s \quad (18)$$

$$\begin{aligned} {}^{in} l_i^p(t) &= {}^{in} l_i^p(t-1) + {}^{in} f_i^p(t) \\ &- \sum_{p' \in P} \alpha_i^p(p') \cdot m_i^{p'}(t) - u_i^p(t) + r_i^p(t) \quad \forall i \in J^0 \end{aligned} \quad (19)$$

$$\begin{aligned} {}^{out} l_i^p(t) &= {}^{out} l_i^p(t-1) - {}^{out} f_i^p(t) + \chi_{t \geq \delta_i^p} \cdot m_i^p(t - \delta_i^p) \\ &+ \chi_{t \geq \sigma_i^p} \cdot u_i^p(t - \sigma_i^p) + b_i^p(t) \quad \forall i \in J^0 \end{aligned}, \quad (20)$$

$$\chi_{t \geq \varepsilon} = \begin{cases} 1 & t \geq \varepsilon \\ 0 & t < \varepsilon \end{cases}$$

$${}^{in} l_i^p(t) = {}^{in} l_i^p(t-1) + {}^{in} f_i^p(t) - D_i^p(t) + r_i^p(t) \quad \forall i \in J^c \quad (21)$$

$${}^v x_{ij}^p(t) \geq 0 \quad (22)$$

$$m_i^p(t) \geq 0 \quad (23)$$

$$u_i^p(t) \geq 0 \quad (24)$$

The overall goal, which is represented by the objective function (1), is to minimize the total costs. In principle, there are four types of costs: transportation cost, production cost, transshipment cost, and inventory cost.

The objective function is minimized with respect to several constraints :

Constraints (2) and (3) ensure that given transport capacities for both the individual and the aggregated case are not exceeded. Constraints (4) and (6) are similar to the capacity constraints. They state, that given production capacities, and given transshipment capacities are not exceeded. Constraints (5) and (7) impose individual upper bounds on transshipment or production at a certain intermediate. Constraints (8), (9) and (10) ensure the nonnegativity of all inventory levels of suppliers and intermediates. The inbound inventory level at customer nodes can only take negative values. This restriction is given in constraint (11). The inbound and outbound inventories at the intermediates are furthermore capacitated. These capacities are considered in constraints (12) and (14). Additionally there are capacity restrictions for each product itself, i.e. restrictions (13) and (15).

In order to calculate the total inflow or outflow at every node, auxiliary equations (16) and (17) have been included. Equations (18) to (21) represent the inventory balance equations for each node. The nonnegativity of the decision variables is ensured in constraints (22) to (24).

The first implementation of the LP model has been done in Xpress/MP, the LP solver of Dash Optimization<sup>1</sup>. Since the goal is to couple the optimization with the simulation in an iterative procedure, it is important to ensure that the optimization algorithm does not exceed a certain time limit. In order to find the limitation of the implementation regarding the problem size, we generated several different sized test instances. The results of these tests have been published in [15]. For example, a problem with 3 suppliers, 29 intermediate nodes, 2 customers, 2 transportation modes, 12 products and 40 periods has been solved in 29.64 seconds.

<sup>1</sup> www.dashoptimization.com

### III. MULTICRITERIA LINEAR PROGRAMMING

In a further step we want to distinguish between (i) production, transportation and inventory costs, which can easily be measured in monetary units; and (ii) penalty costs where it is quite difficult to assign a cost factor. Hence, the objectives are:

$$\begin{aligned} Z_1 &= \sum_{ij \in J} \sum_{p \in P} \sum_{t=1..T} \sum_{v \in V} {}^v c_{ij}^p \cdot {}^v x_{ij}^p(t) + \sum_{i \in J^0} \sum_{p \in P} \sum_{t=1..T} w_i^p \cdot m_i^p(t) \\ &\quad + \sum_{i \in J^0} \sum_{p \in P} \sum_{t=1..T} z_i^p \cdot u_i^p(t) + \sum_{i \in J^0} \sum_{p \in P} \sum_{t=1..T} {}^{in} h_i^p \cdot {}^{in} l_i^p(t) \\ &\quad + \sum_{i \in J^0 \cup J^c} \sum_{p \in P} \sum_{t=1..T} {}^{out} h_i^p \cdot {}^{out} l_i^p(t) \\ Z_2 &= \sum_{i \in J^c} \sum_{p \in P} \sum_{t=1..T} \rho_i^p \cdot (-{}^{in} l_i^p(t)) \\ Z_3 &= \lambda Z_1 + (1 - \lambda) Z_2, \end{aligned}$$

where  $\lambda \in [0,1]$  is a weight factor measuring the importance of the different objectives.

This 2-objective LP model has been implemented on the basis of an algorithm on parametric linear programming by Ehrgott [3]. Applying this algorithm it is possible to identify those values of the weights  $\lambda$ , which lead to a change of the basis and therefore to a possible change of the solution. The goal is to solve the problem with respect to  $Z_3$  using a specific weight  $\lambda$ . Based on the resulting basic feasible solution the reduced costs with respect to  $Z_1$  and  $Z_2$  are calculated and are used to determine the next threshold for  $\lambda$ , i.e. the point where the next change in the basis will occur.

Due to the fact that Xpress/MP does not include features for solving multicriteria optimization problems, we had to use a special combination of Xpress/MP functions in order to get the reduced costs for both objectives. The algorithm has been implemented as follows:

```

 $\lambda := 1$ 
while ( $\lambda > 0$ ) do
    minimize  $Z_3$ 
    calculate reduced costs  $c^1$  of  $Z_1$ 
    with respect to current basis
    (apply primal simplex with 0
    iterations)
    calculate reduced costs  $c^2$  of  $Z_2$ 
    with respect to current basis
    (apply primal simplex with 0
    iterations)
    if (all  $c^2 \geq 0$ )
         $\lambda := 0$ 
    else
         $\lambda := \max\left(\frac{-c_j^2}{c_j^1 - c_j^2}, \forall c^2_j < 0, c^1_j \geq 0\right)$ 
    end-if
end-do

```

As soon as the reduced costs for objective 2 are all positive, the algorithm stops. There will be no further change in the basis, i.e. the current optimal solution will stay the same, even if  $\lambda$  is set to 0.

For the validation of the implementation we generated a small test instance. The network consists of 3 suppliers, 3 intermediates and 3 customers, who demand only one kind of product. The planning horizon consists of 3 periods. In each period, each of the suppliers produces 10 pieces of the products, which can be transported to the intermediates and further to the customers by two different transportation modes. The first one is quite fast and guarantees a delivery in time, whereas the second one will cause a delay in delivery, but it is 4 times cheaper than the first one. As already mentioned above, the decision maker has to decide between minimizing the transportation cost versus avoiding penalty costs. Table I shows the results for this small example.

TABLE I  
RESULTS OF THE TEST INSTANCE FOR THE MULTI-OBJECTIVE CASE

$\lambda$	$Z_1$	$Z_2$	$Z_3$
[1;0.909]	0	600	0
[0.909;0.476]	0	600	54.55
[0.476;0]	660	0	314.29

There are two changes in the basis, the first one as soon as  $\lambda$  gets the value 0.909 and the second one at the value 0.476. The latter one is the point where the decision maker for the first time chooses transportation mode 1 instead of transportation mode 2. At the value 0.909 there is also a change in the basis, but since the values of both objective functions stay the same, the change of the basis consists of an exchange of slack variables. As long as  $\lambda$  is greater than 0.476, the optimal solution for the decision maker is to use the slow transportation mode, because it is more important to have low transportation costs than to avoid penalty costs.

### IV. INTERACTION BETWEEN SIMULATION AND OPTIMIZATION

In this section we want to explain in more detail the interaction between the simulation model, which displays the structure of the supply chain network, and the optimization model, which is a linearization of the previous. The cycle is visualized in Figure 2.

The simulation model includes the complexity of the given supply chain network, considering nonlinearities and stochastics. This discrete-event model should be close to reality as far as possible, respectively necessary, but it should be implemented in a way that facilitates the transformation into a linearized version

of the model.

The interface between the simulation model and the LP-Model will be a data file in Xpress/MP format, including all parameters for the LP-Model. At the beginning of the cycle two things have to be delivered from the simulation model to the LP-Model: the structure of the model and the starting values for all parameters used in the model.

For the latter one there are two possibilities to obtain the relevant information: (i) the values for the parameters are determined by taking the average values after the first simulation runs; (ii) a couple of simulation runs with different assumptions are implemented, and further used for an estimation of the initial parameter values. In the following iterations the parameters can in either case be estimated based on the solutions of the former simulation runs.

The data transfer from the LP-Model to the simulation model is done via another data file. These data contain information about: (i) the optimal solution of the problem; (ii) the solution of the sensitivity analysis; (iii) the paretofront (in case of a multicriteria

The comparison addresses a discrete modelling and simulation problem. It considers a relatively simple network, consisting of four factories, four distributors and a customer. Some parameters of the original definition of the comparison have been changed slightly in order to apply optimization and to meet the requirements of the LP model.

The time horizon considered consists of 10 days (240 hours). We assume that the factories produce 12 different products, whereas each factory produces only 6 of them.

Factories F1 and F3 supply products P1 to P6, and factories F2 and F4 supply products P7 to P12. The amounts of products supplied at each factory have been generated randomly. The 4 distributors order products at the suppliers and deliver them to the customer.

In the simulation model the distributors order the products following an ordering plan, which is at first determined in the optimization model and afterwards fed into the simulation. Both the suppliers and the distributors are provided with an initial inventory level for each product. Furthermore, the network is actually

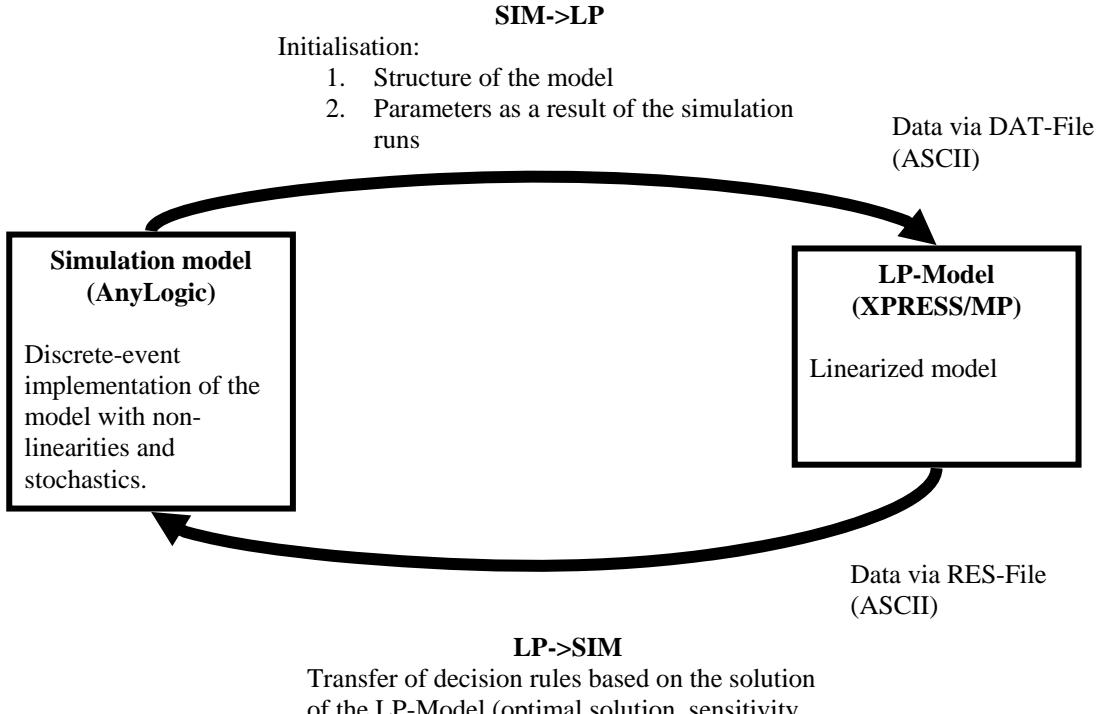


Fig. 2. Interaction between Simulation and Optimization, including information on the data exchange

decision problem). All this information is used to determine decision rules for the simulation.

This approach has been already tested using a small example. The basis for our experiments is Comparison 14, a simulation reference model published by the ARGESIM working group in the SNE (Simulation News Europe, cf. [20]). The simulation model has been implemented in AnyLogic, the simulation tool from XJ Technologies.

separated into two parts. Distributors D1 and D2 can only order products at factories F1 and F2, and distributors D3 and D4 order their products only at factories F3 and F4. Transporting the products from a supplier to a distributor takes some periods of time.

The customer orders products from the distributors, the transportation time between factories and distributors is not taken into account within the model.

The deliveries between factories and distributors can

only take place once a day, i.e. in every 24th period. The inventory levels are always measured at the end of each period.

Therefore, the total costs of the system consist of the inventory costs of the suppliers, the inventory costs of the distributors and the transportation costs between suppliers and distributors. The inventory costs for suppliers and distributors are calculated once a day based on the stock level just before the distributors place their orders. Without taken into account any stochastics, the goal of this experiment was to find the optimal solution for the whole network. For the inventory cost of the factories we use a piecewise linear function, and for the distributors' inventory cost we tested 3 different costs functions: (i) a piecewise linear convex function; (ii) a concave function, and (iii) a piecewise linear step function.

The above cost functions are used only in the simulation model. For the optimization model we linearized these functions.

With all three types of cost functions for the distributors' inventories we performed our algorithm in the following form: After having simulated the flow of products within the system for the complete time horizon, the average inventory costs at the different nodes were computed and incorporated into the optimization model. Clearly, the updated inventory costs led to an adjustment concerning the transportation amounts. For the piecewise linear convex function three iterations of this ping-pong game had to be performed, until the simulation model provided the same average inventory costs as in the previous round.

For the second experiment we used a concave cost function for the distributors' inventories. Again we get the same total cost values from simulation and optimization after 3 iterations.

Although we can gain convergence in these first two examples, we do not know if we are trapped in a local minimum or if we have found the global optimum. Especially if we consider more complex networks with different types of nonlinearities, it will be very difficult to find some general conditions under which we can

guarantee to find a global optimum.

In the case of the step function we cannot reach convergence. After the fifth iteration we get the same results as in the third iteration. Here the problem arises that it is not possible to guarantee convergence in the general case. (For more details of the results see [15]).

In the next example we include a stochastic element. We assume that the time needed for transporting products from the factory to the distributor has a constant part (which is four fifth of the value in the previous examples) and a stochastic part, where we use an exponential distribution (the average transportation time does not change compared to the deterministic case).

The transportation costs are independent of the transportation time. We allow backorders at the distributors, but they are penalized with 100 monetary units per unit short per period. For the inventory costs we consider the same situation as in the first case, i.e. piecewise linear, continuous cost functions at the factory and at the distributor. Now for each iteration we do 5 simulation runs and use the average results as input to the LP-Model. The results are shown in Table II.

We see that after 3 iterations the changes of the parameters are less than 1%, but the difference of the total costs are varying between 0.75% in the third iteration and 2.23% in the fourth iteration. If we perform 5 simulation runs instead of 20 simulation runs in the last iteration the gap is only 1.29% (value given in parentheses). So in the stochastic case, we cannot expect to close this gap completely, but we have to analyze the simulation results in more detail and determine a threshold for the gap between the results of the LP-Model and the simulation model.

## V. CONCLUSIONS AND OUTLOOK

Analyzing and improving supply chains are difficult problems and several different approaches have been proposed in the literature. We presented a new possibility to combine the two main tools, namely simulation and optimization. The aim is to put the

TABLE II  
RESULTS OF THE EXAMPLE WITH A STOCHASTIC ELEMENT: UNIT INVENTORY COSTS

<b>It.</b>	<b>F1</b>	<b>F2</b>	<b>F3</b>	<b>F4</b>	<b>D1</b>	<b>D2</b>	<b>D3</b>	<b>D4</b>	<b>LP</b>	<b>Total costs</b>
1	1.5	1.5	1.5	1.5	2.5	2.5	2.5	2.5	16195	17978
2	3.44	3.29	3.23	3.19	1.00	1.03	1.02	1.01	17159	16672
3	3.39	3.11	3.04	2.96	1.02	1.04	1.05	1.03	16993	16867
4	3.39	3.11	3.04	2.96	1.02	1.04	1.05	1.03	16992	16621 (16776)

advantages of simulation (complex models, nonlinearities) and the advantages of optimization (exact solution) into one method. Within this framework it is possible to test the optimal deterministic solution of the LP-Model in a stochastic environment and to analyze whether this solution is also feasible in the context of a more complex and more realistic simulation model.

The underlying LP-Model is a very general representation of a supply chain, which ensures that it can be applied easily to a large variety of problems. Due to its structure, it is possible to restrict the optimization to parts of the supply chain.

In this paper we presented only some small examples for the combination of simulation and optimization and there are still some open research questions. Especially the example in Section IV, where we did not gain convergence, shows that a special effort has to be undertaken to find out the limitations of this method and maybe to develop further solution methods. Furthermore it is necessary to investigate in more detail, if in the case of convergence we really find the optimal solution. Another possibility might be to apply metaheuristics using the results of our approach as initial solution in order to leave a local optimum and to further improve the solution.

Our further work will improve the interfaces between simulation and optimization on both sides: (i) how can we automatically derive a linearized model out of the discrete-event model and (ii) how can we interpret the solution of the LP model as general decision rules in the simulation model? For the second question it is important to perform analytical investigations of the LP-Model, i.e. sensitivity analysis and multi-objective optimization. The multi-objective optimization can be used as a special extension of the sensitivity analysis regarding the changes of the weights. This information should be used to derive decision rules for the discrete-event model.

In the literature of production/distribution planning there are several examples of complex LP/MIP-Models, and sophisticated methods (e.g. Langrangian relaxation, decomposition) for simplifying and solving such problems available (e.g. [17]). So another possibility for further research is to increase the complexity of the underlying LP-Model. But this would imply a possible loss of generality and more complications for the interface to the simulation model.

## APPENDIX

### Notation used for the LP-Model:

$J$  set of locations

$J = J^s \cup J^o \cup J^c$

$j \in J^s$  raw-material supplier (starting nodes)

$j \in J^c$	customer (end nodes)
$j \in J^o$	nodes between supplier and customer
$P$	set of products
$V$	set of transportation modes
$T$	number of periods
${}^v x_{ij}^p(t)$	flow of product $p$ from location $i$ to location $j$ with transportation mode $v$ (sent away in period $t$ )
$m_i^p(t)$	amount of product $p$ (product $p$ is the end product of the production process at location $i$ ), that starts to be produced at location $i$ in period $t$
$u_i^p(t)$	amount of product $p$ , that starts to be transacted in location $i$ in period time $t$
${}^{in} f_j^p(t)$	amount of product $p$ , arriving at location $j$ in period $t$
${}^{out} f_j^p(t)$	amount of product $p$ , sent away at location $j$ in period $t$
${}^{in} l_i^p(t)$	inbound inventory level of product $p$ at location $i$ in period $t$
${}^{out} l_i^p(t)$	outbound inventory level of product $p$ at location $i$ in period $t$
$S_j^p(t)$	supply of product $p$ at location $j$ in period $t$
${}^{in} h_i^p$	inbound inventory costs (per unit) for product $p$ at location $i$
${}^{out} h_i^p$	outbound inventory costs (per unit) for product $p$ at location $i$
${}^{in} L_i(t)$	maximum capacity of inbound inventory at location $i$
${}^{out} L_i(t)$	maximum capacity of outbound inventory at location $i$
${}^{inv} Cap^p_i(t)$	maximum amount of product $p$ that can be held in the inbound inventory of intermediate $i$ in period $t$
${}^{invout} Cap^p_i(t)$	maximum amount of product $p$ that can be held in the outbound inventory of intermediate $i$ in period $t$
${}^v c_{ij}^p$	transportation costs (per unit) of product $p$ transported from location $i$ to location $j$ with transportation mode $v$
${}^v C_{ij}(t)$	maximum transportation capacity of transportation mode $v$ on the way from location $i$ to location $j$
${}^{prod} C_i(t)$	maximum production capacity at location $i$ in period $t$
${}^{ta} C_i^p(t)$	maximum transaction capacity at location $i$ in period $t$
${}^v Cap^p_{ij}(t)$	amount of product $p$ that transportation mode

	$v$ can transport from location $i$ to location $j$
$prod\ Cap^p_i(t)$	amount of product $p$ that can be produced at location $i$ in period $t$
$ta\ Cap^p_i(t)$	amount of product $p$ that can be transacted at location $i$ in period $t$
$\delta_i^p$	amount of periods required to produce product $p$ at location $i$
$\sigma_i^p$	amount of periods required to transact product $p$ at location $i$
$\alpha_i^p(p')$	amount of product $p'$ required to produce one unit of product $p$ at location $i$
$w_i^p$	production costs of product $p$ at location $i$
$z_i^p$	transaction costs of product $p$ at location $i$
$a_i^p$	factor, indicating the amount of capacity units required to produce one unit of product $p$ at location $i$
$b_i^p(t)$	amount of product $p$ , which is already in production process in period 0, and will be finished in period $t$ (or external increase of inventory)
$d_i^p$	factor, indicating the amount of capacity units required to transact one unit of product $p$ at location $i$
${}^v g^p$	factor, indicating the amount of capacity units required to transport one unit of product $p$ with transportation mode $v$
$q_i^p$	factor, indicating the amount of capacity units required to hold one unit of product $p$ at the inventory of location $i$
${}^v \tau_{ij}$	amount of periods required by transportation mode $v$ requires to go from location $i$ to location $j$
$\rho_i^p$	penalty costs (per unit) at location $i$
$D_i^p(t)$	demand for product $p$ at location $i$ in period $t$
$r_i^p(t)$	amount of product $p$ , which is already transported at period 0, and will arrive at location $i$ in period $t$ (or external increase of inventory)

#### ACKNOWLEDGMENT

We gratefully acknowledge the support of ARC Seibersdorf research and of Dash Optimization.

#### REFERENCES

- [1] Arbib C, Marinelli F (2005), Integrating process optimization and inventory planning in cutting stock with skiving option: An optimization model and its application. European Journal of Operational Research 163: 617-630.
- [2] Dogan K, Goetschalckx M (1999), A primal decomposition method for the integrated design of multi-period production-distribution systems. IIE Transactions 31: 1027-1036.
- [3] Ehrgott, M (2000), Multicriteria Optimization, Lecture Notes in Economics and Mathematical Systems 491, Springer-Verlag.
- [4] Erengüç SS, Simpson NC, Vakharia AJ (1999), Integrated production/distribution planning in supply chains: an invited review. European Journal of Operational Research 115: 219-236.
- [5] Fleischmann, B. (1993), Designing distribution systems with transport economies of scale. European Journal of Operational Research 70, 31-42.
- [6] Fu MC (2002), Optimization for Simulation: theory vs practice. INFORMS Journal on Computing 14: 192-215.
- [7] Kelton WD, Sadowski RP, Sadowski DA (2002), Simulation with Arena, 2nd edn. McGraw-Hill.
- [8] Kleijnen, J. P. C., (2005), Supply chain simulation tools and techniques: a survey, International Journal of Simulation & Process Modelling 1 (1-2) pp. 82—89.
- [9] Kuhn A, Rabe M (1998), Simulation in Produktion und Logistik (Fallbeispieleammlung),Springer.
- [10] Lee, Y.H., Kim, S.H. (2000), Optimal production-distribution planning in supply chain management using hybrid simulation-analytic approach. Proceedings of the 2000 Winter Simulation Conference (Joines, J.A., Barton, R.R., Kang, K., Fishwick, P.A., eds.), 1252-1259.
- [11] Meyr, H. (2002), Simultaneous lotsizing and scheduling on parallel machines. European Journal of Operational Research 139, 277-292.
- [12] Owen SH, Daskin MS (1998), Strategic Facility Location: A Review. European Journal of Operational Research 111: 423-447.
- [13] Pankaj C, Fisher ML (1994), Coordination of production and distribution planning. European Journal of Operational Research 72: 503-517.
- [14] Paraschis, I.N. (1989), Optimale Gestaltung von Mehrproduktssystemen. Physica-Schriften zur Betriebswirtschaft 26, Physica-Verlag Heidelberg.
- [15] Preusser, M., Almeder, C., Hartl, R.F., Klug, M. (2005), LP Modelling and simulation of supply chain networks. In: Günther, H.O., Mattfeld, D.C., Suhl, L., Supply Chain Management und Logistik: Optimierung Simulation, Decision Support. Physica-Verlag, 95-114.
- [16] Schneeweiss C, (2003), Distributed Decision Making, 2nd edn. Springer.
- [17] Shapiro JF (1993), Mathematical Programming Models and Methods for Production Planning and Scheduling, In: Graves SC, Rinnooy Kan AHC, Zipkin PH (eds) Handbooks in Operations Research and Management Science: Logistics of Production and Inventory. North-Holland.
- [18] Stadtler H (2005), Supply chain management and advanced planning – basics, overview and challenges. European Journal of Operational Research 163: 575-588.
- [19] Swisher, J.R., Jacobson, S.H., Hyden, P.D., Schruben, L.W. (2000), A survey of simulation and optimization techniques and procedures. Proceedings of the 2000 Winter Simulation Conference, Joines, J.A., Barton, R.R., Kang, K., and Fishwick, P.A., (eds.), pp.119-128.
- [20] Tauböck SM (2001), C14 supply chain management definition. SNE Simulation News Europe 32/33: 28-29.
- [21] Truong TH, Azadivar F (2003), Simulation based optimization for supply chain configuration design. Proceedings of the 2003 Winter Simulation Conference (S. Chick, P. J. Sánchez, D. Ferrin, and D. J. Morrice, eds.) pp. 1268-1275.
- [22] Tsiakis P, Shah N, Pantelides CC (2001), Design of multi-echelon supply chains under demand uncertainty. Ind. Eng. Chem. Res. 40, 3585-3604.
- [23] Vidal CJ, Goetschalckx M (2001), A global supply chain model with transfer pricing and transportation cost allocation. European Journal of Operational Research 129: 134-158.
- [24] Yaged, B. (1971), Minimum cost routing for static network models. Networks 1, 139-172.

# Tolerance-based Branch and Bound Algorithms

Marcel Turkensteen\*, Diptesh Ghosh†, Boris Goldengorin‡ and Gerard Sierksma§

\*University of Groningen, Fac. of Economics

Landleven 5, PO Box 800, 9700 AV Groningen, Netherlands

Email: m.turkensteen@eco.rug.nl

†Indian Institute of Management, P&QM Area

Ahemdabad 380015, Ahmedabad, India

Email: diptesh@iimahd.ernet.in

‡University of Groningen, Fac. of Economics

Landleven 5, PO Box 800, 9700 AV Groningen, Netherlands

Email: b.goldengorin@eco.rug.nl

§University of Groningen, Fac. of Economics

Landleven 5, PO Box 800, 9700 AV Groningen, Netherlands

Email: g.sierksma@eco.rug.nl

**Abstract**—The selection of entries to be included/excluded in Branch and Bound algorithms is usually done on the base of cost values. In this paper, we propose to use upper tolerances to guide the search for optimal solutions. In spite of the fact that it needs time to calculate tolerances, our computational experiments for Asymmetric Traveling Salesman Problem show that in most situations tolerance-based algorithms outperform cost-based algorithms. The solution time reductions are mainly caused by the fact that the branching process becomes much more effective, so that optimal solutions are found in an earlier stage of the branching process. The use of tolerances also reveals why the widely used choice for branching on a smallest cycle in assignment solutions is on average the most effective one.

**Keywords**—Tolerances, NP-hard problems, Branch and Bound.

## I. INTRODUCTION

THE Traveling Salesman Problem (TSP) is the problem of finding a shortest tour through a given number of locations such that every location is visited exactly once. The cost of traveling from location  $i$  to location  $j$  is denoted by  $c(i, j)$ . These costs are called *symmetric* if  $c(i, j) = c(j, i)$  for each pair of cities  $i$  and  $j$ , and *asymmetric* otherwise. The fact that the TSP is a typical NP-hard optimization problem means, roughly spoken, that solving instances with a large number of cities is very difficult if not impossible. Recent developments in polyhedral theory and heuristics have significantly increased the size of instances which can be solved to optimality. The best known exact algorithms are based

on either the Branch and Bound (BnB) method for the Asymmetric TSP (ATSP) [5] or the Branch and Cut method for the Symmetric TSP (STSP) using the double index formulation of the problem (see [18]). A state-of-the-art account on heuristics for the STSP and the ATSP is presented in [11] and [12], respectively.

Currently, most algorithms for the TSP delete high cost arcs or edges and save the low cost ones. A drawback of this strategy is that costs of arcs and edges are no accurate indicators whether those arcs or edges are saved in an optimal TSP solution. In this paper, it is shown that *tolerances* are better indicators. A tolerance value of an edge/arc is the cost of excluding or including that edge/arc from the solution at hand; see Section II. Although the concept of tolerances has been applied for decades (in sensitivity analysis; see for example [16]), only Helsgaun's version of the Lin-Kernighan heuristic for the STSP applies tolerances; see [8].

We apply upper tolerances in BnB algorithms for the ATSP. A BnB algorithm initially solves a *relaxation* of the original hard problem. In case of the ATSP, the Assignment Problem (AP) is a common choice. The AP is the problem of assigning  $n$  people to  $n$  jobs against minimum cost; an optimal solution of the AP is called a *minimum cycle cover*. If the minimum cycle cover at hand is a complete tour, then the ATSP instance is solved; otherwise, the problem is partitioned into new subproblems by including and excluding arcs. In the course of the process, a *search tree* is generated in which all solved subproblems are listed. BnB algorithms comprise two major steps: *branching* and *bounding*.

The objective of *branching* is to find a good, or even

optimal, ATSP solution in an effective way. If the current AP solution is infeasible, then there may exist a subset of elements of this solution, the so-called *survival set*, which also appears in an optimal TSP solution eventually obtained by the BnB algorithm. An effective BnB algorithm cherishes arcs in survival sets and disposes of the other ones, the *extinction arcs*. Obviously, survival sets are not known beforehand. Predictions of what arcs belong to the survival set are usually based on the arc costs. We claim that the predictions are much more accurate if upper tolerance values of arcs are used instead; see Section III.

The objective of *bounding* is to fathom as many nodes in the search tree as possible. A subproblem is fathomed if its lower bound exceeds the value of the best solution found so far in the process. An AP solution is infeasible for the ATSP, if it consists of two or more subcycles; we call such subcycles *offenders*. To obtain a complete tour, at least one offender must be “broken”, meaning that its arcs are successively prohibited in the next stage of the process. Since the cost of removing an arc is its upper tolerance value, upper tolerance values provide us with the cost of breaking an offender, and hence, they can be used to tighten the lower bounds. The higher the lower bound, the larger set of subproblems that are fathomed; see Section IV.

Compared to their cost-based counterparts, tolerance-based BnB algorithms have one big drawback: whereas cost values need to be looked up, tolerance values must be calculated. So the question is whether the reduction in the size of the search tree is on average sufficiently large to compensate for the additional tolerance computation times. Computational experiments, performed in Section VI, show that it is so for random, sparse, and various ATSPLIB instances. The conclusions and future research directions appear in Section VII.

## II. TOLERANCES FOR COMBINATORIAL OPTIMIZATION PROBLEMS

In this section we introduce the notion of upper and lower tolerances in case of a general Combinatorial Optimization Problem (COP). A *Combinatorial Optimization Problem* COP( $\mathcal{E}, C, \mathcal{D}, f_C$ ) is the problem of finding a solution

$$S^* \in \arg \text{opt}\{f_C(S) \mid S \in \mathcal{D}\},$$

where  $C : \mathcal{E} \rightarrow \mathbb{R}$  is the given *instance* of the problem with *ground set*  $\mathcal{E}$  satisfying  $|\mathcal{E}| = m$  ( $m \geq 1$ ),  $\mathcal{D} \subseteq 2^\mathcal{E}$  is the *set of feasible solutions*, and  $f_C : 2^\mathcal{E} \rightarrow \mathbb{R}$  is the *objective function* of the problem.  $\mathcal{D}^* = \arg \text{opt}\{f_C(S) \mid S \in \mathcal{D}\}$  is the set of optimal solutions.

It is assumed that  $\mathcal{D}^* \neq \emptyset$ . In the remaining part of this paper we take  $\text{opt} = \min$ .

Let  $g \in \mathcal{E}$ , and  $\alpha \geq 0$ . By  $C_{\alpha,g} : \mathcal{E} \rightarrow \mathbb{R}$  we denote the instance defined as  $C_{\alpha,g}(e) = C(e)$  for each  $e \in \mathcal{E} \setminus \{g\}$ , and  $C_{\alpha,g}(g) = C(g) + \alpha$ . Take any  $S^* \in \mathcal{D}^*$ , and  $e \in \mathcal{E}$ . The *upper tolerance* of  $e$  with respect to  $S^*$  is denoted and defined as

$$u_{S^*}(e) = \max\{\alpha \geq 0 : S^* \in \arg \min\{f_{C_{\alpha,e}}(S) : S \in \mathcal{D}\}\},$$

and the *lower tolerance* of  $e$  with respect to  $S^*$  as

$$l_{S^*}(e) = \max\{\alpha \geq 0 : S^* \in \arg \min\{f_{C_{-\alpha,e}}(S) : S \in \mathcal{D}\}\}.$$

I.e.  $u_{S^*}(e)$  is the maximal increase of  $C(e)$  under which  $S^*$  stays optimal, and  $l_{S^*}(e)$  the maximal decrease of  $C(e)$  under which  $S^*$  stays optimal.

We assume that  $f_C$  is *monotone*, meaning that for each  $S \in 2^\mathcal{E}$  and each  $\alpha \geq 0$ , it holds that

$$f_{C_{\alpha,e}}(S) \geq f_{C_{0,e}}(S).$$

*Sum functions* with  $f_C(S) = \sum_{e \in S} C(e)$ , *bottleneck functions* with  $f_C(S) = \max_{e \in S} C(e)$ , and *product functions* with  $f_C(S) = \prod_{e \in S} C(e)$  and  $C(e) \geq 1$  for each  $e \in \mathcal{E}$  are all monotone functions.

We call the set  $\mathcal{D}$  of feasible solutions *non-embedded* if for each  $S_1, S_2 \in \mathcal{D}$  with  $S_1 \neq S_2$ , it holds that neither  $S_1 \subset S_2$  nor  $S_2 \subset S_1$ .

The following theorem, of which the proof is left to the reader, can be seen as a straightforward generalization of Libura's theorem on tolerances (see [15]) for the TSP. We will use the following extra notations. Let  $e \in \mathcal{E}$ . Then  $\mathcal{D}_+(e) = \{S \in \mathcal{D} : e \in S\}$ , and  $\mathcal{D}_-(e) = \{S \in \mathcal{D} : e \notin S\}$ . Clearly,  $\mathcal{D} = \mathcal{D}_-(e) \cup \mathcal{D}_+(e)$  and  $\mathcal{D}_-(e) \cap \mathcal{D}_+(e) = \emptyset$ .  $\mathcal{D}_+^*(e)$  and  $\mathcal{D}_-^*(e)$  are the sets of optimal solutions containing  $e$  and not containing  $e$ , respectively.

*Theorem 1:* Consider COP  $(\mathcal{E}, C, \mathcal{D}, f_C)$  with monotone  $f_C$ . For each  $S^* \in \mathcal{D}^*$ , the following holds:

1.  $e \in \cap \mathcal{D}^*$  iff  
 $u_{S^*}(e) = f_C(S) - f_C(S^*) > 0$   
for each  $S \in \mathcal{D}_-^*(e)$ ,  
 $l_{S^*}(e) = \infty$ ;
2.  $e \in \mathcal{E} \setminus \cup \mathcal{D}^*$  iff  
 $u_{S^*}(e) = \infty, l_{S^*}(e) = f_C(S) - f_C(S^*) > 0$   
for each  $S \in \mathcal{D}_+^*(e)$ ;
3.  $e \in S^* \setminus \cap \mathcal{D}^*$  iff  
 $u_{S^*}(e) = 0, l_{S^*}(e) = \infty$ ;
4.  $e \in \cup \mathcal{D}^* \setminus S^*$  iff  
 $u_{S^*}(e) = \infty, l_{S^*}(e) = 0$ .

If  $|\mathcal{D}^*| = 1$ , then this theorem boils down to Libura's theorem on tolerances. If  $\mathcal{D}_-(e) = \emptyset$  for some  $e \in \mathcal{E}$ , then  $u_{S^*}(e) = \min\{f_C(T) : T \in \mathcal{D}_-(e)\} - f_C(S^*) =$

$\min\{\emptyset\} = \infty$  (by definition). Similarly, for  $\mathcal{D}_+(e) = \emptyset$  we take  $l_{S^*}(e) = \infty$ .

The following statement can be derived from Theorem 1. If one excludes an element  $e$  from  $S^*$ , then the objective value of the new problem will be  $f_C(S^*) + u_{S^*}(e)$ . The same holds for the lower tolerance if the element  $e \in \mathcal{E} \setminus S^*$  is included. So a tolerance-based BnB algorithm knows the cost of including or excluding elements before it selects the element to branch on.

In the remainder of the article we concentrate on AP  $(\mathcal{E}, \mathcal{C}, \mathcal{A}, f_C)$  and ATSP  $(\mathcal{E}, \mathcal{C}, \mathcal{H}, f_C)$ . Note that  $\mathcal{H} \subseteq \mathcal{A}$ . We denote the sets of optimal solutions by  $\mathcal{A}^*$  and  $\mathcal{H}^*$ , respectively.

### III. SURVIVAL SETS

This section explores the *branching* step of BnB algorithms. The goal of branching is to find a good or even optimal solution in the fastest possible way. BnB methods generate sequences of steps in which parts of the solution at hand are included and excluded, until an optimal solution of the original problem is found. If a BnB algorithm predicts correctly which element to delete or to insert, then its search tree will be small. So it is important to predict survival sets accurately. Most algorithms base the predictions on cost values, but the question is: do predictions improve if they are based on upper tolerance values, and: how many survival arcs are there on average?

In our experiments we consider instances with varying degree of symmetry and degree of sparsity. The *degree of symmetry* is defined as the fraction of off-diagonal entries of the cost matrix  $\{c_{ij}\}$  that satisfy  $c_{ij} = c_{ji}$ ; the *degree of sparsity* is the percentage of arcs that is missing in an instance.

Table I shows that the average percentage of common arcs in corresponding AP and ATSP solutions varies between 40 and 80%. Similar investigations show that the Minimum 1-Trees and optimal STSP tours have between 70% and 80% of the edges in common [8].

Assume that we start with a fixed AP solution  $A^* \in \mathcal{A}^*$ , and that  $H^* \in \mathcal{H}^*$  is a fixed shortest complete tour of the same instance. We explore whether there are relationships between the cost values and the upper tolerance values of arcs and their appearance in  $H^*$ . These relationships are measured with the *adjusted Rand index*, which measures the relationship between two partitions; see [10], and with *logistic regression* [6]. The costs and the upper tolerances are continuous variables, and are both compared with the partition of  $A^*$  into survival and extinction arcs.

In the adjusted Rand index analysis [10], we create partitions based on upper tolerances and costs. First, the

TABLE I  
FRACTION OF SURVIVAL ARCS IN OPTIMAL AP AND ATSP SOLUTIONS

Instance type	Fraction of survival arcs
ATSPLIB	53.52%
Degree of symmetry 0.33	69.29%
Degree of symmetry 0.66	51.10%
Full symmetry	43.44%
Asymmetric random	80.49%
Degree of sparsity 50%	86.27%
Degree of sparsity 75%	84.23%
Degree of sparsity 90%	83.46%

arcs in a fixed optimal AP solution  $A^*$  are partitioned into two subsets:  $IN_1$  contains the survival arcs, and the subset  $IN_0$  contains the extinction arcs. Call this partition  $IN = \{IN_0, IN_1\}$ . We try to replicate  $IN$  with partitions  $C$  and  $U$  based on the costs and tolerance values of the arcs, respectively. Define  $C := \{C_0, C_1\}$ , where  $C_0 = \{e \in A^* : c(e) \geq c^*\}$ ,  $C_1 := \{e \in A^* : c(e) < c^*\}$ , and determine  $c^*$  in such a way that  $|C_0| = |IN_0|$ . Arcs are partitioned into a set of low cost arcs  $C_1$  and a class of high cost arcs  $C_0$ . If it is true that all high cost arcs are not in the given shortest tour, then the sets  $IN_0$  and  $C_0$  and the sets  $IN_1$  and  $C_1$  coincide and cost values lead to a perfect prediction. Similarly, define  $U = \{U_0, U_1\}$ , where  $U_0 := \{e \in A^* : u_{A^*}(e) < u^*\}$ ,  $U_1 := \{e \in A^* : u_{A^*}(e) \geq u^*\}$ , and determine  $u^*$  in such a way that  $|U_0| = |IN_0|$ .

The *adjusted Rand index* measures how similar the each of both partitions  $U$  and  $C$  are in comparison with the ideal partition into survival and extinction arcs  $IN$ . The more similar two partitions are, the higher the adjusted Rand index between both partitions is. An adjusted Rand index of 1 indicates that for each nonempty class  $A_i$  of partition  $A$ , there exists a class  $B_j$  of partition  $B$  such that  $A_i = B_j$ . The expected adjusted Rand index is 0, if both partitions assign objects to classes randomly having the original number of objects in each class [10].

The adjusted Rand indices between  $IN$  and  $C$  and between  $IN$  and  $U$  are shown in Table II. They are larger for the tolerance-based partitions  $U$  than for the cost-based partitions  $C$ , which shows that predictions are better if they are based on upper tolerance values.

The adjusted Rand index analysis makes splits in the data based on the cost and the upper tolerance values. It does not include the distances in cost or tolerance value of an arc from the split value. The following

TABLE II  
QUALITY OF THE PREDICTIONS USING UPPER TOLERANCES AND COSTS

Instance type	Adjusted Rand indices		$R^2$ of logit model	
	Tolerance	Cost	Tolerance	Cost
ATSPLIB	0.113	-0.003	0.112	0.035
Degree of symmetry 0.33	0.152	0.007	0.169	0.017
Degree of symmetry 0.66	0.188	0.028	0.132	0.015
Full symmetry	0.158	0.013	0.007	0.011
Asymmetric random	0.287	0.039	0.299	0.023
Sparsity 50%	0.361	0.017	0.407	0.015
Sparsity 75%	0.252	0.033	0.382	0.013
Sparsity 90%	0.219	0.032	0.342	0.017

problem arises. Suppose that an arc with a very high upper tolerance value is not in any shortest ATSP tour. Such an arc is very likely to be excluded already in an early stage of a tolerance-based branching process. If this event occurs, the predictions of upper tolerances should get a bad rating for this instance. The same holds for arcs with low cost values which are not in a shortest ATSP tour. We define the binary variable  $IN$ , ranging over all arcs  $e \in A^*$ , as follows:  $IN(e) = 1$  if  $e \in H^*$  and  $IN(e) = 0$  otherwise.

An appropriate method for estimating the relationship between a dependent binary variable and independent continuous variables is *logistic regression* [7]. Logistic regression is usually applied to explain or predict choices in choice modeling, for example the choice between buying and not buying a product [9]. Based on the values of the independent variables, probabilities  $\pi(e)$  are estimated of the event that the independent variable attains the value 1 for the observation  $e$ . The fit of a model is good if these probabilities  $\pi(e)$  are close to the actual observed values of the dependent variable.

A general measure to determine the fit of a logistic regression model, also called *logit model*, is the  $R^2$  for a *logit model*  $R_{\text{logit}}^2$  [6], which compares the predictive power of a logit model to the predictive power of a model without independent variables. If  $R_{\text{logit}}^2 = 1$ , then all the variance in the independent variable is explained and predictions are perfect. On the other hand, if  $R_{\text{logit}}^2 = 0$ , the independent variables in the model offer no information about the dependent variable.  $R_{\text{logit}}^2$  is similar to  $R^2$  in linear regression.

In order to analyze survival sets, we construct for each instance two separate logit models. In the cost-based model the dependent variable  $IN$  is explained by the cost values of the arcs in an assignment solution; the independent variable in the tolerance-based model

is formed by the upper tolerance values. The values of  $R_{\text{logit}}^2$  of both models are presented in Table II.

The values of  $R_{\text{logit}}^2$  are higher for the tolerance-based models, except for fully symmetric instances. These results confirm that predictions based on upper tolerance values are clearly better than predictions based on costs.

#### IV. NEW LOWER BOUNDS FOR THE ATSP

In this section, we use the upper tolerances of an optimal AP solution  $A^*$  to construct tight lower bounds for the corresponding ATSP. Recall that, if the lower bound of a subproblem is increased, then this subproblem is more likely to be fathomed during the execution of the BnB algorithm.

In BnB algorithms, lower bounds can be obtained by removing sources of infeasibility with respect to the original problem from the current solution. We called such sources of infeasibility *offenders*. In case of the ATSP and its AP relaxation, offenders are subcycles. Let  $A^*$  consist of  $k (> 1)$  cycles, say,  $A^* = \cup_{i=1}^k K_i$ . We define  $\mathcal{C}(A^*)$  as the set of all cycles in  $A^*$ , so  $\mathcal{C}(A^*) = \{K_1, \dots, K_k\}$ .

In order to “break” a subcycle  $K$  (meaning that this cycle does not appear in subsequent AP solutions), at least one arc must be removed. Recall that the cost of removing an arc is equal to its upper tolerance value. Hence, the minimum cost of breaking a subcycle is equal to the lowest upper tolerance value in that cycle. We denote and define for each  $K \in \mathcal{C}(A^*)$  this value by  $u_{A^*}^K := \min\{u_{A^*}(e) : e \in K\}$ . Theorem 2 shows that the cost of breaking a cycle by deleting a minimum tolerance arc can be used to increase the lower bound.

*Theorem 2:* Let  $A^*$  and  $H^*$  be optimal solutions to AP and ATSP instances, respectively, with the same cost matrix  $C$ . Assume that  $A^*$  consists of at least two cycles. Then for each  $K \in \mathcal{C}(A^*)$ , the following inequalities hold:

$$f_C(A^*) \leq f_C(A^*) + u_{A^*}^K \leq f_C(H^*).$$

*Proof:* The first inequality is obvious, since  $u_{A^*}(e) \geq 0$  for every  $e \in A^*$ . Now we show that  $f_C(A^*) + u_{A^*}^K \leq f_C(H^*)$ . Take any  $K \in \mathcal{C}(A^*)$ , and take any  $e \in K \setminus H^*$ . Let  $\mathcal{A}_-(e)$  be the set of all AP solutions without  $e$ , so  $\{A \in \mathcal{A} : e \notin A\}$ . Moreover, let  $A_-^*(e) \in \arg \min\{f_C(A) : A \in \mathcal{A}_-(e)\}$ . From Theorem 1(1) and 1(3), we have that  $u_{A^*}(e) = f_C[A_-^*(e)] - f_C(A^*)$ , and that  $f_C(A^*) + u_{A^*}(e) = f_C[A_-^*(e)] \leq f_C(H^*)$ , since  $H^* \in \mathcal{A}_-(e)$ . Hence  $f_C(A^*) + u_{A^*}^K \leq f_C(A^*) + u_{A^*}(e) = f_C[A_-^*(e)] \leq f_C(H^*)$ . ■

Based on Theorem 2, we may ask the question which subcycle in  $A^*$  should be “broken”. The most effective choice is the cycle  $K$  in which  $u_{A^*}^K$  is maximal, since it causes the largest increase in the lower bound  $f_C(A^*) + u_{A^*}^K$ . However, all tolerance values in all cycles must be computed to guarantee that we obtain an arc with maximal value of  $u_{A^*}^K$ . This is usually a time-consuming matter. Hence, it may be worthwhile to restrict the tolerance calculations to a ‘not too large’ subset of  $\mathcal{C}(A^*)$ . Let  $O \subseteq \mathcal{C}(A^*)$ . Denote and define the *bottleneck tolerance* with respect to  $O \subseteq \mathcal{C}(A^*)$  by  $bu_{A^*}(O) := \max\{u_{A^*}^K : K \in O\}$ , and the corresponding *bottleneck bound* by  $lb_{A^*}(O) = f_C(A^*) + bu_{A^*}(O)$ . The choice for the set of offenders  $O$  determines the values of the bottleneck tolerances and bounds. For example, if  $\mathcal{C}(A^*) = \{K_1, K_2\}$ ,  $u^{K_1} = 1$ , and  $u^{K_2} = 5$ , then  $bu_{A^*}(\mathcal{C}(A^*)) = bu_{A^*}(\{K_1\}) = 5$ , whereas  $bu_{A^*}(\{K_2\}) = 1$  and  $bu_{A^*}(\emptyset) = 0$ .

We consider special sets of offenders in  $\mathcal{C}(A^*)$ :

The *Entire Cycle Set (ECS)*. Define  $O_E := \mathcal{C}(A^*)$ . So the bottleneck tolerance value in the entire set of subcycles of  $A^*$  is taken. Note that  $bu_{A^*}(O_E) \geq bu_{A^*}(O)$  for every  $O \subseteq \mathcal{C}(A^*)$  and for every  $A^*$ .

The *Smallest Cycle Set (SCS)*. Define  $O_S := \{K^*\}$ , where  $K^*$  is a cycle of  $A^*$  with the smallest cardinality, i.e.,  $K^* \in \arg \min\{|K| : K \in \mathcal{C}(A^*)\}$ .

The concept of bottleneck tolerances uses the structure of an assignment solution to increase the lower bound. For instance, suppose that an assignment solution of a randomly generated instance consists of many small cycles. We may expect a high bottleneck tolerance value, since the maximum is taken from a large set of numbers. Note also that if an assignment consists of a large number of cycles, then, on average, the gap between the AP and the ATSP solution values is wide. So there is a relationship between the size of the gap and the value of the bottleneck tolerance.

The ECS choice leads to the tightest upper tolerance-based lower bounds. The SCS choice is taken into account, because it gives a good approximation for the ECS bound in a short time. We claim that, in many situations, the value of  $bu_{A^*}(O_E)$  is attained on a smallest cycle, and hence,  $bu_{A^*}(O_S)$  is a good approximation of it. The intuition behind this claim is the following. Suppose upper tolerance values are randomly dispersed over the arcs of a minimum cycle cover. The minimum tolerance value of a small cardinality cycle is then relatively large; therefore, it is likely that  $bu_{A^*}(O_E)$  is attained on a smallest cycle of  $A^*$ . Table III shows that the fraction

of subproblems in a BnB search tree for which this event occurs, is about 45%. The high percentages for symmetric instances are distorting, since initial minimum cycle covers usually consist of many cycles of cardinality 2.

TABLE III

PERCENTAGE  $bu_{A^*}(O_E)$  IN SMALLEST CYCLES AND REDUCTIONS BY ECS AND SCS LOWER BOUNDS

Instance	$bu_{A^*}(O_E)$ in smallest cycles	$r(O_E)$	$r(O_S)$
ATSP LIB	46.38%	19.97%	6.39%
Degree of symmetry 0.33	60.98%	34.62%	17.07%
Degree of symmetry 0.66	69.48%	26.66%	12.27%
Full symmetry	88.49%	21.64%	14.61%
Asymmetric random	43.09%	50.47%	43.31%
Degree of sparsity 50%	43.73%	56.50%	48.99%
Degree of sparsity 75%	44.06%	45.78%	40.45%
Degree of sparsity 90%	44.49%	49.86%	35.45%

The next natural question is: what is the difference in quality between  $lb_{A^*}(O_C)$  and  $lb_{A^*}(O_S)$ ? To measure the quality of a lower bound, we introduce the *reductions*  $r(O)$  of the gap between  $f_C(A^*)$  and  $f_C(H^*)$  achieved by the lower bound  $lb_{A^*}(O)$ . Define  $r(O) = \frac{bu_{A^*}(O)}{f_C(H^*) - f_C(A^*)} \times 100\%$ . Table III compares  $r(O_E)$  and  $r(O_S)$ . The results show that, for (quasi-)symmetric and ATSP LIB instances, the ECS choice clearly constructs better lower bounds than the SCS choice. However, the SCS choice gives a satisfactory approximation for asymmetric random and sparse instances, while it is generally much faster to compute.

In BnB settings, lower bounds can be computed at every node of the search tree. A high quality bound, such as  $lb_{A^*}(O_E)$  allows the BnB algorithm to discard a large number of nodes, but it requires long computing times at each node considered. In order to find the balance between bound quality and computing times, computational experiments are performed in Section VI.

The approaches are clarified with the following ATSP example, borrowed from [2, page 381].

Consider the ATSP instance with the following inter-city matrix.

City	1	2	3	4	5	6	7	8
1	$\infty$	2	11	10	8	7	6	5
2	6	$\infty$	1	8	8	4	6	7
3	5	12	$\infty$	11	8	12	3	11
4	11	9	10	$\infty$	1	9	8	10
5	11	11	9	4	$\infty$	2	10	9
6	12	8	5	2	11	$\infty$	11	9
7	10	11	12	10	9	12	$\infty$	3
8	10	10	10	10	6	3	1	$\infty$

The unique AP solution  $A^*$  consists of the three cycles  $K_1 = \{(1, 2), (2, 3), (3, 1)\}$ ,  $K_2 = \{(4, 5), (5, 6), (6, 4)\}$ , and  $K_3 = \{(7, 8), (8, 7)\}$ . Moreover,  $f_C(A^*) = 17$  and  $f_C(H^*) = 26$ . Figure 1 depicts  $A^*$  and the upper tolerance values of the arcs.

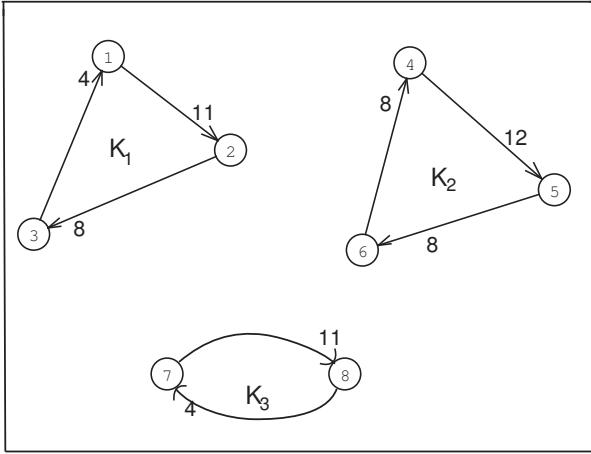


Fig. 1. Minimum cycle cover with arc upper tolerances

In this example,  $u^{K_1} = 4$ ,  $u^{K_2} = 8$ , and  $u^{K_3} = 4$ . So  $bu_{A^*}(O_E) = 8$  is attained on the arcs  $(2, 3)$  and  $(5, 6)$  in cycle  $K_2$ , and  $bu_{A^*}(O_S) = 4$  on arc  $(8, 7)$  in the smallest cycle  $K_3$ . Therefore,  $lb_{A^*}(O_S) = 21$  and  $lb_{A^*}(O_E) = 25$ . Since  $f_C(H^*) = 26$ ,  $r(O_E) = \frac{8}{26-17} \times 100\% = 88.8\%$ ,  $r(O_S) = \frac{4}{26-17} \times 100\% = 44.4\%$ . For this instance,  $lb_{A^*}(O_E) = 25$  is tighter than all other bounds discussed in [2].

Applying the concept of bottleneck tolerances not only increases lower bounds, but it also strengthens the branching. The previous section shows that it is worthwhile to branch on an arc with a small upper tolerance value. But which one should we choose? Figure 2 depicts an enumeration of feasible solutions of an AP instance in a non-decreasing order of cost values. A careful branching strategy, which branches on a smallest upper tolerance arc, obtains all AP solutions with cost smaller than  $f_C(H^*)$ . However, if an algorithm branches on a bottleneck tolerance arc, then it traverses the enumeration with larger steps, and it is likely to arrive

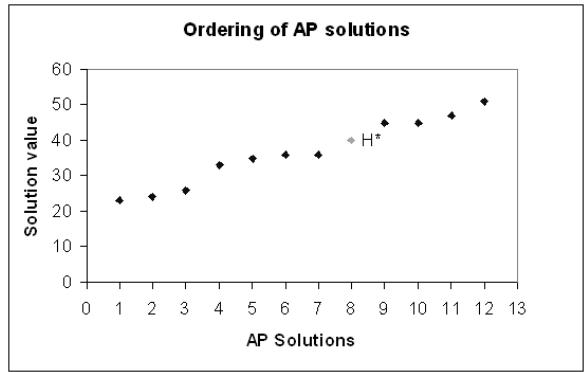


Fig. 2. Enumeration of AP solutions

at  $H^*$  in less branching steps. That is, of course, if it does not exclude survival arcs. Table III indicates that the exclusion of an ECS bottleneck arc from an asymmetric, randomly generated instance brings the solution value of the next subproblem on average about 50% closer to  $f_C(H^*)$ . We propose branching rules based on bottleneck tolerances arcs obtained by the ECS and the SCS choices. We call these the *ECS* and *SCS branching rules*, respectively.

## V. THE DESCRIPTIONS OF THE ALGORITHMS

In this section, we introduce BnB algorithms which use bottleneck tolerances to direct their branching decisions and to tighten their lower bounds. A cost-based BnB algorithm serves as a benchmark for the performance of the tolerance-based algorithms. BnB algorithms are built up from the following four basic ingredients; see for example [17].

The *branching rule* prescribes how the current problem should be partitioned into subproblems. We consider three branching rules: the “normal” branching rule, which branches on a smallest cycle from the current AP solution in a non-increasing order of arc costs, the ECS branching rule, and the SCS branching rule.

The *search strategy* prescribes which subproblems should be expanded next. The fixed search strategy of all algorithms discussed here is *depth first search (DFS)*, which solves the most recently generated subproblem first.

The *upper bounding strategy* prescribes how Hamiltonian tours should be constructed in the BnB process. All algorithms apply the patching procedure from [14] at each node of the search tree.

The *lower bounding strategy* constructs a lower bound of the solution value of each subprob-

lem. The lower bounds  $lb_{A^*}(\emptyset) = f_C(A^*)$ ,  $lb_{A^*}(O_E)$ , and  $lb_{A^*}(O_S)$  are considered.

The DFS strategy is often used, since it allows to solve large and difficult instances to optimality; see for example [3]. In case of DFS, the choice of the branching in the top node fixates the part of the search tree that the algorithm searches through for a long period of time. So the choice of a correct branching variable is vital, particularly in the top nodes of the search tree. If the search strategy is Best First Search (BFS), the choice of the branching variable is far less important.

The normal and the SCS branching rules apply the subtour elimination scheme from [4], where the branching takes place on all arcs in one cycle. The ECS branching rule takes an exceptional position, because it does not necessarily branch on arcs in a single subtour.

The BnB algorithms to be tested are listed in Table IV. All algorithms apply the solver from [13] to solve the APs and to compute the upper tolerance values, and use the reduction procedure from [3] to sparsify the matrix at the top node of the search tree.

The flowchart in Figure 3 provides a schematic view of a tolerance-based BnB algorithm. Algorithms 1, 2, 4, 5, and 7 in Table IV apply the ECS branching rule or bound. In each subproblem of these algorithms  $n$  upper tolerances are computed where  $n$  is the dimension of the current subproblem. Algorithms 3, 6, and 8 in Table IV use the SCS branching rule or bound. The upper tolerances are computed by applying Theorem 1, namely  $u_{A^*}(e) = f_C[A^*(e)] - f_C[A^*]$ . Although solving an AP from scratch takes  $O(n^3)$  time, it is well known (see e.g., [2, pages 370-371]) that for finding an optimal solution  $A^*(e)$  based on the given AP solution  $A^*$ , it is enough to perform only one labeling procedure in the Hungarian method, which can be done in  $O(n^2)$  time. Hence, the complexities are  $O(n^3)$  and  $O(|K^*|n^2)$ , respectively.

## VI. COMPUTATIONAL EXPERIMENTS WITH ATSP INSTANCES

In this section, computational experiments are conducted on the algorithms listed in Table IV. The central questions are: do tolerance-based algorithms have smaller search trees, and if this is true, are the reductions sufficient to compensate for the time invested in tolerance computations?

We have selected instances from the ATSPLIB (see [19]) that are solvable within reasonable time limits. The random instances have degree of symmetry 0, 0.33, 0.66, and 1. The sparse random instances have degree of sparsity of 50%, 75%, and 90%. The sizes of the randomly generated instances are reported in Table V.

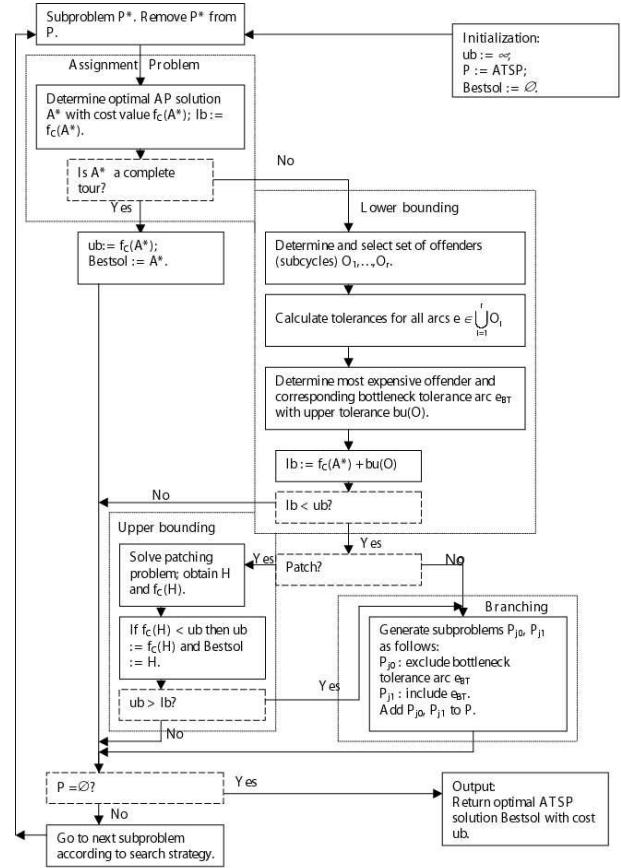


Fig. 3. Flowchart of a tolerance-based BnB algorithm

For each problem set and for all instance sizes, 10 instances have been generated. The experiments are conducted on a Pentium 4 computer with 256 MB RAM memory and 2 GHz speed. In all tables  $s(i)$  and  $t(i)$  ( $i = 0, 1, \dots, 8$ ) are the size of the search trees and the solution time, respectively.

We compare the BnB search trees with tolerance-based lower bounds, with tolerances based branching rules, and with a combination of both. Tables VI and VII show that BnB algorithms with only new lower bounds have smaller search trees than the conventional algorithm, Algorithm 0. The SCS branching rules also achieves reductions for most instances, but the ECS branching rule only reduces the search trees for random instances. The reductions of the joint use of tolerance-based lower bounds and branching rules are often larger than the reductions when they are used separately.

For many instances, it holds that  $s(1) > s(2)$ , i.e., branching on an SCS arc is more effective than branching on an ECS arc. This is counterintuitive, but the discussion in Section III may explain this phenomenon. It was observed there that small upper tolerance arcs are more likely to be in an optimal ATSP solution than large upper

TABLE IV  
VARIANTS OF BNB ALGORITHMS

Algorithm	Lower bound	Branching rule	Comments
0	$lb_{A^*}(\emptyset)$	Normal	basic algorithm
1	$lb_{A^*}(\emptyset)$	ECS	efficiency of ECS branching rule
2	$lb_{A^*}(\emptyset)$	SCS	efficiency of SCS branching rule
3	$lb_{A^*}(O_E)$	Normal	efficiency of ECS bound
4	$lb_{A^*}(O_E)$	ECS	efficiency of ECS BnB
5	$lb_{A^*}(O_E)$	SCS	not considered
6	$lb_{A^*}(O_S)$	Normal	efficiency of SCS bound
7	$lb_{A^*}(O_S)$	ECS	not considered
8	$lb_{A^*}(O_S)$	SCS	efficiency of SCS BnB

TABLE V  
SIZE  $n$  OF THE INSTANCES USED IN THE EXPERIMENTS

Tables	Sparse	Usual random	Degree of symmetry	Full symmetry
			0.33 and 0.66	
I,II	$n = 60, 70, 80$	$n = 60, 70, 80$	$n = 60, 70, 80$	$n = 60, 70$
VII		$n = 60, \dots, 200$		
III,XI	$n = 100, 200, 400$	$n = 60, \dots, 200$	$n = 60, 70, 80$	$n = 60, 70, 80$
IX		$n = 60, \dots, 1000$		
X	$n = 100, 200, 400$		$n = 60, 70, 80$	$n = 60, 70, 80$

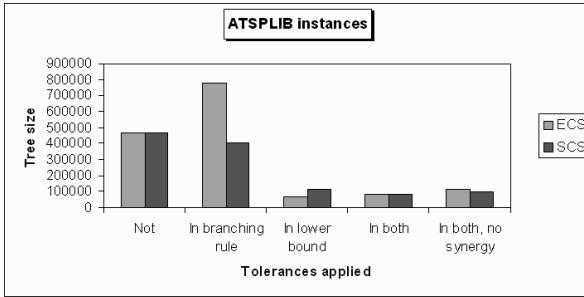


Fig. 4. Synergy effects for ATSPLIB instances

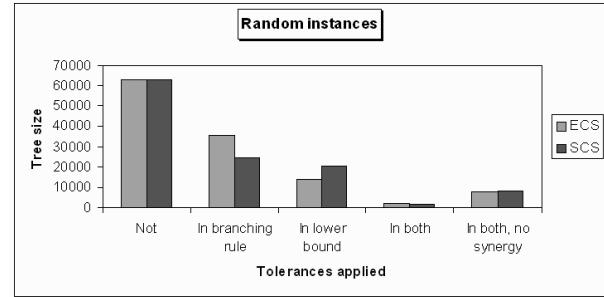


Fig. 5. Synergy effects for random instances

tolerance arcs. Since the upper tolerance value of an ECS arc is generally higher than the upper tolerance value of an SCS arc, the ECS branching rule is more likely to delete survival arcs than the SCS branching rule.

The search trees of ATSPLIB and random instances are depicted in Figure 4. The first column shows the values of  $s(0)$ , the second column the values  $s(1)$  and  $s(2)$ , the third the values  $s(3)$  and  $s(6)$ , and the fourth the values of  $s(4)$  and  $s(8)$ . If the reductions of tolerance based lower bounds and branching rules would have been independent, then the expected size of the search trees  $\bar{s}(4)$  and  $\bar{s}(8)$  are  $\bar{s}(4) = s(0) \frac{s(1)}{s(0)} \frac{s(3)}{s(0)}$ , and  $\bar{s}(8) =$

$s(0) \frac{s(2)}{s(0)} \frac{s(6)}{s(0)}$ , i.e., the reduction of the branching rule times the reduction of the lower bound. These values are represented in the column “Both without synergy” in Figures 4 and 5. However, the actual values of  $s(4)$  and  $s(8)$  are lower than those of  $\bar{s}(4)$  and  $\bar{s}(8)$ , indicating that the joint use of tolerance-based lower bounds and branching rules leads to additional search tree reductions. We call this remarkable phenomenon the *synergy effect*. Since Algorithms 4 and 8 benefit from the synergy effect, we concentrate on these algorithms in the experiments below.

The question we ask now is whether the search tree re-

TABLE VI  
SEARCH TREE SIZES FOR ATSPLIB INSTANCES

Instance	$n$	$s(0)$	Entire Cycle Set				Smallest Cycle Set			
			$s(1)$	$s(3)$	$s(4)$	$\frac{s(0)}{s(4)}$	$s(2)$	$s(6)$	$s(8)$	$\frac{s(0)}{s(8)}$
ft53	53	20111	*	7039	*		89511	17703	20545	0.98
ft70	70	25831	70047	5619	5861	4.41	22843	6717	4993	5.17
ftv33	34	7065	3867	1983	748	9.45	6007	3137	1569	4.50
ftv35	36	6945	18871	2553	3109	2.23	12047	3219	2265	3.07
ftv38	39	6195	9726	1381	1523	4.07	14663	2821	2387	2.60
ftv44	45	619	976	187	165	3.75	937	249	195	3.17
ftv47	48	29025	29121	8017	3302	8.79	48345	9703	8393	3.46
ftv55	56	92447	98433	12413	10100	9.15	114641	26483	26023	3.55
ftv64	65	43441	89752	9449	8319	5.22	162639	11007	17173	2.53
ftv70	71	253873	459532	25939	50024	5.07	136296	52289	18937	13.41

\*Memory exhausted

TABLE VII  
SEARCH TREE SIZES FOR RANDOM INSTANCES

n	$s(0)$	Entire Cycle Set				Smallest Cycle Set			
		$s(1)$	$s(3)$	$s(4)$	$\frac{s(0)}{s(4)}$	$s(2)$	$s(6)$	$s(8)$	$\frac{s(0)}{s(8)}$
60	3808	3275	978	323	11.79	1032	2832	221	17.23
70	4528	4085	1138	312	14.51	1286	1781	247	18.33
80	9014	3937	2414	217	41.53	2494	2746	256	35.21
100	9002	3645	1978	174	51.74	2188	5306	135	66.68
200	36390	20497	7114	858	42.41	17612	7612	796	45.72

ductions of Algorithms 4 and 8 are sufficient to compensate for the time invested in the tolerance calculations. Tables VIII, IX, and X show that Algorithm 8 obtains the shortest solution times for asymmetric instances, sparse instances, and instances with degree of symmetry 0.33 and 0.66, but the solution times of Algorithm 0 are slightly better for ATSPLIB and fully symmetric instances. For sparse instances, the solution times of Algorithm 8 are also shorter, but the advantage reduces as sparsity increases. Algorithm 4, which uses the ECS lower bound and branching rule, generally requires too much tolerance calculation time to be competitive, in spite of its small search trees.

We expect that the ECS branching rule is more effective than SCS if there are many optimal solutions of the AP relaxation. One optimal solution may be a better starting point for BnB than the other. If an algorithm

branches on an element with upper tolerance 0, then it jumps from one optimal assignment to another. As a consequence, it may become trapped in a solution which is a bad starting point for further branching. This is more likely to happen for the SCS branching rule, since the value of the SCS bottleneck tolerance is always at most the value of the ECS upper tolerance. So the ECS branching rule is most likely to escape from this trap; if there is an offender with strictly positive cost of removal, then it will select this one. For such cases, it pays off to select a larger set of offenders than only a single smallest cycle.

Finally, we analyze the source of the search tree reductions of Algorithms 4 and 8. A BnB algorithm first finds an optimal solution and then proves the optimality of that solution, i.e., all remaining subproblems are discarded. Table XI shows that the cost-based Algorithm

TABLE VIII  
SEARCH TREE SIZES AND SOLUTION TIMES OF ATSPLIB INSTANCES

Instance	n	s(0)	t(0)	s(4)	t(4)	s(8)	t(8)
ft53	53	20111	2.31	*	*	20545	6.34
ft70	70	25831	3.85	5861	28.30	4993	1.87
ftv33	34	7065	0.22	748	0.49	1569	0.11
ftv35	36	6945	0.22	3109	2.03	2265	0.27
ftv38	39	6195	0.22	1523	1.54	2387	0.44
ftv44	45	205	0.06	165	0.22	195	0.05
ftv47	48	29025	1.32	3302	6.15	8393	1.59
ftv55	56	92447	4.51	10100	29.56	26023	10.00
ftv64	65	43441	3.13	8319	34.73	17173	11.43
ftv70	71	253873	23.08	50024	205.11	18937	9.12

\*Memory exhausted

TABLE IX  
SEARCH TREE SIZES AND SOLUTION TIMES OF ASYMMETRIC RANDOM INSTANCES

n	s(0)	t(0)	s(4)	t(4)	s(8)	t(8)
60	3808	0.33	323	1.21	221	0.27
70	4528	0.38	312	1.76	247	0.27
80	9014	1.26	217	2.36	256	0.38
100	9002	1.92	174	2.64	135	0.22
200	36390	33.	858	73.	796	11.
300	178498	481.	936	287.	1506	66.
400	284994	1410.	742	541.	1216	120.
500	434576	3687.	1878	2684.	2253	439.
1000	922890	39516.	1421	15569.	3739	5360.

TABLE X  
SEARCH TREE SIZES AND SOLUTION TIMES OF SYMMETRIC AND SPARSE INSTANCES

Instance	s(0)	t(0)	s(4)	t(4)	s(8)	t(8)
Degree of sparsity 50%	368736	1341.	2687	1173.	1785	70.
Degree of sparsity 75%	386468	1467.	3259	1247.	2432	117.
Degree of sparsity 90%	423284	1669.	3466	1909.	2521	141.
Degree of symmetry 0.33	58878	8.08	5846	25.66	4173	2.91
Degree of symmetry 0.66	202894	32.42	65777	288.79	32914	18.57
Full symmetry	13390054	1759.	910398	3502.	11382356	3631.

TABLE XI  
PERCENTAGE OF SUBPROBLEMS SOLVED BEFORE AN OPTIMAL  
SOLUTION IS FOUND

Instance	Algorithm 0	Algorithm 4	Algorithm 8
ATSLIB	30.48%	64.48%	39.63%
Degree of symmetry 0.33	92.60%	62.59%	70.42%
Degree of symmetry 0.66	85.12%	49.14%	61.65%
Full symmetry	35.70%	31.71%	32.08%
Asymmetric random	90.02%	77.27%	75.76%
Degree of sparsity 50%	95.15%	82.49%	76.88%
Degree of sparsity 75%	96.56%	80.77%	77.65%
Degree of sparsity 90%	95.83%	72.30%	82.35%

0 spends a relatively large amount of time on finding an optimal solution compared to Algorithms 4 and 8, particularly for non-symmetric random instances. This result indicates that the improved branching of tolerance-based algorithms is the predominant source of the search tree reductions. Table XI also indicates that fast algorithms often spend the smallest fraction of time on finding an optimal solution. The value of an optimal solution is the tightest possible upper bound and can be used to fathom a large number of subproblems. We conclude that accurate branching is the key to good performance of depth first search BnB algorithms. Moreover, since tolerance-based algorithms find optimal solutions faster, the results in case of premature termination are likely to be better than for cost-based algorithms. This property may be very useful in case of solving large problems within limited times; see [20].

There seems to exist the following paradox. The use of tolerance-based *lower bounds* cause the largest search tree reductions according to Tables VI and VII, and hence, one may expect that these algorithms need less time to prove the optimality of the solution at hand. However, Table XI points out that tolerance-based BnB algorithms need relatively less time to find an optimal solution. An explanation is that the new lower bounds cut off a large number of subproblems *before* an optimal solution is found.

## VII. SUMMARY AND FUTURE RESEARCH DIRECTIONS

We presented an experimental analysis of tolerance-based BnB type algorithms for the ATSP, and compared it with cost-based BnB algorithms. Tolerance-based algorithms reduce the search tree sizes substantially. For random instances, including both instances with degree of symmetry 0.33 and 0.66, and sparse instances, the computation times are substantially better.

The better performance of tolerance-based BnB algorithms is mainly caused by improved *branching*: a better choice of entries to be included and excluded. Upper tolerances provide better predictions of which arcs in a relaxation solution should be preserved, the *survival set*, and which arcs should be deleted, the *extinction set*.

We apply the concept of *offenders*: sources of infeasibility which must be removed from a relaxation solution in order to obtain a feasible solution for the original hard problem. The minimal cost of removing such an offender can be determined using tolerance values. This idea is used to construct new lower bounds, but it also enables the BnB algorithm to make branching steps with large increases in solution value without “jumping” across an optimal ATSP solution. The largest increase in lower bound is obtained if we consider all offenders, the *Entire Cycle Set*, which has of course the drawback of very long tolerance calculation times. It is shown that a good approximation is the *Smallest Cycle Set*, which only uses a smallest cycle in the set of offenders, so that only a few tolerances need to be calculated. Branching on the smallest cycle is a good choice, not only because a small number of new subproblems is generated, but also because it is very likely that a large branching step towards an optimal ATSP solution is made.

Tolerance-based BnB algorithms have one major drawback: they have to compute tolerances at every node of the search tree. In spite of this drawback, it turns out that BnB algorithms with the Smallest Cycle Set bound and branching rule often require shorter solution times than cost-based BnB algorithms.

The idea of branching on tolerances can be seen as similar to the idea of *strong branching* in Integer Programming; see [1]. Strong branching first explores the additional cost of setting a fractional variable at an integer value, and then decides on which variable to branch. Similar to tolerance-based branching, it first computes the additional cost of removing infeasibilities, the fractional value of a variable, before it takes the branching step. In [1], it is found that strong branching should be done only at specific nodes of the search tree. Similar strategies can be developed for tolerance-based algorithms.

An interesting direction of research is to develop book-keeping techniques that accelerate tolerances computations, and lead to solution time reductions for ATSP instances. Another direction of research is to construct tolerance-based algorithms for other COPs. Preliminary experiments with these algorithms are very promising as well.

## REFERENCES

- [1] T. Achterberg, T. Koch, and A. Martin, "Branching Rules Revisited," *Operations Research Letters*, vol. 33, no. 1, pp. 42–54, 2004.
- [2] E. Balas and P. Toth, "Branch and Bound Methods," in *The Traveling Salesman Problem*, E. Lawler, J. Lenstra, A. Rinnooy Kan, and D. Shmoys, Eds. John Wiley & Sons, Chichester, 1985, ch. 10, pp. 361–401.
- [3] G. Carpaneto, M. Dell'Amico, and P. Toth, "Exact Solution of Large-scale Asymmetric Traveling Salesman Problems," *ACM Transactions on Mathematical Software*, vol. 21, no. 4, pp. 394–409, 1995.
- [4] G. Carpaneto and P. Toth, "Some New Branching and Bounding Criteria for the Asymmetric Traveling Salesman Problem," *Management Science*, vol. 21, pp. 736–743, 1980.
- [5] M. Fischetti, A. Lodi, and P. Toth, "Exact Methods for the Asymmetric Traveling Salesman Problem," in *The Traveling Salesman Problem and its Variations*, G. Gutin and A. Punnen, Eds. Kluwer, Dordrecht, 2002, ch. 9, pp. 169–194.
- [6] G. Gessner, N. Malhotra, W. Kamakura, and M. Zmijevski, "Estimating Models with Binary Dependent Variables: Some Theoretical and Empirical Observations," *Journal of Business Research*, vol. 16, no. 1, pp. 49–65, 1988.
- [7] J. Hair, R. Anderson, R. Tatham, and W. Black, *Multivariate Data Analysis*, 5th ed. Prentice-Hall, 1998.
- [8] K. Helsgaun, "An Effective Implementation of the Lin-Kernighan Traveling Salesman Heuristic," *European Journal of Operational Research*, vol. 12, pp. 106–130, 2000.
- [9] D. Hosmer and S. Lemeshow, *Applied Logistic Regression*, 1st ed. John Wiley & Sons, 1989.
- [10] L. Hubert and P. Arabie, "Comparing Partitions," *Journal of Classification*, vol. 2, pp. 193–218, 1985.
- [11] D. Johnson, G. Gutin, L. McGeoch, A. Yeo, W. Zhang, and A. Zverovich, "Experimental Analysis of Heuristics for the ATSP," in *The Traveling Salesman Problem and its Variations*, G. Gutin and A. Punnen, Eds. Kluwer, Dordrecht, 2002, ch. 10, pp. 445–489.
- [12] D. Johnson and L. McGeoch, "Experimental Analysis of Heuristics for the STSP," in *The Traveling Salesman Problem and its Variations*, G. Gutin and A. Punnen, Eds. Kluwer, Dordrecht, 2002, ch. 9, pp. 369–444.
- [13] R. Jonker and A. Volgenant, "Improving the Hungarian Assignment Algorithm," *Operations Research Letters*, vol. 5, pp. 171–175, 1986.
- [14] R. Karp and J. Steele, "Probabilistic Analysis of Heuristics," in *The Traveling Salesman Problem*. Wiley, New York, 1990, ch. 5, pp. 181–205.
- [15] M. Libura, "Sensitivity Analysis for Minimum Hamiltonian Path and Traveling Salesman Problems," *Discrete Applied Mathematics*, vol. 30, pp. 197–211, 1991.
- [16] M. Libura, E. Van der Poort, G. Sierksma, and J. Van der Veen, "Stability Aspects of the Traveling Salesman Problem Based on  $k$ -best Solutions," *Discrete Applied Mathematics*, vol. 87, pp. 159–185, 1998.
- [17] D. Miller and J. Pekny, "Exact Solution of Large Asymmetric Traveling Salesman Problems," *Science*, vol. 251, pp. 754–761, 1991.
- [18] D. Naddef, "Polyhedral Theory and Branch-and-Cut Algorithms for the Symmetric TSP," in *The Traveling Salesman Problem and its Variations*, G. Gutin and A. Punnen, Eds. Kluwer, Dordrecht, 2002, ch. 2, pp. 29–116.
- [19] G. Reinelt, "TSPLIB - a Traveling Salesman Problem Library," *ORSA Journal on Computing*, vol. 3, pp. 376–384, 1991.
- [20] W. Zhang, "Truncated Branch-and-Bound: A Case Study on the Asymmetric TSP," in *AAAI-93 Spring Symposium on AI and NP-Hard Problems*. Stanford, 1993, pp. 160–166.

# Decision support system for Attention Deficit Hyperactivity Disorder diagnostics

Iryna Yevseyeva<sup>\*</sup>, Kaisa Miettinen<sup>†</sup> and Pekka Räsänen<sup>‡</sup>

<sup>\*</sup>University of Jyväskylä/Dept. of Mathematical Information Technology  
P. O. Box 35 (Agora) FI-40014 University of Jyväskylä, Finland

Email: [iyevsev@cc.jyu.fi](mailto:iyevsev@cc.jyu.fi)

<sup>†</sup> Helsinki School of Economics  
P.O. Box 1210 FI-00101 Helsinki, Finland  
Email: [kaisa.miettinen@hkkk.fi](mailto:kaisa.miettinen@hkkk.fi)

<sup>‡</sup> Niilo Mäki Institute, University of Jyväskylä  
P. O. Box 35 FI-40014 Jyväskylä, Finland  
Email: [pekka.rasanen@nmi.jyu.fi](mailto:pekka.rasanen@nmi.jyu.fi)

**Abstract**—In this paper, we propose a Dichotomic Classification method for ordinal classification of a set of objects estimated by verbal values on criteria. The method is implemented in the framework of decision support system and applied for the diagnostics of Attention Deficit Hyperactivity Disorder. The proposed method is based on the principles of verbal decision analysis. To perform classification, however, it adopts the bisection approach. We illustrate the method with a simple example.

**Keywords**— Multiple criteria analysis, Decision support systems, Verbal Decision Analysis, Dichotomic Classification, Attention Deficit Hyperactivity Disorder.

## I. INTRODUCTION

In this work, we propose a Dichotomic Classification method implemented in a Decision Support System (DSS) and applied for diagnostics of Attention Deficit Hyperactivity Disorder (ADHD).

In neuropsychology there are a number of disorders that are difficult to identify and for which diagnostic criteria are not completely defined. ADHD is one of the most common and the most extensively studied child psychiatric syndromes [16]. This disorder is characterized by symptoms of inattention, hyperactivity and impulsivity [1]. Major research of ADHD is based on the studies of the children of the school-age between 6 and 12 years. Even though some symptoms may appear at younger age it is difficult to estimate whether a child's behavior differs from the normal one. At the school age the child needs to exercise concentration and vigilance; he or she should also demonstrate the abilities of motor control and working memory

[4]. Although most of the research is done on young ADHD children, several investigations on adults indicate [14] developmental problems during the lifespan of persons affected by ADHD. Thus, ADHD can cause various negative consequences including poor achievement at school, more visits to the emergency rooms and more automobile accidents [14].

Currently, diagnostics of ADHD is performed with the standard questionnaire Structured Interview for Diagnostic Assessment of Children (SIDAC) for Diagnostic and Statistical Manual for Mental Disorders (DSM). However, this standard is constantly modified and has already undergone several revisions (DSM-I – DSM-IV) [1], [3]. Thus, the ADHD diagnostics is not yet even properly defined and cannot be objectively measured [2].

Another feature of the ADHD diagnostics is the descriptive character of the symptoms that allow estimating the behavior of a person (in this work, we assume the person under study to be a child). The usual way of diagnostics is based on the evaluation of the child's behavior by parents and teachers as well as on the child's own reports. These data should be enough for the clinician to be able to detect the existence of one of ADHD subtypes or the absence of ADHD.

The diagnostic problems, including ADHD detection, are typical classification tasks where it is necessary to assign the set of objects estimated on a set of criteria into predefined set of classes. When diagnosing ADHD, the clinician should evaluate the children according to the set of common ways of behavior or symptoms typically associated with ADHD sufferers. They should

then either be assigned to the set of ADHD subtypes or the absence of ADHD should be concluded.

One possible way of solving classification problems is the multiple criteria analysis [5], according to which each object to be classified contains a value for each criterion. Thus, each criterion has scale of ordered values, while the criteria itself can be ordered or unordered.

We may face two types of classification problems: ordinal where the classes are ordered, and nominal where there is no possibility to establish the order between classes. In this work, we assume ordered classes. In such a case, the classes can be defined with boundary objects. A boundary object is an object that lies on the border between two classes, and changing values on at least one criterion will move this object to a neighboring class. By dominance relation between any object and lower and upper boundaries of the class we may get to know whether the object belongs to the class or not. The object should be assigned to the class if it dominates any lower boundary of the class and is dominated by any upper boundary of the same class.

In the ordinal classification problem the number of classes is predefined, but the boundary objects are usually unknown and should be defined in the framework of preference elicitation procedure.

In the multiple criteria analysis the person that is responsible for the obtained decision is called decision maker (DM). In the ADHD diagnostics the DM is a clinician.

The above mentioned features of the ADHD diagnostic problems, among them poorly or subjectively measured nature of symptoms as well as the descriptive or verbal manner of criteria estimations, narrow down the set of possible methods that can be applied to resolve such classification tasks.

In the framework of Verbal Decision Analysis (VDA) several methods that take into account described features have been developed by Larichev and his colleagues [6-13]. Two methods have been developed recently for the ordinal classification: ORdinal CLASSification (ORCLASS) [8], [10] and Subset Alternative Classification (SAC) [6], [11], [12]. The methods aim at reducing the number of objects classified by the DM directly when compared to the complete set of given objects. The main feature of the VDA methods is the ability to operate with

qualitative information without directly transforming it into a quantitative form. Usually, multicriteria methods are not adapted to work with verbal information in their pure form: instead, the DM is asked to express his or her preferences in a numerical analogue. However, in the cases such as ADHD diagnostics, it is difficult to give precise numerical estimations when describing behavior.

While developing the VDA methods, researchers were exploring ways to avoid transformation, by the DM, of verbal criteria values into numerical analogues. They used the fact that values on the scales of criteria are ordered according to their importance to the DM in such a way that the maximal rank has the most desirable value. Thus, to each value on the scale of a criterion a rank corresponds. Operating with ranks allow saving transparency of verbal values for the DM.

Classification with VDA is interactive. The preference elicitation procedure is built in the methods and is realized through a dialog with the DM: at each iteration the DM assigns an object selected by the VDA method to some class. Based on the answer of the DM some objects are automatically classified. The procedure is repeated until each object is assigned. The consistency of information received from the DM is checked with the information obtained previously. This is necessary due to the human nature: people may make errors and may be inconsistent with their preferences. In case of contradiction, it is proposed for the DM to rethink the answers. At the end of classification the DM can get explanation about the appearance of each object in the class.

We developed the new Dichotomic Classification (DC) method according to the principles of VDA, but with improved effectiveness when compared to the previously created methods.

The rest of the paper is organized in the following way. Section 2 presents the DC method and Section 3 illustrates the method with a simple example. In Section 4, we describe the problem area of the ADHD diagnostics where DC has been applied. In Section 5, the DSS, in the framework of which the DC and SAC methods have been tested is presented. In the same section we consider the multiple criteria analysis model of ADHD. Finally, we conclude the presented here study.

## II. DICHOTOMIC CLASSIFICATION METHOD

The proposed Dichotomic Classification method realizes classification of a set of given objects (that may have verbal values on the criteria scales) into ordered classes. We consider two cases: a case where the set of criteria is ordered according to the importance for the DM and a case where there is no order available.

The proposed method is based on the principles of VDA [10] and the bisection method [17] or modification of the more popular dichotomous search [15]. The method improves the efficiency of the classification procedure when compared to the other VDA methods where the object that the DM is supposed to classify is found after extensive calculations. This task become computationally complex and time-consuming when problem gets bigger. In other words, VDA methods try to identify the most informative object to be shown to the DM, but if this task is too complex, the classification suffers. For this reason, we here suggest procedure with simple rule for the object selection to be classified by the DM at each iteration of the preference elicitation procedure.

The initial information for the DC method is the same as for SAC: the number of classes and their order, the set of criteria with the scale of possible values for each of them and the set of given objects must be available.

In this paper, we consider the set of given objects to be equal to the set of objects obtained as the Cartesian product of scales of criteria as in ORCLASS. However, the method also works with the set of objects given by the DM.

The initial data are specified: the set of criteria  $G = \{g_1, g_2, \dots, g_n\}$  and the set of classes  $L = \{l_1, l_2, \dots, l_s\}$ . Furthermore, the set of objects is defined as the Cartesian product of all scales of criteria  $A = S(g_1) \times S(g_2) \times \dots \times S(g_n)$ . For each criterion  $g_j$  from the set  $G$  the scale of values  $S(g_j) = \{g_{j1}, g_{j2}, \dots, g_{jt}\}$  with the following ranks for them  $r_j = \{1, 2, \dots, t_j\}$  (where  $t_j$  is the number of possible values on the scale of criterion  $g_j$ ) are defined. The values on the criteria scales are ordered as well as classes. The most desirable criterion value and the most desirable class have the biggest ranks. We consider two cases: when criteria can be ordered

according to the importance for the DM (see relation (4)) and when the information about the order between criteria is absent. We are solving a maximization problem, where higher values are preferred.

At the beginning, for each object  $a_i$  from the set  $A$  the set of possible classes is the set of all classes and  $L_i = L$ . However, with the classification (direct or indirect) of some objects the number of possible classes is reduced (see *Step 6* of the DC algorithm below) until for each object it becomes equal to one  $|L_i| = 1$ . When this is the case for all objects, the classification is finished.

In ORCLASS [8], [10] and SAC [6], [11], [12], the object, which is proposed to the DM for the classification, is selected based on its informativeness. This property of object shows how many objects can be classified indirectly (see *Step 6* of the DC algorithm below) after the classification of this object. The informativeness can be calculated according to the relation of preference between objects and with regards to the relation of order between classes [10]. Thus, at each iteration the object with the maximal informativeness is selected and proposed to the DM. The recalculation of the informativeness for each object is done at each iteration. This task becomes computationally complex and time-consuming for large object sets. We propose to select the object to be shown to the DM for the classification according to the bisection of the set of given objects at each iteration. In bisection the set of objects is simply divided into two sets and the DM is asked to classify the object in the middle. At the next iterations the middle objects are searched in both sides of the divided searching space.

In the same way as in the other VDA methods, the qualitative values on the criteria, are compared not directly, but with ranks of objects on the scales of criteria (or ranks of objects in the case of ordered criteria). For comparison of verbal values we should assign the ranks to the values on the scales of criteria according to their order. Rank operations are valid only with the condition that difference between two neighboring values on the scale of criteria is the same for all pairs of neighboring values.

The DC method utilizes two basic relations of the ORCLASS and SAC methods:

1. The dominance (or outranking) relation between two objects, according to which the object  $a_i$  is preferred to the object  $a_f$  if it is preferred on at least one criterion  $j$  so that  $g_j(a_i)Pg_j(a_f)$  and indifferent on the rest of criteria  $g_z(a_i)Ig_z(a_f)$ ,  $z = 1, \dots, n; j \neq z$ . But here instead of real values of objects on the scales of criteria their ranks are compared. For instance,  $r_j(a_i)$  and  $r_j(a_f)$  are ranks of objects  $a_i$  and  $a_f$  on the scale of criterion  $g_j$ . Consequently, for dominance of the object  $a_i$  when compared to the object  $a_f$  the following relations are estimated  $r_j(a_i)Pr_j(a_f)$  and  $r_j(a_i)Ir_j(a_f)$ . Then the dominance relation between the two objects is defined as follows:

$$a_iSa_f \text{ (} a_i \text{ dominates } a_f \text{),} \quad (1)$$

$$\text{if } r_j(a_i) \geq r_j(a_f), \\ j=1, \dots, n; i, f=1, \dots, m; i \neq f$$

and  $r_j(a_i) > r_j(a_f)$  on at least one  $j$ .

However, this relation is not always enough when considering all the situations that may appear in the decision aiding process. That is why the next relation is also used.

2. The ordering relation between objects from different classes. For instance, objects from the second class may be preferred to objects from the first one and so on. If the object  $a_i$  belongs to the class  $l_p$  and the object  $a_f$  to the class  $l_k$ , where  $k$  and  $p$  are class ranks, and we know that the object  $a_i$  dominates the object  $a_f$ , then the class  $l_p$  should have a bigger rank when compared to the class  $l_k$ :  $p > k$  (as we are considering a maximization problem, and the bigger rank of the class is preferred to the smaller one). The following binary relation on the set of objects  $A$  and the set of classes  $L$  is established:

$$a_iOa_f \text{ (} a_i \text{ belongs to the class} \\ (2)$$

with bigger rank than the rank of the class, to which  $a_f$  belongs),

$$\text{if } a_iSa_f \text{ and } a_i \in l_p,$$

$$\text{then } a_f \in l_k, \text{ where } p > k.$$

For the non-contradictory classification the following relation should be fulfilled:

$$\text{if } a_iSa_f \text{ is true, then } a_fOa_i \text{ is not true.} \\ (3)$$

This means that objects from the class with a smaller rank cannot dominate the objects from the class with a bigger rank.

In this work we have also been considering the case where it is possible to order criteria according to their importance to the DM. Here an additional condition of preference between criteria is used (if such information is available):

$$g_jPg_z \text{ (criterion } g_j \text{ is more} \\ (4) \\ \text{important than } g_z \text{ for the DM),}$$

$$\text{if } j > z \text{ and } j, z=1, \dots, n; j \neq z.$$

Let us see how these relations are utilized in the DC method.

The algorithm of the DC method consists of the following steps:

*Step 1* At the first stage the data are initialized: the “best”  $a_{best}$  (the object that has the most desirable value on each criterion) and the “worst”  $a_{worst}$  (the object that has the least desirable value on each criterion) objects from the set of objects  $A$  (defined as the Cartesian product of criteria scales) are classified into the classes with the highest and the lowest ranks, respectively. At this stage, for each object  $a_i$  from the set  $A$  the set of possible classes  $L_i = \{l_1, l_2, \dots, l_s\}$  is assigned. Initially this set contains all classes  $L_i^{old} = L_i = L$ , where  $L_i^{old}$  is the set of possible classes at the previous iteration.

*Step 2* If all objects has been classified, the procedure stops. Due to this, each object  $a_i \in A_r$  is classified exactly to one class  $|L_i|=1$ . This condition is checked at the beginning of each iteration: if the size of the not yet classified objects set is bigger than two then classification continues followed by *Step 3*, else if the size of the not yet classified objects set is less than two the DM is asked to classify the rest of objects one by one at *Step 4*.

*Step 3* The “middle” object  $a_{middle}$  is calculated as an object that has middle values on criteria with regards to the ranks of the values of the “best”  $a_{best}$  and the “worst”  $a_{worst}$  objects according to the searching set.

$$r_j(a_{middle}) = \frac{r_j(a_{best}) + r_j(a_{worst})}{2},$$

where  $j = 1, \dots, n$ .

At the first iteration this set contains all the objects between the best and the worst ones. However, on the second iteration there will be two searching sets: the first set between the best and middle objects and the second set between the middle and worst objects. Thus, the number of searching sets and, thus, the number of middle objects grows at each iteration. Then the middle objects from the set are proposed to the DM one by one.

*Step 4* The DM is asked to classify the middle object. Let us assume, for instance, that the DM assigns the object  $a_{middle}$  to the class  $l_k$ :  $a_{middle} \in l_k$ .

*Step 5* At this point, the absence of inconsistencies between the current and previously made classifications is verified. The contradictions are checked with the following expressions:

$$\text{if } a_{middle} \in l_k \text{ and } a_{middle} Sa_i, \quad (5)$$

then  $a_i \notin l_p$ , where  $p > k$ ,

$$\text{if } a_{middle} \in l_k \text{ and } a_i Sa_{middle},$$

then  $a_i \notin l_p$ , where  $p < k$ .

Due to these conditions the object cannot belong to the more desirable class if it is dominated by the object that belongs to the less desirable class.

*Step 6* With regards to the classification made by the DM at *Step 3*, the set of possible classes is recalculated for each not yet classified object according to the following conditions:

$$\text{if } a_{middle} \in l_k \text{ and } a_{middle} Sa_i, \quad (6)$$

$$\text{then } L_i = L_i^{old} \bigcap \{l_1, l_2, \dots, l_{k-1}\},$$

$$\text{if } a_{middle} \in l_k \text{ and } a_i Sa_{middle},$$

$$\text{then } L_i = L_i^{old} \bigcap \{l_k, l_{k+1}, \dots, l_s\}.$$

In such a way the maximal possible class for any object  $a_i$  that is dominated by the most recently classified object  $a_{middle} \in l_k$  will be changed to the class  $l_{k-1}$ , and for any object  $a_i$  that dominate the most recently classified object  $a_{middle} \in l_k$  the minimal possible class will be changed to the class  $l_k$ . At this stage some of the objects may happened to have only one possible class  $|L_i|=1$ , which means that these objects are classified indirectly. Go to *Step 2*.

In the case of ordered criteria the relation of the dominance between objects is substituted with the preference relation that excludes situations where the objects can be of equal importance (indifferent to each other) for the DM or they may be incomparable. Then it is possible to order all the objects before the classification starts and each object will have its rank, for instance,  $r^i$  would be the rank of objects  $a_i$ . The procedure of classification is simplified and *Steps 3, 5 and 6* are modified in the following way.

At *Step 3* the middle object is searched with regards to the ranks of the searching set boundaries:

$$r^{middle} = \frac{r^{best} + r^{worst}}{2}.$$

This value is rounded if necessary (when the obtained rank is not an integer) to the smaller integer.

At *Step 5* the existence of inconsistencies is checked with regards to the conditions:

$$\text{if } a_{\text{middle}} \in l_k \text{ and } r^{\text{middle}} > r^i, \quad (7)$$

then  $a_i \notin l_p$ , where  $p > k$ ,

$$\text{if } a_{\text{middle}} \in l_k \text{ and } r^i > r^{\text{middle}},$$

then  $a_i \notin l_p$ , where  $p < k$ .

The recalculation of the set of classes for each object at *Step 6* is done in the following way:

$$\text{if } a_{\text{middle}} \in l_k \text{ and } r^{\text{middle}} > r^i, \quad (8)$$

$$\text{then } L_i = L_i^{\text{old}} \cap \{l_1, l_2, \dots, l_{k-1}\},$$

$$\text{if } a_{\text{middle}} \in l_k \text{ and } r^i > r^{\text{middle}},$$

$$\text{then } L_i = L_i^{\text{old}} \cap \{l_k, l_{k+1}, \dots, l_s\}.$$

Here  $r^i$ ,  $r^{\text{middle}}$  are the ranks of objects  $a_i$ ,  $a_{\text{middle}}$ , respectively.

The idea of finding the middle object has been used in CYCLE [13], which is another classification method; however, the rule for the object selection at each iteration of the preference elicitation procedure is different.

Let us now illustrate the DC method with a simple example.

### III. ILLUSTRATIVE EXAMPLE OF DICHOTOMIC CLASSIFICATION METHOD PERFORMANCE

In order to demonstrate the DC method we consider, as an example, the problem with two criteria  $G = \{g_1, g_2\}$  with equal importance for the DM, three values on each criterion  $g_j = \{1, 2, 3\}$ ,  $j = 1, 2$  and two classes  $L = \{l_1, l_2\}$  that are ordered in increasing order.

As we are solving a maximization problem, the best criteria values and the best class for the DM have maximal ranks. The size of the Cartesian product of the scales of criteria is equal to  $|A| = 3^2 = 9$ . The set of objects contains the following objects  $A = \{(1,1); (1,2); (1,3); (2,1); (2,2); (2,3); (3,1); (3,2); (3,3)\}$ . Here, the criterion values in the object are numbers for simplicity, but they could be verbal as well because, actually, we are operating with ranks. Thus, it is necessary to classify the nine objects into two classes. The DC method performs the following operations.

*Step 1* For each object assign the set of possible classes  $L_i^{\text{old}} = L_i = \{l_1, l_2\}$ . Allocate the object with all the best values i.e.  $a_9 = (3,3)$  into the best class  $a_9 \in l_2$  and object with all the worst values i.e.  $a_1 = (1,1)$  into the worst class  $a_1 \in l_1$ .

#### Iteration 1

*Step 2* At this point it is checked whether there are not yet classified objects for which  $|L_i| \neq 1$ . There are seven not yet classified objects  $\{a_2, \dots, a_8\}$ .

*Step 3* The set of not yet classified objects is bigger than two, thus, we search the middle object with regards to the best  $a_1$  and worst  $a_9$  objects in the following way. We obtain the rank of the middle object on each criterion by averaging the ranks of the best and worst objects.

$$r_1(a_{\text{middle}}) = \frac{r_1(a_1) + r_1(a_9)}{2} = \frac{1+3}{2} = 2;$$

$$r_2(a_{\text{middle}}) = \frac{r_2(a_1) + r_2(a_9)}{2} = \frac{1+3}{2} = 2.$$

Thus, we should take as the middle object the object with the rank two at the first criterion scale and with the rank two at the second criterion scale. We obtain the first middle object  $a_5 = (2,2)$ .

*Step 4* The middle object is proposed to the DM for classification. Let us assume that the DM assigns the object  $a_5 = (2,2)$  to the class  $l_2$ .

*Step 5* Now we should check the selection of the DM for inconsistencies with the previous classifications. At the initial step two objects were

classified  $a_1$  and  $a_9$ . According to the condition (5) the object  $a_5$  should dominate the object  $a_1$  and should be dominated by the object  $a_9$ . Both of these conditions are satisfied because the ranks of the object  $a_1$  on both criteria values are less than the ones of the object  $a_5$  and the ranks of the object  $a_9$  on both criteria values are bigger than the ones of the object  $a_5$ :  $r_j(a_1) < r_j(a_5)$  and  $r_j(a_9) > r_j(a_5)$ ,  $j = 1, 2$ .

*Step 6* At this step the set of possible classes for the not yet classified objects is recalculated according to the classification of the object at the previous step. With regards to the relation (6) we check whether there are objects that dominate or are dominated by the object classified by the DM at the *Step 4*. We find that the objects  $a_2 = (1,2)$  and  $a_4 = (2,1)$  are dominated by the object  $a_5 = (2,2)$  because  $r_j(a_5) \geq r_j(a_i)$ ,  $j = 1, 2$ ;  $i = 2, 4$ ; as well as the objects  $a_6 = (2,3)$  and  $a_8 = (3,2)$  dominate the object  $a_5 = (2,2)$  because  $r_j(a_5) \leq r_j(a_i)$ ,  $j = 1, 2$ ;  $i = 6, 8$ . According to the relation (6) the set of possible classes for the objects  $a_2, a_4, a_6, a_8$  should be recalculated. For instance, we have  $a_5 Sa_2$ , and, thus,  $L_2 = L_2^{old} \bigcap \{l_{2-1}\} = \{l_1, l_2\} \bigcap \{l_1\} = l_1$ .

In such a way these objects are classified indirectly:  $a_2 \in l_1$ ,  $a_4 \in l_1$ ,  $a_6 \in l_2$ ,  $a_8 \in l_2$ . After this step the process is repeated.

#### *Iteration 2*

*Step 2* We should check whether there are not yet classified objects. At this moment the set of not yet classified objects is  $\{a_3, a_7\}$ .

*Step 3* In the general case, if the set of not yet classified objects is bigger than two we again continue the search of the middle object on each of the two searching sets: the first set is defined with regards to the best  $a_1$  and the middle  $a_5$  objects obtained at the previous iteration, and the second set between the middle  $a_5$  and the worst  $a_9$  objects in the same way as in the previous iteration. However, in our situation we have only two objects to be classified; so, we select the first of them: the object  $a_3 = (1,3)$ .

*Step 4* The selected object is proposed to the DM for classification. Let us assume that the DM assigns object  $a_3 = (1,3)$  to the class  $l_2$ :  $a_3 \in l_2$ .

*Step 5* The contradictions with the previous classifications are checked at this step. Accordingly, we should estimate the dominance relation between the objects  $a_5$  and  $a_3$ . With regards to the condition (5) there are no inconsistencies between the classification of these two objects, because they belong to the same class.

*Step 6* Unfortunately, classification of the object  $a_3$  does not say anything about the classification of the last not yet classified object  $a_8 = (3,1)$  because the relation of the dominance does not work for these two objects. Consequently, the size of the possible classes for this object cannot be reduced.

Now we should go to the *Iteration 3*, where at *Step 4* the DM will classify the object  $a_8$ , for instance, into the class  $l_1$ :  $a_8 \in l_1$ ; and the classification is finished, because there are no more not yet classified objects.

In such a way the set of nine objects has been classified, with three iterations (three questions to the DM), into two classes. The boundary objects between the two classes have been defined:  $a_3 = (1,3)$ ,  $a_5 = (2,2)$ , and  $a_8 = (3,1)$ . They allow the unambiguous classification of any object from the set  $A$ .

For instance, for someone interested in knowing to which class the object  $a_4 = (2,1)$  belongs, the comparison with the object  $a_8$  shows it belonging to the class  $l_1$ .

For the case where criteria are ordered according to the importance for the DM, for instance, in the increasing order  $G = \{g_1, g_2\}$ , the classification procedure is simplified. The objects can be ordered from the very beginning of the classification procedure and there is always only one boundary object between classes (because the situations of incomparability and indifference between objects are excluded).

Thus, for our illustrative example the objects in the set  $A = \{a_1, \dots, a_9\}$  are ordered from the very beginning in increasing order and the set of corresponding ranks is available  $R = \{r^1, \dots, r^9\} = \{1, \dots, 9\}$ .

The middle object is selected with regards to the ranks of searching set boundaries. In our illustrative example at *Step 3* of *Iteration 1* the rank of first middle object is obtained as follows:

$$r^{middle} = \frac{r^1 + r^9}{2} = \frac{1+9}{2} = 5.$$

Thus, as a middle object we should take the object with the rank five. Then the first middle object would be the object  $a_5 = (2,2)$ .

*Step 5* and *Step 6* are modified in such a way that the relation of the dominance between objects is substituted with the relation of preference between the object ranks.

In our example at *Step 5* of *Iteration 1* inconsistencies are checked according to the condition (7). Then at *Step 6* of *Iteration 1* the recalculation of the set of possible classes for each object is done with regards to the relation (8). According to this condition the objects  $a_2, a_4, a_6, a_7, a_8$  are classified indirectly:  $a_2 \in l_1, a_4 \in l_1, a_6 \in l_2, a_7 \in l_2, a_8 \in l_2$ . After this step the object  $a_3$  is the only not yet classified object and it should be proposed to the DM for classification.

In the following section we consider the problem area of our application: ADHD diagnostics.

#### IV. ADHD DIAGNOSTICS PROCEDURE

Here we present a brief overview of ADHD.

The structured interview assumes that all participating persons answer the diagnostic questions about the behavior of the child. Usually, the following individuals are involved in the questioning procedure: the parents (usually, the mother), the teacher, and the child itself; the last modifications to DSM-IV include the clinician's opinion about the behavior of the child when the diagnosis is made. There are different opinions about the importance of the answers of different persons. Some researchers point out that the teacher's opinion is the most important, the others do not place similar importance on this aspect [2]. We assume that the opinions of the mother and the teacher are the most important; however, the child self-reporting is also taken into account, and we include the clinician on the stage of decision making. In this we follow the current practice in the ADHD diagnostics.

The SIDAC for DSM has undergone several changes in the course of time. The last version of SIDAC for DSM-IV diagnoses for children ages 6-12 can be found in [1], [3]. The structured interview for ADHD from SIDAC for DSM-IV [1] is presented in the Appendix. The ADHD consists of three groups of symptoms: inattention, impulsivity and hyperactivity. The decision about the presence of one of the three following subtypes of ADHD: 1) Attention - deficit / Hyperactivity Disorder with Predominantly Inattentive Type (ADHD with PIT), 2) Attention - deficit / Hyperactivity Disorder with Predominantly Hyperactive - Impulsive Type (ADHD with PHIT), 3) Attention - deficit / Hyperactivity Disorder: Combined Type (ADHD: CT) or the absence of any of these disorders (no ADHD) is made based on the reports of the three persons involved in the decision making. The whole questionnaire contains eighteen questions. The first nine questions allow making conclusions about the presence or absence of the inattention deficit. The second part of interview consists of other nine questions that show whether there is the hyperactivity / impulsivity disorder. The conclusion about ADHD with PIT is made when the child has shown six of the inattention symptoms and less than six of the hyperactivity / impulsivity symptoms. The ADHD with PHIT is detected when the child's behavior accords with six symptoms from the hyperactivity / impulsivity group and with less than six of the inattention group. For the ADHD: CT, six symptoms from the first and six ones from the second group should be satisfied. Otherwise the conclusion about the absence of the ADHD is made. These three rules are used in order to imitate the behavior of the clinician when making a decision about the presence of the disorder discussed here.

Next we have a look at the DSS realization of the proposed DC method and results of its application to ADHD diagnostics.

#### V. ADHD DIAGNOSTICS WITH DECISION SUPPORT SYSTEM FOR CLASSIFICATION

##### A. Decision Support System for classification

Let us briefly discuss the DSS in the framework of which we implement the DC classification method. The DSS is developed in the Java environment, which is platform independent (e.g. Windows, Linux, Unix). Figure 1 demonstrates the initial information input in the form of criteria with name, description and the scale that can have verbal or numerical values and classes with name

and description. The system creates the objects set as a Cartesian product of criteria scales and allows to deselect the objects that are not presented at the moment in the problem. We are also planning to add the possibility of a more natural, but time-consuming (especially for the large data sets) way for the objects representation: the input of each given object by the DM.

After the initial information input the classification method should be selected. At the moment there are possibilities in the DSS to assign objects into classes when the criteria are ordered and when not by means of SAC or DC methods. When the method is selected, the DM is asked to classify objects one by one into one of the possible classes. Figure 2 shows one of the iterations of such questioning procedure. If no classification tool is used, each object should be assigned to a class individually. By using DSS for classification the number of questions to the DM is significantly reduced when compared to the set of given objects.

When the method has run its course the results of classification can be browsed; an example of such screening is presented in Figure 3. Except for standard functions for the creation and modification of the problem by adding, removing and editing the criteria and classes sets, there are options for saving and opening the separate files with a problem model or with the DM interview or with the classification results.

#### *B. ADHD model for classification with multiple criteria decision analysis*

In order to see how the Dichotomic Classification method works for the ADHD diagnostics let us present the problem in multiple criteria analysis terms. We consider diagnostics of ADHD with PIT and ADHD with PHIT separately because the symptoms for these two diagnoses are different and they do not interact with each other (see the Appendix). The conclusion about ADHD:CT or the absence of ADHD is made based on the combination of the results from the ADHD with PIT and the ADHD with PHIT diagnostics.

Thus, we will have three stages in our decision aiding classification: 1) diagnostics of ADHD with PIT with two possible classes: either the child met the diagnostic criteria or not; 2) diagnostics of ADHD with PHIT with two possible classes: either the child met these diagnostic criteria or not; and 3) ADHD:CT with two possible classes: the child met the diagnostic criteria for both previous stages or none of them.

The criteria for the first and second stages of the decision aiding classification consists of questions about the behavior of the child for detecting ADHD with PIT and ADHD with PHIT. The values on each criterion are the same and they reflect the answers of the mother, teacher and child itself. Even though we assume the answers of mother and teacher to be more important than the child self-reporting, the clinician assists in the identification overall estimation of child's behavior. Thus, for each of nine symptoms (criteria) we compare the answers obtained from the three persons about behavior of the child (criteria scale): if the majority of the interviewees (2 or 3 from 3) agree, we assume that the symptom exists, else if only minority (0 or 1 from 3) agree we assume that the symptom does not exist. Then Cartesian product of the scales of criteria allows us to obtain the following objects set  $|A| = 2^9 = 512$ .

Thus, ADHD diagnostics according to the SIDAC for DSM-IV diagnoses of children ages 6-12 (see the Appendix) is done in three stages. At the first stage the set of nine questions with four possible answers on each of them allows the creation of the set of all possible combinations (Cartesian product) of answers – objects set A. This set is classified into two classes: ADHD with PIT either exists or not with 242 questions to the clinician by the DC method (when compared to 336 questions asked by the SAC method). The objects have been assigned in the following way: from whole set of 512 possible combinations of the symptoms, 382 do not indicate ADHD with PIT and 130 define this disorder. The resulting table is partially (due to the space limitation) presented in Figure 3. The second stage is done by analogy with the first one with the only difference that another set of nine questions are combined for evaluation of whether the child meets with the ADHD with PHIT conditions or not. Due to the similarity of the classification rule the number of questions to the DM is the same 242 when the DC method is applied and 336 when the SAC method is used. At the third stage of the decision aiding classification we aggregate the results of the two previous stages and obtain the evaluation for the ADHD:CT: 33 children from 512 meet both subtypes of criteria (ADHD with PIT and ADHD with PHIT) and 285 children do not have any of the ADHD subtypes.

## VI. CONCLUSION

In this study we presented the interactive DC method for ordinal classification of the set of objects that can have verbal or numerical evaluations on the criteria. We have also proposed the use of the ordered criteria set. When there is possibility of establishing such a relation, the classification procedure is simplified significantly. The developed method was demonstrated with a simple illustrative example.

The proposed DC method has been implemented in the Decision Support System and

compared with the previously developed SAC VDA method.

We have applied the DC method to the problem of ADHD diagnostics. These tests are done in cooperation with the Niilo Maki Institute, where there is considerable experience in solving such problems; however, the need for computerized DSS exists. The system can assist an expert as well as a new specialist in neuropsychological diagnostics.

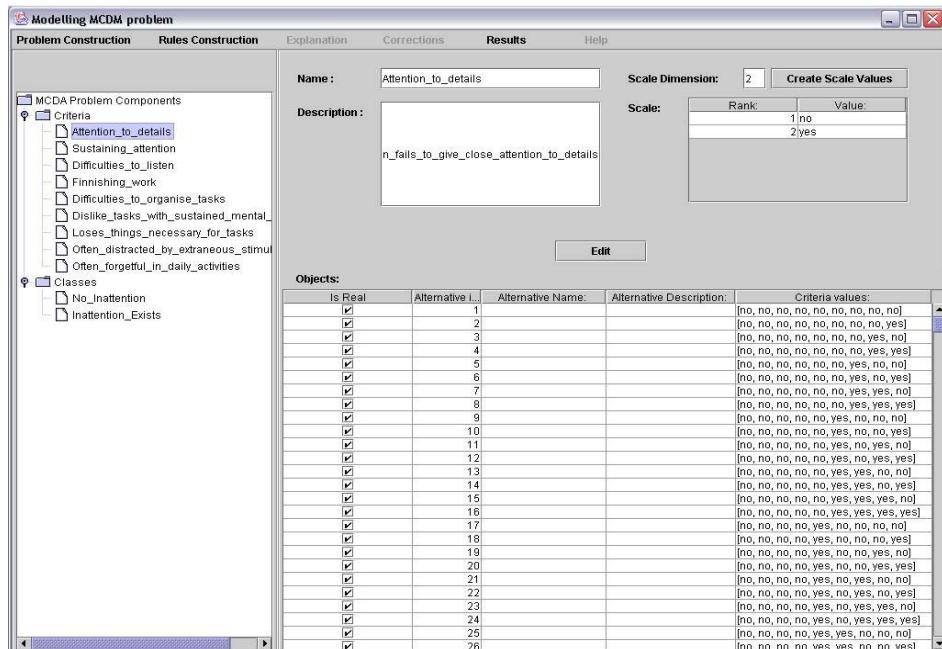


Fig. 1. Screen of the initial information input

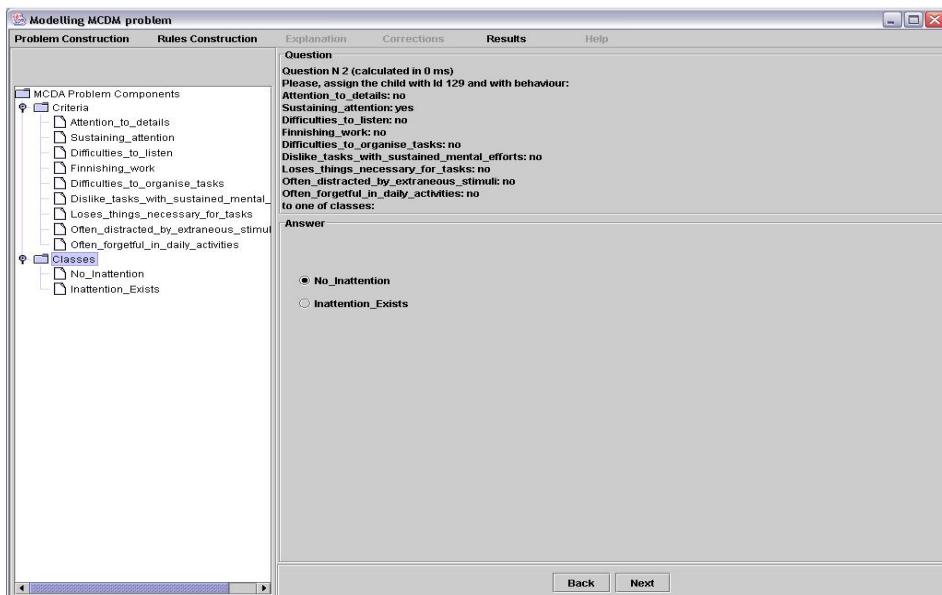


Fig. 2. Screen of the questioning procedure iteration

Modelling MCDM problem		Problem Construction	Rules Construction	Explanation	Corrections	Results	Help
		Results of Classification:					
		Sort by Id					
<b>MCDM Problem Components</b>							
Criteria							
Classes							

Predominantly Hyperactive-Impulsive Type:

At least 6 of the hyperactivity / impulsivity items and less than 6 of the inattention items endorsed; meets exclusionary criteria above.

Yes  No

314.02 Attention-Deficit / Hyperactivity Disorder: Combined Type:

At least 6 of the inattention items and at least 6 of the hyperactivity / impulsivity items endorsed; meets exclusionary criteria above.

Yes  No

## ACKNOWLEDGEMENTS

The authors would like to thank the researchers of the Scientific Computing laboratory and Optimization Group at the University of Jyväskylä for their remarks that helped to improve the paper significantly.

The research was supported by grants from the University of Jyväskylä and COMAS Graduate School at the University of Jyväskylä.

## REFERENCES

- [1] American Psychiatric Association, *Diagnostic and statistical manual of mental disorders* (4th ed.). Washington: DC, 1994.
- [2] D. S. Crystal, R. Ostrander, R. S. Chen, and G. J. August, Multimethod assessment of psychopathology among DSM-IV subtypes of children with Attention-Deficit / Hyperactivity Disorder: self-, parent, and teacher reports, *Journal of Abnormal Child Psychology*, vol. 29, no. 3, pp. 189-205, 2000.
- [3] C. Gillberg, *Clinical Child Neuropsychiatry*, Cambridge, England: Cambridge University Press, 1995.
- [4] A. C. Kalff, J. G. M. Hendriksen, M. Kroes, J. S. H. Vles, J. Steyaert, F. J. M. Feron, T. M. C. B. van Zeben, and J. Jolles, Neurocognitive performance of 5- and 6-year-old children who met criteria for Attention Deficit / Hyperactivity Disorder at 18 months follow-up: results from a prospective population study. *Journal of Abnormal Child Psychology*, vol. 30, no. 6, pp. 589-598, 2002.
- [5] R. L. Keeney and H. Raiffa, *Decisions with multiple objectives - Preferences and value tradeoffs*. New York, NY: John Wiley and Sons, 1976.
- [6] D. Yu. Kochin, "Decision support system for classification of a finite set of alternatives," Master's Thesis., Moscow Institute for Physics and Technology (Technical University). Moscow, Russia, 2002, (in Russian).
- [7] O. I. Larichev, A. I. Mechitov, V. K. Morgoev, H. M. Moshkovich, E. M. Furems, Exact duplicates of human judgments, in K. Borchering, O. Larichev, D. Messick (Eds.), *Contemporary Issues in Decision Making*, Eds. Amsterdam, The Netherlands: Elsevier Science, 1990.
- [8] O. I. Larichev, H. M. Moshkovich, Decision Support System ORCLASS. in *Proc. of the Tenth International Conference on Multiple Criteria Decision Making*, Taiwan, vol. 1, pp. 341-350, 1994.
- [9] O. I. Larichev, A study on the internal organization of expert knowledge, *Pattern Recognition and Image Analysis*, vol. 5, no. 1, pp. 57– 63, 1995.
- [10] O. I. Larichev and H. Moshkovich, *Verbal Decision Analysis for Unstructured Problems*. Boston: Kluwer, 1997.
- [11] O. I. Larichev, D. Yu. Kochin and L. L. Ustinovius, Method SNOD for Investment Project Analysis of Building Reconstruction, in *Proc. 56th EURO WG MCDA Meeting*, Coimbra, Portugal, Oct. 2002, pp. 293-313.
- [12] O. I. Larichev, A. V. Kortnev, and D. Yu Kochin, Decision support system for classification of finite set of multicriteria alternatives, *Decision Support System*, vol. 33, pp. 13-21, 2002.
- [13] O. I. Larichev, A. Asanov, and Y. Naryzhny, Effectiveness evaluation of expert classification methods, *European Journal of Operational Research*, vol. 138, pp. 260-273, 2002.
- [14] R. Robera, J. K. Penberthy, T. Loboschefski, D. Cox, and B. Kovatchev, Combined psychophysiological assessment of ADHD: a pilot study of Bayesian probability approach illustrated by appraisal of ADHD in female collage students, *Applied Psychophysiology and Biofeedback*, vol. 29, no. 1, pp.1-18, 2004.
- [15] H. A. Taha, *Operational Research: An Introduction*. 6th ed., NJ: Prentice-Hall, 1997.
- [16] R. Tannock, Attention Deficit Hyperactivity Disorder: Advances in cognitive, neurobiological, and genetic research. *Journal of the Child Psychology and Psychiatry*, vol. 39, pp. 65-99. 1998.
- [17] E. W. Weisstrain, (1999) "Bisection," MathWorld – A Wolfram Web Resources. [Online]. Available: <http://mathworld.wolfram.com/Bisection.html>

# Optimality conditions in preference-based spanning tree problems

Miguel Ángel Domínguez-Ríos, Sergio Alonso, Marcos Colebrook and Antonio Sedeño-Noda

University of La Laguna/Departamento de Estadística e Investigación Operativa

Avda. Astrofísico Francisco Sánchez, s/n,

Edificio de la Facultad de Matemáticas, 4<sup>a</sup> planta

38205 La Laguna, Santa Cruz de Tenerife, España

Email: madoming@ull.es

**Abstract**—Spanning trees problems defined in a preference-based environment are addressed. In this approach, optimality conditions for the well-known minimum-weight spanning tree problem are generalized to be used with more general preference orders. Then, necessary and/or sufficient properties of the preference relations are studied, in order to assure that the set of ‘most preferred’ trees is the set of spanning trees verifying the optimality conditions. Finally, an algorithm for the construction of the set of spanning trees fulfilling the optimality conditions is designed.

**Keywords**—Preference-based modelling; Multiobjective spanning tree problems.

## I. INTRODUCTION

OPTIMIZATION problems involving trees have been intensively studied in the literature. The minimum-weight spanning tree problem, (MST), is a classic reference and usually appears as a subproblem in other more complex studies. Its resolution using greedy approaches given mainly by Kruskal [6] or Prim [8] is well-known. For the validation of these methods, two optimality conditions are defined in the following sense: any spanning tree is minimum-weight if and only if verifies any of them. In fact, the classic algorithms mentioned above build one tree verifying any of the optimal conditions for the MST. See [1] and [2] for a complete development.

In this paper, we follow this approach to generalize the MST problem to a preference-based environment. That is, the goal is to find the maximal or most preferred trees among all spanning trees in a connected undirected graph. The preference order is given by a binary relation between pairs of subsets of edges on the graph. Multicriteria spanning tree problems can be described using different preference orders defined on vector costs. For example, the search of efficient multiobjective spanning trees in the

Pareto sense (see [4], [9], [10] and [5]) or the construction of all minimum lexicographic spanning trees.

Recently, [7] developed algorithms and proved conditions to be verified for a preference system in order to assure that, these methods really build the set of optimal solutions. This paper studies not only the spanning trees problems mentioned, but also the most-preferred paths problem on digraphs. Unfortunately, the algorithms described are inefficient because they have to check that an optimal tree is not built more than once. Moreover, the way in which the correction of the algorithm is proved along with the conditions required for the preference order is rather complex.

In the next section, some basic definitions and properties are stated. Section III generalizes the optimality conditions for the MST problem to a preference-based framework. Sections IV and V deal with the necessary and sufficient conditions for the most-preferred trees to verify the optimality conditions. Section VI describe an algorithm to build the set of spanning trees that verify the optimality conditions, and finally, the work ends with some conclusions.

## II. NOTATION, DEFINITIONS AND BASIC RESULTS

Let  $G=(V, E)$  be an undirected and connected graph. We denote by  $V(G)$  the set of  $n$  nodes of  $G$  and by  $E(G)$  the set of edges in  $G$ . A spanning tree of  $G$  is a subgraph that spans its  $n$  nodes with no cycles. We denote by  $\sigma(G)$  the set of spanning trees in  $G$ . Then, every  $T \in \sigma(G)$  is a subgraph of  $G$  that spans the set of nodes  $V(G)$  using certain subsets of edges  $E(G)$ , that is,  $T = (V(G), E(T))$ . Therefore, any  $T \in \sigma(G)$  is

completely defined through its  $n-1$  edges.

*Definition 1.* Given a pair of nodes  $i, j \in V(G)$ , and  $T \in \sigma(G)$ , we define by  $\text{path}\langle T, (i, j) \rangle$  the set of edges in  $T$  which connects the nodes  $i$  and  $j$ . Note that the path connecting a pair of nodes in a spanning tree is unique.

*Definition 2.* Given  $T \in \sigma(G)$ , we say that an edge  $f \in E(G)$  belongs to the set  $\text{cut}\langle T, e \rangle$  if connects the two connected components of  $T - \{e\}$ . That is,

$$\text{cut}\langle T, e \rangle = \{f \in E(G) / T - e + f \in \sigma(G)\}.$$

Now, we must point out a basic but important first result easy to prove.

*Proposition 1.* Given  $T \in \sigma(G)$ , for any  $e \in T$  and any  $f \in E(G)$ ,  $f \in \text{cut}\langle T, e \rangle$  iff  $e \in \text{path}\langle T, f \rangle$ .

#### A. A preference-based environment

Let us first state some definitions about preference relations in optimization problems. We will work with a binary relation  $\succ$  defined on a certain set  $X$ , verifying, at least, to be asymmetric and transitive (and then irreflexive). With these properties,  $\succ$  is a basic preference order (see [3]). Then, we call indifference  $\sim$  as the absence of strict preference  $\succ$ , that is,

$$\forall x, y \in X, x \sim y \text{ iff } (\text{not } x \succ y \text{ and not } y \succ x).$$

We will work with the union of these binary relations, that is,

$$\forall x, y \in X, x \succsim y \text{ iff } (x \succ y \text{ or } x \sim y).$$

Given a non empty subset  $X_0 \subseteq X$ , we define and denote the maximal elements in  $X_0$  as the following,

$$\max X_0 = \{x \in X_0 / \forall y \in X_0, \text{if } y \succsim x \text{ then } y \sim x\}$$

Therefore, given a connected and undirected graph  $G=(V, E)$ , with a basic preference relation  $\succ$  defined on the set of subsets of edges  $E(G)$ , the maximal or most-preferred spanning tree problem is solved with the construction of the set  $\max \sigma(G)$ .

The MST problem can be stated as a preference-based spanning tree problem. Given the weight function  $w$  for the problem defined for each edge in  $E(G)$ , if we extend this definition to any subset of edges,  $E_0 \subseteq E(G)$ , as

$$w(E_0) = \sum_{e \in E_0} w(e),$$

the preference order we need is,

$$E_0, E_1 \subseteq E(G), E_0 \succ E_1 \text{ iff } w(E_0) < w(E_1).$$

Therefore any  $T \in \max \sigma(G)$  is a minimum weight spanning tree.

Moreover, if the weight function for each edge is defined in  $\mathbb{R}^k$ , the set  $\max \sigma(G)$  can be the set of lexicographical-minimum spanning trees if we use the lexicographic order or the set of efficient spanning trees if we use the Pareto order.

#### B. Optimality conditions

The optimality conditions for the MST problem are the following:

*OptCUT condition.* Given a  $T \in \sigma(G)$ , we say that  $T$  verifies the *optimality cut condition* iff  $\forall e \in T, w(e) \leq w(f), \forall f \in \text{cut}\langle T, e \rangle$ . We denote by  $\text{OptCUT}(G)$  the set of these spanning trees.

*OptPATH condition.* Given a  $T \in \sigma(G)$ , we say that  $T$  verifies the *optimality path condition* iff  $\forall f \in E(G), w(e) \leq w(f), \forall e \in \text{path}\langle T, f \rangle$ . We denote by  $\text{OptPATH}(G)$  the set of these spanning trees.

For the MST problem, it is well known that,

$$\max \sigma(G) = \text{OptCUT}(G) = \text{OptPATH}(G), \quad (1)$$

and algorithms as Kruskal or Prim, really build one tree in  $\text{OptCUT}(G)$  or  $\text{OptPATH}(G)$ , respectively.

We need an appropriate generalization of these optimality conditions in a preference system framework, to study the conditions under which equality (1) is still valid.

### III. OPTIMALITY CONDITIONS GENERALIZED.

Given an undirected and connected graph  $G=(V, E)$ , with a preference basic order  $\succ$  defined on the subsets of  $E(G)$ , a correct generalization of *OptCUT* and *OptPATH* properties must maintain the relation with the optimality conditions of the MST problem. We can consider the following as a valid generalization:

*OptCUT condition.* Given a  $T \in \sigma(G)$ , we say that  $T$  verifies the *generalized optimality cut*

condition iff  $\forall e \in T, \forall f \in \text{cut}\langle T, e \rangle$ , if  $f \succsim e^*$  then  $f \sim e$ . We denote by  $\text{OptCUT}(G)$  the set of these spanning trees.

*OptPATH condition.* Given a  $T \in \sigma(G)$ , we say that  $T$  verifies the *generalized optimality path condition* iff  $\forall f \in E(G), \forall e \in \text{path}\langle T, f \rangle$ , if  $f \succsim e$  then  $f \sim e$ . We denote by  $\text{OptPATH}(G)$  the set of these spanning trees.

Note that the definitions given for the MST problem are now particular cases of them, so we keep the same names of the conditions.

Finally, we state that, based in *proposition 1*, the next and obvious result holds.

*Proposition 2.* Given an undirected and connected graph  $G=(V, E)$ ,  $\text{OptCUT}(G)=\text{OptPATH}(G)$ .

#### IV. NECESSARY CONDITIONS.

In this section, we are going to study the conditions required for the basic preference order to assure that any spanning tree  $T \in \max\sigma(G)$ , verifies  $\text{OptCUT}$  (or  $\text{OptPATH}$ ).

*Proposition 3.* Given  $E_0 \subseteq E(G)$  and  $e, f \in E(G) - E_0$ , if a preference basic order  $\succ$  in  $\sigma(G)$  satisfies both of the following:

- (i) if  $e \succsim f$  then  $E_0 \cup \{e\} \succsim E_0 \cup \{f\}$ , (*additivity*), and,
- (ii) if  $E_0 \cup \{e\} \sim E_0 \cup \{f\}$  then  $e \sim f$ , (*simplification*).

then  $\max\sigma(G) \subseteq \text{OptCUT}(G)=\text{OptPATH}(G)$ .

*Proof.* Let  $T \in \max\sigma(G)$ ,  $e \in T$  and  $f \in \text{cut}\langle T, e \rangle$ . Thus if  $T = T_0 \cup \{e\}$ , then  $T' = T_0 \cup \{f\} \in \sigma(G)$ . Suppose that  $f \succsim e$ , then  $T_0 \cup \{f\} \succsim T_0 \cup \{e\}$  applying *additivity*; that is,  $T' \succsim T$ , and as  $T \in \max\sigma(G)$ ,  $T' \sim T$ ; thus,  $T_0 \cup \{f\} \succsim T_0 \cup \{e\}$ , and then, applying *simplification*, we have  $f \sim e$ , and the result follows  $\square$ .

Both properties, *additivity* and *simplification*, hold

for several multicriteria preference orders. The lexicographical order and the preference order in the Pareto sense are examples. In these cases, the weight of any subset of edges is obtained by the sum in  $\mathbb{R}^k$  of the vector costs of their edges, that is, the vectorial sum. In summary, both preference orders are based in the next common definition. Given  $E_0, F_0 \subseteq E(G)$ ,

$$E_0 \succ F_0 \text{ iff } w(E_0) = \sum_{e \in E_0} w(e) \succ w(F_0) = \sum_{e \in F_0} w(e).$$

Moreover, both verify,

$$w(e) \succ w(f) \text{ iff } w(e) + w(E_0) \succ w(f) + w(E_0),$$

and then,

$$w(e) \sim w(f) \text{ iff } w(e) + w(E_0) \sim w(f) + w(E_0),$$

for any  $E_0 \subseteq E(G)$  and  $e, f \in E(G) - E_0$ .

These necessary conditions are, however, not sufficient to assure that any tree verifying  $\text{OptCUT}$  (or  $\text{OptPATH}$ ) is maximal.

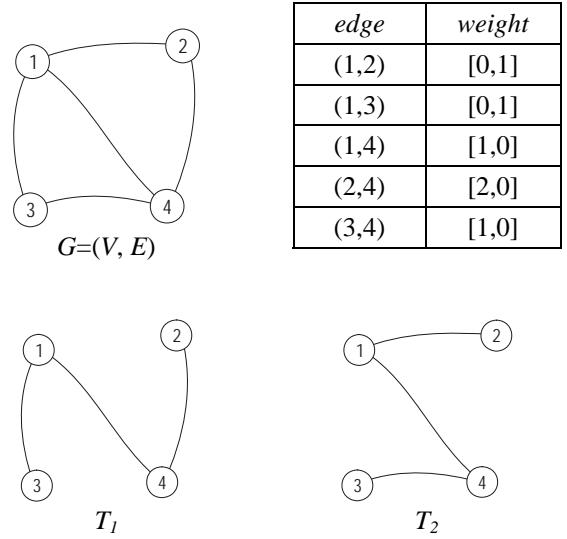


Figure 1. A counterexample.

Given  $G=(V, E)$  illustrated in figure 1, the table with a biojective cost for each edge in  $E(G)$  is shown on the right. We study the set  $\max\sigma(G)$  using the preference relation based on the Pareto order. Table 1 lists, for each edge  $e$  in  $T_1$ , the edges in the cuts and, respectively, its weights. Note that  $T_1 \in \text{OptCUT}(G)$ . However  $T_1 \notin \max\sigma(G)$ , being  $w(T_1) = (3,1)$  dominated by  $w(T_2) = (2,1)$ .

\* We relax the correct notation  $\{f\} \succsim \{e\}$  by  $f \succsim e$ , when the subsets of edges are unitary

$e$ in $T_1$	$\text{cut}\langle T_1, e \rangle$	weights
(1,3)	{(1,3),(3,4)}	{[0,1],[1,0]}
(1,4)	{(1,2),(1,4),(3,4)}	{[0,1],[1,0],[1,0]}
(2,4)	{(1,2),(2,4)}	{[0,1],[2,0]}

Table 1. Cuts and weights in the counterexample.

## V. SUFFICIENT CONDITIONS.

Necessary conditions have to be enriched to assure that any spanning tree in  $\text{OptCUT}(G)$  belongs to  $\max \sigma(G)$ .

*Definition 2.* A preference basic order is said to verify the *strong additivity property* when given  $E_0 \subseteq E(G)$  and  $e, f \in E(G) - E_0$ ,

$$e \succ f \text{ iff } E_0 \cup \{e\} \succ E_0 \cup \{f\}.$$

Note that *strong additivity* implies *additivity* and *simplification*.

Suppose now a preference basic order holding the *strong additivity property*. Let  $T \in \text{OptCUT}(G)$

Suppose now that, using the same previous conditions,  $T \in \text{OptCUT}(G)$  and  $T' \in \sigma(G)$  differ in more than one edge, and  $T' \succsim T$ . Then, we can separate the common edges from the others and so,  $T' = T_0 \cup T_f$  and  $T = T_0 \cup T_e$ . Given  $e_0 \in T_e$ , there is an edge  $f_0 \in T_f$  such that  $f_0 \in \text{cut}\langle T, e_0 \rangle$  (see [1]). As  $T \in \text{OptCUT}(G)$ ,  $e_0 \succsim f_0$  and then  $(T_0 \cup T_f - \{f_0\}) + \{e_0\} \succsim (T_0 \cup T_f - \{f_0\}) + \{f_0\}$ , that is,  $(T_0 \cup T_f - \{f_0\}) + \{e_0\} \succsim T_0 \cup T_f \succsim T_0 \cup T_e$ . We can choose another edge  $e_1 \in T_e - \{e_0\}$ , and the corresponding  $f_1 \in T_f - \{f_0\}$  such that  $f_1 \in \text{cut}\langle T, e_1 \rangle$ . Then as  $T \in \text{OptCUT}(G)$ ,  $e_1 \succsim f_1$ , and  $(T_0 \cup T_f - \{f_0, f_1\} + \{e_0\}) + \{e_1\} \succsim (T_0 \cup T_f - \{f_0, f_1\} + \{e_0\}) + \{f_1\}$ , that is,  $T_0 \cup T_f - \{f_0, f_1\} + \{e_0, e_1\} \succsim T_0 \cup T_f - \{f_0\} + \{e_0\} \succsim T_0 \cup T_f \succsim T_0 \cup T_e$  if we complete the preference chain. Hence, we can continue replacing any edge  $f_k \in T_f$  by the corresponding edge in  $e_k \in T_e$  such that  $f_k \in \text{cut}\langle e_k, T \rangle$ , until we obtain the closed preference chain,

$$T_0 \cup T_e \succsim \dots \succsim T_0 \cup T_f - \{f_0, f_1\} + \{e_0, e_1\} \succsim T_0 \cup T_f - \{f_0\} + \{e_0\} \succsim T_0 \cup T_f \succsim T_0 \cup T_e \quad (2)$$

*Proposition 4.* Let  $G=(V, E)$  be a connected and undirected graph with a preference basic order  $\succ$  defined on the subsets in  $E(G)$  verifying the *strong additivity property*. If  $\succ$  is negative transitive then  $\max \sigma(G) = \text{OptCUT}(G) = \text{OptPATH}(G)$ .

*Proof.* We only have to prove that  $\text{OptCUT}(G) \subseteq \max \sigma(G)$ . If  $\succ$  is negative transitive, then  $\sim$  and  $\succsim$  are transitive, and, in the conditions of (2), we have,  $T_0 \cup T_e \succsim T_0 \cup T_f \succsim T_0 \cup T_e$ , and thus,  $T' \sim T \square$ .

and  $T' \in \sigma(G)$  such that both spanning trees differ in only an edge. If  $T' \succsim T$ , we can denote  $T = T_0 \cup \{e\}$  and  $T' = T_0 \cup \{f\}$ , and then,  $T_0 \cup \{f\} \succsim T_0 \cup \{e\}$ . Thus, using the *strong additivity property* we can deduce that  $f \succsim e$ , because:

- if  $T_0 \cup \{f\} \succ T_0 \cup \{e\}$  then directly  $f \succ e$ , and
- if  $T_0 \cup \{f\} \sim T_0 \cup \{e\}$ , then either  $e \succ f$  or  $f \succ e$  cannot be possible, so  $f \sim e$ .

As  $T \in \text{OptCUT}(G)$  and  $f \in \text{cut}\langle T, e \rangle$ , from  $f \succsim e$  we can deduce  $e \sim f$  and then  $T' \sim T$ . However, we already know that *strong additivity property* is not enough to generalize this result to spanning trees which differ in more than one edge. The preference order in the Pareto sense verifies this property and  $T_1 \in \text{OptCUT}(G)$  and  $T_2 \in \max \sigma(G)$  differs in two edges.

Both lexicographic and Pareto orders verify the *strong additivity property*, but only the indifference of the lexicographic one is transitive.

As a final remark we want to mention that the *strong additivity property* plays here an analogous role to the denominated *Independence Axiom* in [7]. It can be written in the following form,  $\forall A, B, C \subseteq E(G)$ , such that,  $C \cap (A \cup B) = \emptyset$ , if  $A \succ B$  then  $A \cup C \succ B \cup C$ .

## VI. AN ALGORITHM FOR $OPTCUT(G)$ .

In this final section, we design an algorithm that compute the set  $OptCUT(G)$  (and then,  $OptPATH(G)$ ) when the preference order verifies certain requirements.

For the construction of spanning trees in an undirected and connected graph  $G=(V, E)$ , there are two main approaches. *Kruskal's* approach selects an edge as a valid candidate if it keeps acyclic the subset of edges. A general scheme is:

---

```
T=∅ ;
while |T|<n-1 do
    select an edge e such that T ∪ {e} is
    acyclic;
    T=T ∪ {e};
```

---

Nevertheless, *Prim's* approach expands the connectivity from an arbitrary node  $i$  until the rest of nodes are reached. A general scheme is:

---

```
T=∅ ;
M={i} ;
while |T|<n-1 do
    select an edge e with an extreme in M
    and the other, j, in V(G)-M;
    T=T ∪ {e};
    M=M ∪ {j};
```

---

The success of these approaches relies on the fact that an efficient implementation of both selection processes is available. Moreover, if the edge to be added is also, the one with minimum weight, then we have Kruskal and Prim algorithm for the MST problem. The adaptation of these strategies to a more general preference-based order must be carried out carefully. Note that, these algorithms for the MST problem build *only one* minimum-weight spanning tree, even if more than one is possible. In a preference-based framework, every tree in the optimal set  $\max \sigma(G)$  has to be computed in an efficient way. In [7], the algorithms have to check if a partial/total solution is obtained several times since they add edges one by one. That is, if edges in  $\{e_1, e_2\}$  belong to a tree, the sequence of addition  $\{e_1\} + \{e_2\}$  leads to the same set even if the order is  $\{e_2\} + \{e_1\}$ . A more efficient algorithm should allow adding more than one edge in every step.

The following result needs no proof.

*Proposition 5.* Given a basic preference order  $\succ$  in  $\sigma(G)$ , for any subset  $X \subseteq E(G)$ , if  $\sim$  is transitive, then all edges in  $\max X$  are indifferent among them, that is,  $\forall e, e' \in \max X, e \sim e'$ . Moreover, if there are edges  $f \in E(G)$  and  $e_0 \in \max X$ , such

that  $f \succ e_0$  or  $e_0 \succ f$  then, respectively,  $f \succ e$  or  $e \succ f$  for any  $e \in \max X$ .

We will use the graph in figure 2 to illustrate the performance of the algorithm described in the next section.

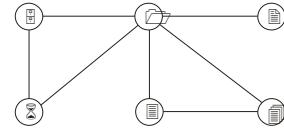


Figure 2. A grid graph as example

In Table 2, the transitive preference order  $\sim$  among edges is defined. Edges in the same row are indifferent, while edges in a higher row are preferred to those in a lower one.

(3,4)
(1,3)(1,4)(1,5)(1,6)
(5,6)(1,2)

Table 2. Preference order in the example.

In Kruskal's approach, the selection of edges must keep the subset  $T$  with no cycles. Then, an edge can be selected if and only if its extremes merge different connected components in  $T$ . Note that several edges can join the same pair of connected components in  $T$ , and therefore, they all belong to the same cut in the spanning trees containing  $T$ . Therefore, more than one edge can be selected, but only if they join different pairs of connected components and we do not exceed  $n - 1$  edges. Thus, given a basic preference order  $\succ$  in  $\sigma(G)$ , if  $\sim$  is transitive, a correct method to build the set  $OptCUT(G)$  is the following.

---

```
procedure optimalcutset(T, k)
Let E0=max{e∈E(G)/T ∪ {e} is acyclic};
Let k0 the number of connected
components in T ∪ E0;
For each subset T0 of cardinal k-k0 in
E0 being T ∪ T0 acyclic do
    if k0=1 then
        T ∪ T0 ∈ OptCut(G)
    else optimalcutset(T ∪ T0, k0).  


```

---

The main call of the algorithm is **optimalcutset**(∅,n).

The algorithm is now used to obtain the  $OptCUT$  set for the graph described in Figure 2 and Table 2.



Initial phase: no edges added and each node is a unitary connected component, so  $k_0=n=6$ .



The set  $E_0$  only contains the edge (3,4); then  $k_0=5$  and  $T_0=\{(3,4)\}$ .



The set  $E_0=\{(1,3),(1,4),(1,5),(1,6)\}$  and  $k_0=2$ ; there are two suitable subsets  $T_0$ ,  $\{(1,3),(1,5),(1,6)\}$  and  $\{(1,4),(1,5),(1,6)\}$



Finally the edge (1,2) is added.

We remark that, if the preference order  $\succ$  in  $\sigma(G)$  also satisfies the *strong additivity property*, we have  $OptCUT(G) = \max \sigma(G)$ . The *optimalcutset* algorithm is a generalization of *AllMST* algorithm for the MST problem in [9].

## VII. CONCLUSIONS.

This paper generalizes the two optimality conditions used for solving the MST problem as an algorithmic tool to solve preference-based spanning tree problems. Both optimality conditions generalized are proved to be equivalent and the subsets of spanning trees which satisfy them are the same.

Then, the main interest is to study necessary and sufficient conditions that preference orders have to be verified to assure that the ‘most preferred’ trees are those that fulfil the optimality conditions. We prove that the preference order is required to be ‘negative transitive’ and to verify the denominated ‘strong additivity property’ to achieve this result.

Finally, we also design an algorithm for building efficiently the subset of spanning trees which satisfy the generalized optimality cut condition. Therefore, this algorithm can be used, for example, to build the set of lexicographical-minimum spanning trees of a connected graph.

## ACKNOWLEDGEMENTS

The authors would like to thank the anonymous referees all their valuable comments that have

improved the paper.

## REFERENCES

- [1] R. K. Ahuja, T. L. Magnanti and J. B. Orlin, *Network Flows. Theory, Algorithms and Applications*. Prentice Hall, 1993.
- [2] T. H. Cormen, C. E. Leiserson and R. L. Rivest, *Introduction to Algorithms*. The MIT Press, 1990.
- [3] P. C. Fishburn, *Utility Theory for Decision Making*, Publications in Operations Research, no. 18. Wiley & Sons, 1970.
- [4] H. W. Hamacher and G. Ruhe, “On spanning tree problems with multiples objectives”, in *Annals of Operations Research*, 52, pages 209-230, 1994.
- [5] J. D. Knowles and D. W. Corne, “Enumeration of Pareto optimal multi-criteria spanning trees – a proof of the incorrectness of Zhou and Gen’s proposed algorithm”, in *European Journal of Operational Research*, 143(3), pages 543-547, 2002.
- [6] J. B. Kruskal, “On the shortest spanning subtree of a graph and the traveling salesman problem”, in *Proceeding of the American Mathematical Society*, 7, pages 48-50, 1956.
- [7] P. Perny and O. Spanjaard, “A preference-based approach to spanning trees and shortest path problems”, in *European Journal of Operational Research*, vol. 162, issue 3, pages 584-601, May 2005.
- [8] R. C. Prim, “Shortest connection networks and some generalization” in *Bell System Technical Journal*, 36, pages 1389-1401, 1957.
- [9] R. M. Ramos, S. Alonso, J. Sicilia and C. González, “The problem of the optimal biobjective spanning tree” in *European Journal of Operational Research*, vol. 111(3), pages 617-628, 1998.
- [10] G. Zhou and M. Gen, “Genetic algorithm approach on multi-criteria minimum spanning tree problem”, in *European Journal of Operational Research*, 114(1), pages 141-152, 1999.

# Robust 1-median location problem on a tree

Rim KALAI<sup>\*</sup>, Mohamed Ali ALOULOU<sup>\*</sup>, Philippe VALLIN<sup>\*</sup> and Daniel VANDERPOOTEN<sup>\*</sup>

<sup>\*</sup>Paris-Dauphine University/LAMSADE

Place du Maréchal de Lattre de Tassigny, 75775 Paris Cedex 16

Email: kalai,aloulou,vallin,vdp@lamsade.dauphine.fr

**Abstract**—In combinatorial optimization, and particularly in location problems, the most used robustness criteria rely either on maximal cost or on maximal regret. However, it is well known that these criteria are too conservative. In this paper, we present a new robustness approach, called *lexicographic  $\alpha$ -robustness*, which compensates for the drawbacks of the criteria based on the worst case. We apply this notion to the 1-median location problem under uncertainty and we give a polynomial algorithm to determine robust solutions in the case of a tree graph.

**Keywords**—Robustness, 1-median location problem, minmax cost/regret, scenario-based uncertainty.

## I. INTRODUCTION

**R**OUSTNESS analysis looks for solutions in a context where the imprecise, uncertain and generally badly known parameters of a problem make inappropriate the search of optimal solutions [19] [23]. Unlike deterministic or stochastic approaches which are aimed at determining the best solution for a certain instance of values (or scenario), robust approaches try to find a solution or a set of solutions that is acceptable for any considered scenario. In combinatorial optimization and particularly in location problems, the most used robustness criteria rely either on the maximal cost or on the maximal regret [14]: a robust solution is one that minimizes the maximal cost or regret among all scenarios. Nevertheless, grasping the notion of robustness through only one measure (the maximal cost or regret) is questionable, since this often leads to favor only the worst case scenario. Furthermore, no tolerance is considered in this measure.

These two drawbacks of the criteria founded on the worst case suggest considering alternative robustness criteria. In the case of deterministic public location problems, Ogryczak departs from considering only the worst case by introducing the notion of *lexicographic minimax* [16]. In this paper, we use and extend this idea in order to define a new robustness approach when the set of scenarios is finite and propose a polynomial time

algorithm to compute the corresponding robust solutions for the 1-median problem on a tree.

Our paper is organized as follows. In section 2, we define the *1-median problem* and review the main works on robustness for this problem. In section 3, we introduce a relation called  *$\alpha$ -leximax*, and use it to define a set of robust solutions. We also present a general algorithm which computes this robust set when the set of solutions is finite and apply it to the *vertex 1-median problem*. In section 4, we apply our robustness approach to the *1-median problem* on a tree for which we present a specific algorithm that finds the robust points of the tree. In a final section, we summarize the important points of this work and suggest some perspectives.

## II. LITTERATURE REVIEW

Network location problems are aimed at locating new facilities in order to meet the demand of a certain number of customers [8]. Demand and travel between demand sites and facilities are assumed to occur only on a graph  $G = (V, E)$  composed of a set  $V = \{v_i, i = 1, \dots, n\}$  of  $n$  nodes (or vertices) and a set  $E$  of  $m$  edges. The length of each edge  $(v_i, v_j)$ , i.e. the distance between site  $v_i$  and site  $v_j$ , is denoted  $c_{ij}$ . We assume that demands occur only at the nodes of the network and that they can be characterized by a weight vector  $W = (w_1, w_2, \dots, w_n)$  where  $w_i$  is the weight associated with node  $v_i$  for  $i = 1, \dots, n$ .

The *absolute 1-median problem* is to locate the *absolute median* of a graph  $G$ , that is the point of  $G$  which minimizes the total weighted distance to all nodes of the graph. A *point* of the graph corresponds either to a node or to any point on an edge. Let us denote  $d(a, b)$  the minimum distance between two points  $a$  and  $b$  of  $G$ . The 1-median problem is formulated as follows :

$$\min_{x \in G} C(x) = \sum_{i=1}^n w_i d(x, v_i) \quad (1)$$

The Hakimi property stipulates that an absolute median of a graph is always at a vertex of the graph [12].

The absolute 1-median problem is then equivalent to the *vertex 1-median problem* which can be written as :

$$\min_{v \in V} C(v) = \sum_{i=1}^n w_i d(v, v_i) \quad (2)$$

Consequently, given all distances  $d(v_i, v_j)$ , the problem can be solved in linear time by enumerating and evaluating the  $n$  possible solutions.

Deterministic approaches assume that the problem parameters (node weights and edge lengths) are fixed and well known. In practice, however, it often appears difficult to determine in a reliable and irrevocable way all the data of a given problem. The decision-maker is often confronted with uncertainty that makes the deterministic reasoning inappropriate. Uncertainty situations are divided into two classes: if it exists a perfectly known probability distribution on the set of the nature states, we are in a *risk situation*. Otherwise, it is impossible to allocate probabilities to the possible outcomes of a decision and we say that we are in an *uncertainty* (or *true uncertainty*) *situation*. The latter case arises, for example, when the outcome of a decision may depend on a simultaneous or subsequent decision of a competitor whose objectives conflict with one's own, or on future external events of non-repeatable variety, for which the estimation of probabilities is a dubious exercise [18]. Robustness analysis concerns uncertainty situations.

Let us assume that the node weights and the edge lengths can take many different values and that there is a (finite or infinite) set  $S$  of possible scenarios (possible values of the parameters). For a given scenario  $s$  and a point  $x$  of  $G$ , the cost function under scenario  $s$  is defined as follows:

$$C^s(x) = \sum_{i=1}^n w_i^s d^s(x, v_i) \quad (3)$$

where  $w_i^s$  and  $d^s(a, b)$  denote respectively the weight of node  $v_i$  and the minimum distance between points  $a$  and  $b$  under scenario  $s$ . The regret of solution  $x$  (also called *opportunity loss* or *absolute deviation* [14]) is the difference between the cost of  $x$  under scenario  $s$  and the cost of the best solution under the same scenario:

$$R^s(x) = C^s(x) - C^s(x^{*s}) \quad (4)$$

where  $x^{*s}$  is the optimal solution of the 1-median problem under scenario  $s$ .

To determine the robust solutions for the 1-median problem, authors often attempted to optimize the worst case performance of the system by minimizing the maximal cost or the maximal regret (see [2], [4], [5],

[7], [14] and [22]). The *minmax 1-median problem* is defined as follows:

$$\min_{x \in G} \max_{s \in S} C^s(x) \quad (5)$$

and the *minmax regret 1-median problem* has the following expression :

$$\min_{x \in G} \max_{s \in S} R^s(x) = \min_{x \in G} \max_{s \in S} (C^s(x) - C^s(x^{*s})) \quad (6)$$

In the literature on minmax (regret) 1-median problem, the authors distinguish many models according to the graph structure (tree, network), the location sites (on nodes or on edges) as well as the nature of the scenario set. Indeed, uncertainty on a parameter may be modelled either as a discrete set of scenarios, or as an interval data. Figure 1 summarizes the main results with regard to the minmax regret 1-median problem on a tree, the uncertainty being on weights (assumed to be positive). When weights can be negative and are represented by uncertainty intervals, Burkard and Dollani give an algorithm in  $O(n^2)$  for the problem on a tree [6]. As for the minmax regret 1-median problem on a general network (with uncertainty on weights), Averbakh and Berman present in [4] two approaches in  $O(nm^4)$  time and  $O(mn^2 \log n)$  time for the absolute problem (location anywhere on the graph), the vertex problem having an order of complexity of  $O(n^3)$ . When edge lengths are uncertain, Chen and Lin [7] show that, in the case of a tree graph and interval data, the problem can be reduced to the deterministic problem under the scenario with maximal lengths. On the other hand, on a general network, the problem with uncertain lengths becomes NP-hard [2].

It is generally admitted that minmax cost and minmax regret criteria are too conservative since they are based only on the worst case. Besides, the worst case performance is often reached for a scenario with a small likelihood of occurrence, especially when uncertainty is represented by intervals. To remedy the conservatism of the minmax regret model for the *p-median* problem ( $p$  is the number of facilities to locate), Daskin et al [9] introduce a new variant of this problem called  *$\alpha$ -reliable p-minimax regret* problem. In this model, the decision-maker associates a probability with each scenario. The model then selects a subset of scenarios whose collective probability of occurrence is at least some user-specified value  $\alpha$  ( $0 \leq \alpha \leq 1$ ) which is called *reliability level*. The model identifies the solution that minimizes the maximum regret with respect to the chosen subset of scenarios. An appropriate choice of  $\alpha$  guarantees that the solution is not based on a scenario with a very small likelihood of occurrence.

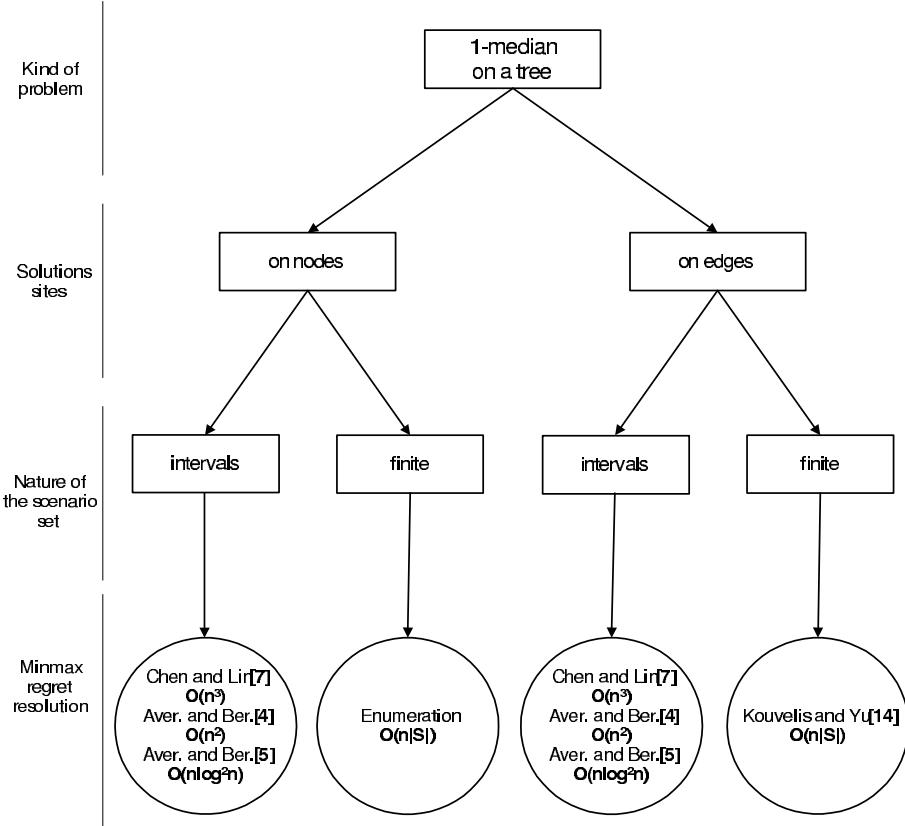


Fig. 1. Minmax regret 1-median problem on a tree with uncertainty on node weights

In a recent work, Snyder and Daskin [21] present the *stochastic p-robust P-median problem* (*p-SPMP*) ( $P$  is the number of facilities to locate). They use a measure called *p-robustness* which was first introduced by Kouvelis et al in [13]. This measure imposes a constraint dictating that the cost under each scenario must be within  $(100 + p)\%$  of the optimal cost for that scenario, where  $p \geq 0$  is an external parameter (completely independent of  $P$  the number of facilities). Moreover, the authors assign a probability to each scenario. Thus, they build a new robustness measure consisting in determining the *p*-robust solutions which minimize the expected-cost. Snyder and Daskin prove that *p-SPMP* is NP-hard and discuss a mechanism for detecting infeasibility since *p*-robust solutions may not exist (especially for small values of  $p$ ).

The main drawback of these last two approaches is that they require to express a probability for each scenario.

According to this review, we can distinguish two families of approaches to find robust solutions for a given problem. The first family looks for solutions which optimize a certain objective function (e.g. minmax approaches) whereas the second one imposes conditions that solutions must satisfy in order to be considered as

robust (e.g.  $p$ -robustness). In the following, we define a new robustness approach which belongs to the second family of approaches.

### III. DEFINITION OF A NEW ROBUSTNESS APPROACH

Let us suppose that, for a given problem, one (or several) of the parameters cannot be determined in a certain and definite way and that there is a finite set  $S$  of scenarios. Let  $X$  denote the set of feasible solutions and  $q$  the number of scenarios. Since the reasoning and the results are valid for costs as for regrets, we use in what follows the term “cost” and the notation  $C$  indifferently for cost and regret. A robust solution according to the maximal cost criterion is a solution that verifies:

$$\min_{x \in X} \max_{s \in S} C^s(x) \quad (7)$$

In the next subsections, we introduce a new preference relation that we call  *$\alpha$ -leximax* and use it to define a set of robust solutions.

#### A. The *$\alpha$ -leximax* relation

Let  $x$  be a solution of  $X$ . We associate to  $x$  a cost vector denoted by  $C(x) = (C^{s^1}(x), \dots, C^{s^q}(x))$  where  $C^{s^j}(x)$  is the cost of solution  $x$  under scenario  $s^j$ ,

$1 \leq j \leq q$ . By ordering the coordinates of  $C(x)$  in a non-increasing order, we get a vector  $\hat{C}(x)$  called *disutility vector* [15]. We have  $\hat{C}^1(x) \geq \hat{C}^2(x) \geq \dots \geq \hat{C}^q(x)$ . Thus,  $\hat{C}^j(x)$  is the  $j^{th}$  largest cost of  $x$ .

**Definition 1:** Let  $x$  and  $y$  be two solutions of  $X$ ,  $\hat{C}(x)$  and  $\hat{C}(y)$  the associated disutility vectors. The *leximax* relation, denoted by  $\succsim_{lex}$ , is defined as follows [11]:

$$\begin{aligned} x \succ_{lex} y &\Leftrightarrow \exists k \in \{1, \dots, q\} : \hat{C}^k(x) < \hat{C}^k(y) \\ \text{and } \forall j \leq k-1, &\hat{C}^j(x) = \hat{C}^j(y) \end{aligned}$$

$x$  is said to be (strictly) preferred to  $y$  in the sense of the *leximax* relation.

$$x \sim_{lex} y \Leftrightarrow \forall k \in \{1, \dots, q\}, \hat{C}^k(x) = \hat{C}^k(y)$$

$x$  and  $y$  are said to be equivalent in the sense of the *leximax* relation.

In other words, comparing two cost vectors in the sense of the *leximax* relation is equivalent to comparing the first distinct coordinates of the disutility vectors. Remark that reordering cost vector implies that we implicitly assume that the vector obtained by the permutation of the cost vector coordinates is equivalent to the original cost vector (the *leximax* relation is said to be *anonymous* [16]). This is justified by the fact that, in a situation of true uncertainty, none of the scenarios can be distinguished. The *leximax* relation is complete, reflexive and transitive. Therefore, it is a *weak order*.

The previous definition of the *leximax* relation requires a perfect equality between the disutility vector coordinates of two solutions in order to consider them equivalent. Nevertheless, in practice, it may exist a tolerance threshold under which the decision-maker either cannot perceive the difference between two elements, or refuse to give his opinion on the preference for one of them [20]. Taking an indifference threshold  $\alpha$  into account leads to the following definition:

**Definition 2:** Let  $x$  and  $y$  be two solutions of  $X$ ,  $\hat{C}(x)$  and  $\hat{C}(y)$  the associated disutility vectors, and  $\alpha$  a positive real value. The  $\alpha$ -*leximax* relation, denoted by  $\succsim_{lex}^\alpha$ , is defined as follows :

$$\begin{aligned} x \succ_{lex}^\alpha y &\Leftrightarrow \exists k \in \{1, \dots, q\} : \hat{C}^k(x) < \hat{C}^k(y) - \alpha \\ \text{and } \forall j \leq k-1, &|\hat{C}^j(y) - \hat{C}^j(x)| \leq \alpha \end{aligned}$$

$x$  is said to be (strictly) preferred to  $y$  in the sense of the  $\alpha$ -*leximax* relation.

$$x \sim_{lex}^\alpha y \Leftrightarrow \forall k \in \{1, \dots, q\}, |\hat{C}^k(y) - \hat{C}^k(x)| \leq \alpha$$

$x$  and  $y$  are said to be indifferent in the sense of the  $\alpha$ -*leximax* relation.

The  $\alpha$ -*leximax* relation is a lexicographic aggregation of semiorders. It is known that, for  $\alpha \neq 0$ , such an aggregation is not a semiorder [17] (for  $\alpha = 0$ , the relation resulted from the aggregation is none other than the weak order *leximax*). Actually, neither its asymmetric part  $\succ_{lex}^\alpha$  nor its symmetric part  $\sim_{lex}^\alpha$  are transitive, which may lead to preference cycles.

**Example 1:**  $\hat{C}(x) = (3, 3, 2), \hat{C}(y) = (5, 0, 0), \hat{C}(z) = (4, 3, 0)$  and  $\alpha = 1$ .

We have  $x \succ_{lex}^\alpha y$  and  $y \succ_{lex}^\alpha z$  but  $z \succ_{lex}^\alpha x$ .

Despite its failure to comply with some properties, the  $\alpha$ -*leximax* preference relation remains suitable for the determination of robust solutions since it takes into account several measures (costs under different scenarios), offers some tolerance (indifference threshold) and takes into account, at least initially, the solutions given by the minmax criteria (cost/regret).

### B. Lexicographic $\alpha$ -robust solutions

We want to determine the set of robust solutions by relying on the  $\alpha$ -*leximax* relation. As noticed at the end of section II, there are two families of robustness approaches: the first family looks for solutions given by optimizing a chosen criterion and the second one imposes some robustness properties that solutions must satisfy. Let  $x^*$  be an ideal solution (most of the time fictitious) such that:

$$\hat{C}(x^*) = (\hat{C}^1(x_1^*), \hat{C}^2(x_2^*), \dots, \hat{C}^q(x_q^*)) \quad (8)$$

where  $\hat{C} = (\hat{C}^1, \hat{C}^2, \dots, \hat{C}^q)$  is the disutility vector and  $x_k^* = \arg \min_{x \in X} \hat{C}^k(x)$  for all  $k \in \{1, \dots, q\}$ . Let us consider the following set:

$$\begin{aligned} A(\alpha) &= \{x \in X : \text{not}(x^* \succ_{lex}^\alpha x)\} \\ &= \{x \in X : x \sim_{lex}^\alpha x^*\} \end{aligned} \quad (9)$$

where the second equality results from the fact that  $\succsim_{lex}^\alpha$  is complete and that we cannot have  $x \succ_{lex}^\alpha x^*$  by definition of  $x^*$ .

Using the definition of  $\alpha$ -*leximax* relation, the set  $A(\alpha)$  can also be written as follows:

$$A(\alpha) = \{x \in X : \forall k \leq q, \hat{C}^k(x) - \hat{C}^k(x_k^*) \leq \alpha\} \quad (10)$$

Any solution of  $A(\alpha)$  performs well with regard to the disutility vector since  $A(\alpha)$  is the set of solutions whose the  $k^{th}$  largest cost is close to the minimum for all  $k \leq q$ . If we consider this last condition as a robustness

property, then we can consider  $A(\alpha)$  as a set of robust solutions that we will call set of *lexicographic  $\alpha$ -robust solutions*.

It is obvious that for small values of  $\alpha$ , this set can be empty. The minimum value of  $\alpha$  that guarantees the existence of lexicographic  $\alpha$ -robust solutions is:

$$\alpha_{\min} = \min_{x \in X} \max_{1 \leq k \leq q} (\hat{C}^k(x) - \hat{C}^k(x_k^*)) \quad (11)$$

Moreover, the set of lexicographic  $\alpha$ -robust solutions is stable with regard to parameter  $\alpha$ , that is if  $\alpha' \leq \alpha$  then  $A(\alpha') \subseteq A(\alpha)$ .

We present, in appendix I, a simple algorithm named  $\alpha$ -LEXROB, for the determination of lexicographic  $\alpha$ -robust solutions in the case of a finite set of solutions. Algorithm  $\alpha$ -LEXROB is based on an iterative procedure which determines, in each iteration  $k \in \{1, \dots, q\}$ , the subset:

$$A^k(\alpha) = \{x \in A^{k-1}(\alpha) : \hat{C}^k(x) - \hat{C}^k(x_k^*) \leq \alpha\} \quad (12)$$

The algorithm requires  $O(|X|q)$  elementary operations where  $|X|$  is the number of elements of  $X$  and  $q$  the number of scenarios.

### C. Example

Let us consider the vertex 1-median problem. We have  $X = V$  and  $|X| = n$  where  $V$  is the set of all nodes of the graph and  $n$  the number of nodes. For this problem, algorithm  $\alpha$ -LEXROB is polynomial ( $O(nq)$ ). We consider the complete graph of figure 2.

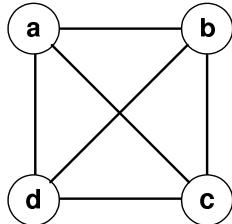


Fig. 2. A complete graph with four vertices

All edges are 1 unit long whereas node weights can take two possible values depending on the scenario (see table I). The cost of each vertex is computed according to equation (3). We look for lexicographic  $\alpha$ -robust solutions among vertices  $a, b, c$  and  $d$ .

According to the maximal cost criterion, the robust solution is  $b$  since it has the minimum cost in the worst case. It is clear that this solution is not so robust since it does not perform well under all scenarios. Indeed, its cost under scenarios  $S_1$  and  $S_2$  is rather high.

We give hereafter the lexicographic  $\alpha$ -robust solutions for different values of  $\alpha$ .

TABLE I  
Weights and costs of nodes under scenarios  $S_1$  and  $S_2$

vertex	weights		costs		dis. vect.	
	$S_1$	$S_2$	$S_1$	$S_2$	$\hat{C}^1$	$\hat{C}^2$
a	14	3	14	30	30	14
b	3	8	25	25	25	25
c	1	17	27	16	27	16
d	10	5	18	28	28	18

- $\alpha = 1 \Rightarrow A(1) = \emptyset$ .
- $\alpha = 2 \Rightarrow A(2) = \{c\}$ .
- $\alpha = 3 \Rightarrow A(3) = \{c\}$ .

### IV. LEXICOGRAPHIC $\alpha$ -ROBUST 1-MEDIAN PROBLEM ON A TREE

We consider the 1-median problem on a tree in the case of uncertainty on node weights. Kouvelis and Yu [14] consider the minmax cost and the minmax regret versions of this problem using scenario-based weights. They present an  $O(nq)$  algorithm to determine the minmax (regret) 1-median of the tree, where  $n$  is the number of nodes and  $q$  the number of scenarios.

Instead of finding a unique robust 1-median on the tree, we want to determine the lexicographic  $\alpha$ -robust set  $A(\alpha)$ , if it is not empty, in order to define *robust segments* of the tree. Since the feasible solution set is infinite, algorithm  $\alpha$ -LEXROB cannot be applied. After reminding the principle of Kouvelis and Yu's algorithm, we present a specific polynomial algorithm for the *lexicographic  $\alpha$ -robust 1-median* problem on a tree. We recall that the notation  $C$  and the word "cost" refer indifferently to cost or to regret.

#### A. Principle of Kouvelis and Yu's algorithm

Let  $T$  be a tree. The removal of any edge  $(v_i, v_j)$  of  $T$  partitions the tree into two connected components made up of node subsets  $V_i$  and  $V_j$ . For each point  $x$  on edge  $(v_i, v_j)$  of length  $c_{ij}$ , we denote by  $y$  the distance between node  $v_i$  and  $x$  ( $0 \leq y \leq c_{ij}$ ). The minimum maximal cost is defined as :

$$\min_{0 \leq y \leq c_{ij}} \max_{s \in S} C_{ij}^s(y) \quad (13)$$

where  $C_{ij}^s(y)$  is the cost under scenario  $s$  of the point of  $(v_i, v_j)$  at a distance  $y$  from node  $v_i$ . For the 1-median problem on a tree, the cost  $C_{ij}^s(y)$  can be written as:

$$C_{ij}^s(y) = \lambda_{ij}^s + \mu_{ij}^s y \quad (14)$$

with :

$$\mu_{ij}^s = \sum_{v_k \in V_i} w_k^s - \sum_{v_k \in V_j} w_k^s,$$

$$\lambda_{ij}^s = \sum_{v_k \in V_i} w_k^s d(v_i, v_k) + \sum_{v_k \in V_j} w_k^s (d(v_j, v_k) + c_{ij})$$

if  $C$  represents the cost and

$$\lambda_{ij}^s = \sum_{v_k \in V_i} w_k^s d(v_i, v_k) + \sum_{v_k \in V_j} w_k^s (d(v_j, v_k) + c_{ij}) - C^s(x^{*s})$$

if  $C$  represents the regret,  $C^s(x^{*s})$  being the minimum cost under scenario  $s$ .

In their approach, Kouvelis and Yu determine, for a given edge  $(v_i, v_j)$ , the solution  $y_{ij}^*$  which minimizes the maximal cost on the edge. They describe a procedure that computes  $y_{ij}^*$  by solving:

$$C_{ij}(y_{ij}^*) = \min_{0 \leq y \leq c_{ij}} \max_{s \in S} C_{ij}^s(y) \quad (15)$$

After applying this procedure to all edges of the tree, they use a linear time algorithm to find the minmax (regret) 1-median by determining, among all points  $y_{ij}^*$  found, one with a minimum maximal cost (regret).

### B. Determination of the robust segments of the tree

#### 1) Principle and notations:

We want to find the robust segments of a tree  $T$ , that is the set of lexicographic  $\alpha$ -robust solutions when  $X = T$ . We present here an algorithm which determines acceptable intervals that are reduced at each iteration and finally gives robust intervals.

For a given edge  $(v_i, v_j)$  of length  $c_{ij}$  and a point  $x \in (v_i, v_j)$ ,  $\hat{C}_{ij}^k(x)$  represents the  $k^{th}$  largest cost of  $x$  on interval  $[0, c_{ij}]$ ,  $1 \leq k \leq q$ . It is obvious that, unlike  $C_{ij}^s(\cdot)$ , costs  $\hat{C}_{ij}^k(\cdot)$  are not linear functions on  $[0, c_{ij}]$ .

We define the following subsets for  $k \in \{1, \dots, q\}$  and  $(v_i, v_j) \in E$ :

$$I_{ij}^k(\alpha) = \{y \in [0, c_{ij}] : \hat{C}_{ij}^k(y) - \hat{C}^k(x_k^*) \leq \alpha\} \quad (16)$$

where  $x_k^* = \arg \min_{x \in X} \hat{C}^k(x)$ . Subsets  $I_{ij}^k(\alpha)$  are called *acceptable intervals of order k*.

Let  $A_{ij}^k(\alpha)$  be the acceptable subsets defined as follows:

$$A_{ij}^1(\alpha) = I_{ij}^1(\alpha) \text{ and}$$

$$A_{ij}^k(\alpha) = A_{ij}^{k-1}(\alpha) \cap I_{ij}^k(\alpha) \text{ for } k \geq 2 \quad (17)$$

Then, the acceptable subset  $A^k(\alpha)$ ,  $k \geq 1$ , defined in equation (12) can be written as :

$$A^k(\alpha) = \bigcup_{(v_i, v_j) \in E} A_{ij}^k(\alpha) \quad (18)$$

Therefore, in order to determine the set of lexicographic  $\alpha$ -robust solutions, we use the following algorithm.

### Algorithm $\alpha$ -LEXROB(1MT)

**Begin**

$$A^0(\alpha) \leftarrow X;$$

$$A_{ij}^0(\alpha) \leftarrow [0, c_{ij}] \text{ for all } (v_i, v_j) \in E;$$

$$k \leftarrow 1;$$

**while** ( $k \leq q$  and  $A^{k-1}(\alpha) \neq \emptyset$ ) **do**

    Compute  $x_k^*$ ;

**for** all  $(v_i, v_j) \in E$  **do**

        Determine  $I_{ij}^k(\alpha)$ ;

        Determine  $A_{ij}^k(\alpha) \leftarrow A_{ij}^{k-1}(\alpha) \cap I_{ij}^k(\alpha)$ ;

        Determine  $A^k(\alpha) \leftarrow \bigcup_{(v_i, v_j) \in E} A_{ij}^k(\alpha)$ ;

$k \leftarrow k + 1$ ;

**End.**

If for a given  $k \leq q$ ,  $A^k(\alpha) = \emptyset$ , then it is obvious that  $A(\alpha) = \emptyset$ .

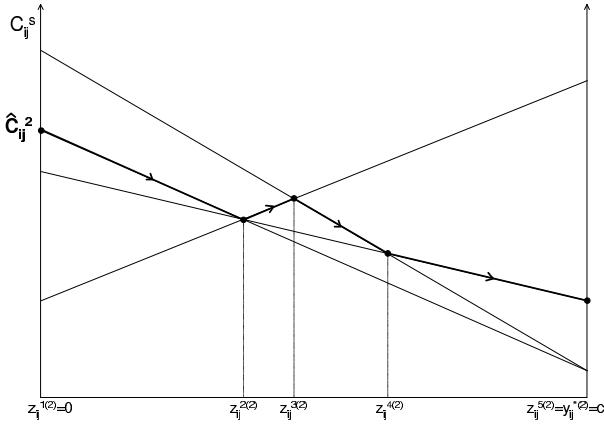
In the following, we detail the procedures required by the algorithm.

#### 2) Determination of $x_k^*$ :

We begin by determining, for each edge  $(v_i, v_j)$ , the point  $y_{ij}^{*(k)}$  which minimizes the cost  $\hat{C}_{ij}^k$  on  $(v_i, v_j)$ . The point  $x_k^*$  corresponds to the point  $y_{ij}^{*(k)}$  with the minimum cost  $\hat{C}^k$ . For  $k > 1$ , functions  $\hat{C}_{ij}^k(\cdot)$  are piecewise linear but not convex unlike functions  $\hat{C}_{ij}^1(\cdot)$ . As a result, it is not possible to use the Kouvelis and Yu's procedure since it is based on the convexity of  $\hat{C}_{ij}^1(\cdot)$ . We propose another approach consisting in determining, for each interval  $[0, c_{ij}]$ , all points corresponding to a breakpoint of function  $\hat{C}_{ij}^k(\cdot)$ . Indeed, if it is different from 0 and  $c_{ij}$ ,  $y_{ij}^{*(k)}$  is bound to be one of the points where the function slope changes (see figure 3). We call these points  $z_{ij}^{h(k)}$ ,  $1 \leq h \leq h_{ij}^k$ , where  $h_{ij}^k$  is the number of breakpoints of function  $\hat{C}_{ij}^k$  on  $[0, c_{ij}]$ . Points  $z_{ij}^{1(k)}$  and  $z_{ij}^{h_{ij}^k(k)}$  correspond respectively to 0 and  $c_{ij}$ .

We present, in appendix II, a procedure ( $Find(y_{ij}^{*(k)})$ ) giving for an edge  $(v_i, v_j)$  all points  $z_{ij}^{h(k)}$ ,  $1 \leq h \leq h_{ij}^k$ , as well as the point  $y_{ij}^{*(k)}$  whose cost  $\hat{C}_{ij}^k$  is minimum.

The main idea of procedure  $Find(y_{ij}^{*(k)})$  is to determine the scenarios  $s_{ij}^{h(k)}$ ,  $h \in \{1, \dots, h_{ij}^k\}$ , which give the  $k^{th}$  largest cost function  $\hat{C}_{ij}^k$  on the interval  $[0, c_{ij}]$ . At each iteration, the procedure determines, at the current breakpoint  $z_{ij}^{h(k)}$ , the line that must be chosen from those (two or more) available (see figure 3) as well as the adjacent breakpoint.

Fig. 3. Illustration of procedure  $Find(y_{ij}^{*(k)})$  when  $k = 2$  and  $q = 4$ 

**Lemma 1:** The complexity of procedure  $Find(y_{ij}^{*(k)})$  is  $O(q^{7/3} \log q)$

*Proof:* see appendix III.

**Lemma 2:** Finding  $x_k^*$  requires  $O(nq^{7/3} \log q)$  elementary operations.

*Proof:* Lemma 2 follows directly from lemma 1 and from the following relation:

$$x_k^* = \arg \min_{x \in X} \hat{C}^k(x) = \arg \min_{(v_i, v_j) \in E} \hat{C}_{ij}^k(y_{ij}^{*(k)}) \quad (19)$$

We remind that a tree has  $n - 1$  edges.  $\square$

### 3) Determination of acceptable intervals $I_{ij}^k(\alpha)$ :

Unlike acceptable intervals of order 1, subsets  $I_{ij}^k(\alpha)$ ,  $2 \leq k \leq q$ , are not necessarily connected because of the non convexity of functions  $\hat{C}_{ij}^k(\cdot)$ . Nevertheless, for the sake of convenience, we will continue to call them *acceptable intervals*. Let us notice that the nature of these intervals depends on parameter  $\alpha$ . A subset  $I_{ij}^k(\alpha)$  can be represented by a unique interval for a given value of  $\alpha$ , and change into the union of several intervals for a different value of this parameter.

If it is not empty, the acceptable interval  $I_{ij}^k(\alpha)$  can be represented by the union of  $p_{ij}^k$  elementary intervals as follows:

$$I_{ij}^k(\alpha) = [y_{ij}^{1(k)}, y_{ij}^{2(k)}] \cup \dots \cup [y_{ij}^{2p_{ij}^k-1(k)}, y_{ij}^{2p_{ij}^k(k)}] \quad (20)$$

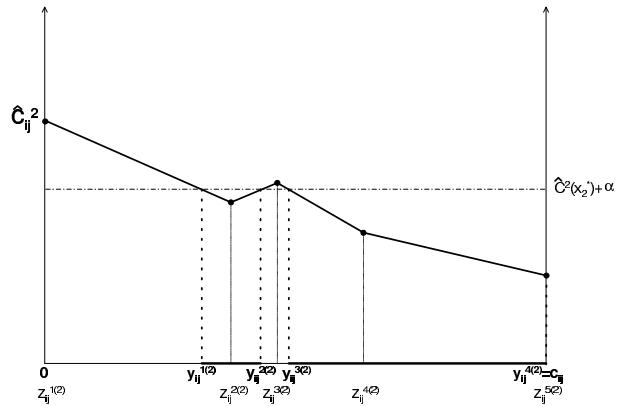
with  $0 \leq y_{ij}^{1(k)} \leq \dots \leq y_{ij}^{2p_{ij}^k(k)} \leq c_{ij}$  and  $p_{ij}^k \in \mathbb{N}^*$ .

Let us denote by  $E_{ij}^k$  the set of points corresponding to the bounds of  $I_{ij}^k(\alpha)$ :

$$E_{ij}^k = \{y_{ij}^{1(k)}, y_{ij}^{2(k)}, \dots, y_{ij}^{2p_{ij}^k-1(k)}, y_{ij}^{2p_{ij}^k(k)}\} \quad (21)$$

Remark that for  $k = 1$ ,  $E_{ij}^1 = \{y_{ij}^1, y_{ij}^2\}$  and  $p_{ij}^1 = 1$ .

Remind that  $z_{ij}^{h(k)}$ ,  $1 \leq h \leq h_{ij}^k$ , are the breakpoints of  $\hat{C}_{ij}^k$  on  $[0, c_{ij}]$  determined by procedure  $Find(y_{ij}^{*(k)})$ . It is obvious that for a given  $h \in \{1, \dots, h_{ij}^k - 1\}$ , if we have  $z_{ij}^{h(k)} \notin I_{ij}^k(\alpha)$  and  $z_{ij}^{h+1(k)} \notin I_{ij}^k(\alpha)$  (that is  $\hat{C}_{ij}^k(z_{ij}^{h(k)}) > \hat{C}_{ij}^k(x_k^*) + \alpha$  and  $\hat{C}_{ij}^k(z_{ij}^{h+1(k)}) > \hat{C}_{ij}^k(x_k^*) + \alpha$ ), then  $[z_{ij}^{h(k)}, z_{ij}^{h+1(k)}] \cap I_{ij}^k(\alpha) = \emptyset$ . Similarly, if  $z_{ij}^{h(k)} \in I_{ij}^k(\alpha)$  and  $z_{ij}^{h+1(k)} \in I_{ij}^k(\alpha)$ , then  $[z_{ij}^{h(k)}, z_{ij}^{h+1(k)}] \subset I_{ij}^k(\alpha)$  (see figure 4). On the other hand, if one of them belongs to  $I_{ij}^k(\alpha)$  and not the other, then only a part of the interval  $[z_{ij}^{h(k)}, z_{ij}^{h+1(k)}]$  is included in  $I_{ij}^k(\alpha)$ . Therefore, we just have to enumerate the points  $z_{ij}^{h(k)}$  in a non-decreasing order for  $h$  varying from 1 to  $h_{ij}^k$  in order to determine the parts of intervals  $[z_{ij}^{h(k)}, z_{ij}^{h+1(k)}]$  which belong to  $I_{ij}^k(\alpha)$  and afterwards deduce the set  $E_{ij}^k$ .

Fig. 4. Illustration of procedure  $Find(I_{ij}^k)$  when  $k = 2$ 

A detailed procedure,  $Find(I_{ij}^k)$ , is presented in appendix II. Since we go over all points  $z_{ij}^{h(k)}$ ,  $1 \leq h \leq h_{ij}^k$ , the complexity of procedure  $Find(I_{ij}^k)$  is  $O(h_{ij}^k)$ , so  $O(q^{4/3})$  (see proof of lemma 1 in appendix III).

**Lemma 3:** Given  $x_k^*$ , finding the set  $I^k(\alpha)$  which is the union of  $I_{ij}^k(\alpha)$  over  $E$  requires  $O(nq^{4/3})$  elementary operations.

### 4) Determination of the acceptable subsets $A_{ij}^k(\alpha)$ :

We have  $A_{ij}^1(\alpha) = I_{ij}^1(\alpha)$ . For  $k \geq 2$ ,

$$A_{ij}^k(\alpha) = A_{ij}^{k-1}(\alpha) \cap I_{ij}^k(\alpha) \quad (22)$$

$A_{ij}^k(\alpha)$  is then the union of  $r_{ij}^k$  subintervals of  $[0, c_{ij}]$ :

$$A_{ij}^k(\alpha) = [a_{ij}^{1(k)}, a_{ij}^{2(k)}] \cup \dots \cup [a_{ij}^{2r_{ij}^k-1(k)}, a_{ij}^{2r_{ij}^k(k)}] \quad (23)$$

with  $0 \leq a_{ij}^{1(k)} \leq \dots \leq a_{ij}^{2r_{ij}^k(k)} \leq c_{ij}$  and  $r_{ij}^k \in \mathbb{N}^*$ .

Let us denote by  $B_{ij}^k$  the set of points corresponding to the bounds of  $A_{ij}^k(\alpha)$ :

$$B_{ij}^k = \{a_{ij}^{1(k)}, a_{ij}^{2(k)}, \dots, a_{ij}^{2r_{ij}^k-1(k)}, a_{ij}^{2r_{ij}^k(k)}\} \quad (24)$$

For  $k = 1$ ,  $B_{ij}^1 = \{a_{ij}^1, a_{ij}^2\} = \{y_{ij}^1, y_{ij}^2\}$  and  $r_{ij}^1 = 1$ .

Given  $B_{ij}^{k-1}$  (bounds of  $A_{ij}^{k-1}(\alpha)$ ) and  $E_{ij}^k$  (bounds of  $I_{ij}^k(\alpha)$ ), then  $B_{ij}^k \subset (B_{ij}^{k-1} \cup E_{ij}^k)$ . Consequently, to find the bounds of  $A_{ij}^k(\alpha)$ , we have to look for them among those of  $A_{ij}^{k-1}(\alpha)$  and  $I_{ij}^k(\alpha)$ . The figure 5 shows how to find the set  $B_{ij}^k$  (see appendix II for the detailed procedure).

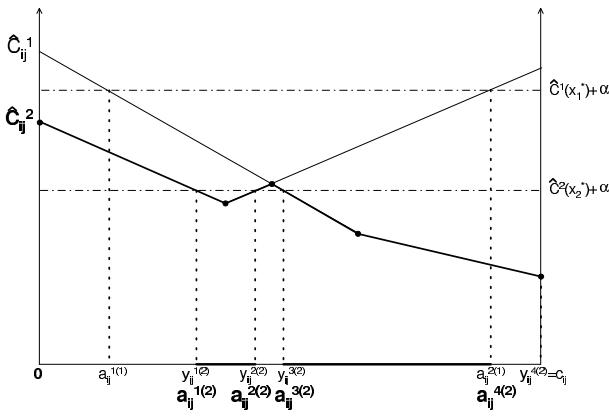


Fig. 5. Illustration of procedure  $\text{Find}(A_{ij}^k)$  when  $k = 2$

**Lemma 4:** The complexity of procedure  $\text{Find}(A_{ij}^k)$  is  $O(q^3)$ .

*Proof:* see appendix III.

5) Complexity of lexicographic  $\alpha$ -robust segments of the tree:

**Theorem 1:** Lexicographic  $\alpha$ -robust 1-median on a tree can be solved in  $O(nq^4)$  time.

*Proof:* Theorem 1 follows immediately from lemmas 2, 3 and 4. Indeed, the largest number of elementary operations needed is due to the determination of sets  $A^k(\alpha)$  for all  $k \leq q$ .  $\square$

### C. Example

Consider the tree  $T$  of figure 6 where values on edges represent lengths. Uncertainty on node weights is modelled by four scenarios as shown in table II ( $v_s^*$  is the median under scenario  $s$ ,  $s \in \{S_1, S_2, S_3, S_4\}$ ).

The minmax median  $x^*$  of the tree is the point of  $(v_1, v_4)$  at a distance 2.5 from node  $v_1$  and the minmax

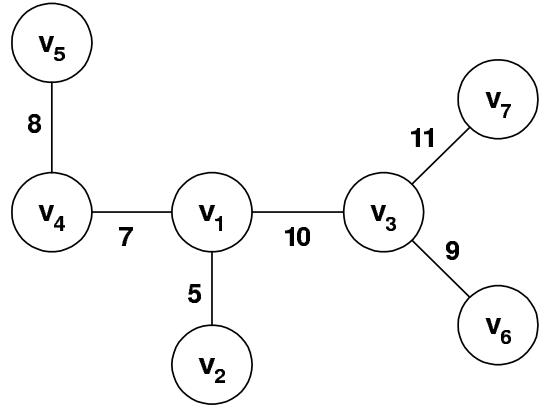


Fig. 6. Example

TABLE II  
Weight scenarios

weights	$S_1$	$S_2$	$S_3$	$S_4$
$v_1$	1	1	1	1
$v_2$	1	10	1	1
$v_3$	1	1	10	1
$v_4$	1	1	1	1
$v_5$	1	1	1	10
$v_6$	1	1	1	1
$v_7$	1	1	1	1
$v_s^*$	$v_1$	$v_2$	$v_3$	$v_5$

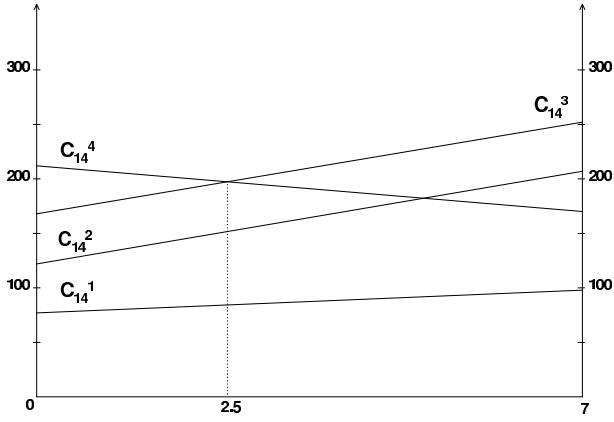
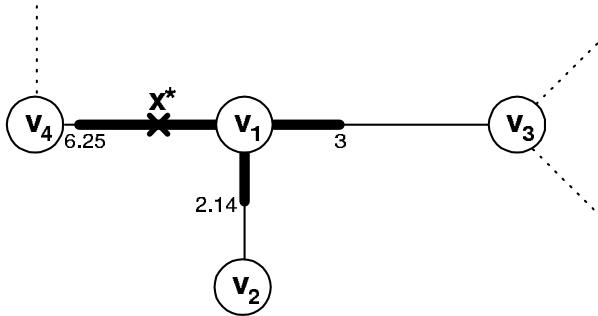
cost is 197. We present in figure 7 the cost functions on edge  $(v_1, v_4)$  ( $C_{14}^j$  denotes the cost function on interval  $[0, 7]$  under scenario  $j$ ). The points which minimize costs  $\hat{C}^k$ ,  $k = 1, \dots, 4$ , are  $x_1^* = x^*$ ,  $x_2^*$  the point of edge  $(v_1, v_3)$  at a distance 2.5 from node  $v_1$ ,  $x_3^* = v_3$  and  $x_4^* = v_1$ . If we choose a threshold  $\alpha = 45$ , the first iteration of algorithm  $\alpha$ -LEXROB(1MT) gives the set:

$$A^1(45) = \{x \in T : \hat{C}^1(x) - 197 \leq 45\} \quad (25)$$

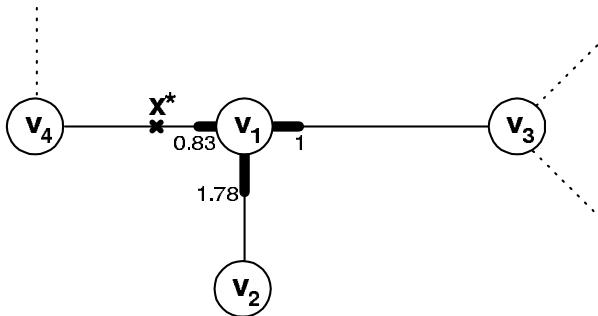
represented by bold segments in figure 8.

After four iterations, we get the set  $A(45)$  of lexicographic  $\alpha$ -robust solutions of the tree.  $A(45)$  is represented by the union of three segments  $(v_1, v'_2)$ ,  $(v_1, v'_3)$  and  $(v_1, v'_4)$  where  $v'_2$  is the point of edge  $(v_1, v_2)$  at a distance 1.78 from node  $v_1$ ,  $v'_3$  the point of edge  $(v_1, v_3)$  at a distance 1 from node  $v_1$  and  $v'_4$  the point of edge  $(v_1, v_4)$  at a distance 0.83 from node  $v_1$  (see figure 9). Remark that  $x^*$ , the minmax robust solution, is outside the lexicographic  $\alpha$ -robust set. Indeed, it performs well for the maximal cost function, but not well enough for  $\hat{C}^2$ ,  $\hat{C}^3$  and  $\hat{C}^4$ , compared with node  $v_1$  for example.

The minimum value of parameter  $\alpha$  which guarantees the existence of lexicographic  $\alpha$ -robust solutions

Fig. 7. Cost functions on edge  $(v_1, v_4)$ Fig. 8. Set  $A^1(45)$ 

is  $\alpha_{min} = 35$ , that is  $\alpha < 35 \Rightarrow A(\alpha) = \emptyset$  and  $\alpha \geq 35 \Rightarrow A(\alpha) \neq \emptyset$ .

Fig. 9. Minmax robust solution ( $x^*$ ) and lexicographic  $\alpha$ -robust solutions of the tree for  $\alpha = 45$ 

## V. CONCLUSION AND PERSPECTIVES

In this paper, we introduced a new robustness approach, called *lexicographic  $\alpha$ -robustness*, suitable for the scenario-based uncertainty. Compared with minmax criteria, this approach has three main advantages. First, it takes into account several measures, that is to say costs. Second, it offers some tolerance since it includes an indifference threshold  $\alpha$ . Finally, it keeps the importance of the largest cost. We presented a simple algorithm

for the determination of lexicographic  $\alpha$ -robust solutions when the set of solutions is finite. We also studied a special problem in the case of an infinite set of solutions, the 1-median location problem on a tree graph and we presented a polynomial algorithm for this problem.

It is obvious that lexicographic  $\alpha$ -robustness adds some more complexity to minmax versions of a problem, that is why it is reasonable to apply this approach only to problems which remain polynomially solvable under minmax criteria. In facility location context, the *minmax 1-center* problem on a tree ([3], [6]) and the *minmax 1-median* problem on a general network ([4]) are shown to remain polynomially solvable in the case of interval uncertainty on weights. We recall that the *1-center* problem is to locate a facility on a graph such that it minimizes the maximal distance between the facility and different nodes. We consider the application of lexicographic  $\alpha$ -robustness to these problems to be an avenue for future research.

## ACKNOWLEDGEMENTS

This research was partially funded by the cooperation agreement CNRS/CGRI-FNRS n° 18 227.

## APPENDIX I

### Algorithm $\alpha$ -LEXROB

**Input:**  $\hat{C}(x)$  for all  $x \in X$  and  $\alpha$ .

**Output:**  $A(\alpha)$ .

**Begin**

```

 $A^0 \leftarrow X;$ 
 $k \leftarrow 1;$ 
while ( $k \leq q$  and  $A^{k-1} \neq \emptyset$ ) do
     $x_k^* \leftarrow \arg \min_{x \in X} \hat{C}^k(x);$ 
     $A^k \leftarrow \emptyset;$ 
    for all  $x \in A^{k-1}$  do
        if  $\hat{C}^k(x) - \hat{C}^k(x_k^*) \leq \alpha$  then  $A^k \leftarrow A^k \cup \{x\};$ 
     $k \leftarrow k + 1;$ 
     $A(\alpha) \leftarrow A^{k-1};$ 
End.

```

The condition  $A^{k-1} \neq \emptyset$  is used for stopping the algorithm if  $A^{k-1}(\alpha)$  is empty, since if  $A^{k-1}(\alpha) = \emptyset$  then for all  $l \geq k$ ,  $A^l(\alpha) = \emptyset$ .

## APPENDIX II

*Procedure Find( $y_{ij}^{*(k)}$ )*

**Input:**  $\lambda_{ij}^s$  and  $\mu_{ij}^s$  for all  $s \in S$ ,

**Output:**  $y_{ij}^{*(k)}$ ,  $\hat{C}_{ij}^k(y_{ij}^{*(k)})$ ,  $h_{ij}^k$ ,  $z_{ij}^{h(k)}$  and  $s_{ij}^{h(k)}$  for  $h \in \{1, \dots, h_{ij}^k\}$ .

**Begin**

$z_{ij}^{1(k)} \leftarrow 0$ ;

$h \leftarrow 1$ ;

**repeat**

1) Compute costs  $\hat{C}_{ij}^1(z_{ij}^{h(k)})$  to  $\hat{C}_{ij}^k(z_{ij}^{h(k)})$ ;

2) Find set of scenarios  $S^k(z_{ij}^{h(k)})$  such that:

$S^k(z_{ij}^{h(k)}) = \{s \in S; \lambda_{ij}^s + \mu_{ij}^s z_{ij}^{h(k)} = \hat{C}_{ij}^k(z_{ij}^{h(k)})\}$ ;

3) Let  $d$  be the first index such that:

$\hat{C}_{ij}^{k-d}(z_{ij}^{h(k)}) \neq \hat{C}_{ij}^k(z_{ij}^{h(k)})$ ;

4) Let  $s_{ij}^{h(k)}$  be the scenario of  $S^k(z_{ij}^{h(k)})$  with the  $d^{\text{th}}$  largest slope (slope =  $\mu_{ij}^s$ );

5) Compute the intersection points of segment  $\{\lambda_{ij}^{s_{ij}^{h(k)}} + \mu_{ij}^{s_{ij}^{h(k)}} z, z \in [z_{ij}^{h(k)}, c_{ij}]\}$  with all lines corresponding to other scenarios;

6) The point  $z_{ij}^{h+1(k)}$  is the nearest intersection point from  $z_{ij}^{h(k)}$ :

$z_{ij}^{h+1(k)} \leftarrow \min\{c_{ij};$

$\min_{s \in S \setminus \{s_{ij}^{h(k)}\}} \{z^s = \frac{\lambda_{ij}^{s_{ij}^{h(k)}} - \lambda_{ij}^s}{\mu_{ij}^{s_{ij}^{h(k)}} - \mu_{ij}^s}; z^s \in [z_{ij}^{h(k)}, c_{ij}]\}\}$ ;

7)  $h \leftarrow h + 1$ ;

**until**  $z_{ij}^{h(k)} = c_{ij}$ ;

$h_{ij}^k \leftarrow h$ ;

$y_{ij}^{*(k)} \leftarrow z_{ij}^{t(k)}$  where  $t \leftarrow \arg \min_{1 \leq h \leq h_{ij}^k} \{\hat{C}_{ij}^k(z_{ij}^{h(k)})\}$ ;

**End.**

Step 1 computes the  $k$  largest costs of the current breakpoint  $z_{ij}^{h(k)}$ . Steps 2, 3 and 4 are used for determining the line that must be chosen from those (two or more) available in order to determine the scenario which gives  $\hat{C}^k$  (see figure 3). Steps 5 and 6 are used to determine the adjacent breakpoint  $z_{ij}^{h+1(k)}$ .

*Procedure Find( $I_{ij}^k$ )*

**Input:**  $\hat{C}^k(x_k^*)$ ,  $\alpha$ ,  $\hat{C}^k(y_{ij}^{*(k)})$ ,  $h_{ij}^k$ ,  $z_{ij}^{h(k)}$ ,  $s_{ij}^{h(k)}$  and  $\hat{C}_{ij}^h(z_{ij}^{h(k)})$  for  $h \in \{1, \dots, h_{ij}^k\}$ .

**Output:**  $E_{ij}^k$ ,  $p_{ij}^k$ .

**Begin**

**if**  $\hat{C}_{ij}^k(y_{ij}^{*(k)}) > \hat{C}^k(x_k^*) + \alpha$  **then**  $E_{ij}^k \leftarrow \emptyset$ ;

**else**

$E_{ij}^k \leftarrow \emptyset$ ;

$p \leftarrow 0$ ;

**if**  $\hat{C}_{ij}^h(z_{ij}^{1(k)}) \leq \hat{C}^k(x_k^*) + \alpha$  **then**

$E_{ij}^k \leftarrow E_{ij}^k \cup \{z_{ij}^{1(k)}\}$ ;

$p \leftarrow p + 1$ ;

**if**  $\hat{C}_{ij}^h(z_{ij}^{h_{ij}^{k(k)}}) \leq \hat{C}^k(x_k^*) + \alpha$  **then**

$E_{ij}^k \leftarrow E_{ij}^k \cup \{z_{ij}^{h_{ij}^{k(k)}}\}$ ;

$p \leftarrow p + 1$ ;

**for**  $h = 1$  to  $h_{ij}^k - 1$  **do**

**if**  $\hat{C}_{ij}^h(z_{ij}^{h(k)}) > \hat{C}^k(x_k^*) + \alpha$  **then**

**if**  $\hat{C}_{ij}^h(z_{ij}^{h+1(k)}) \leq \hat{C}^k(x_k^*) + \alpha$  **then**

$y \leftarrow \frac{\hat{C}^k(x_k^*) + \alpha - B_{ij}^{s_{ij}^{h(k)}}}{A_{ij}^{s_{ij}^{h(k)}}}$ ;

$E_{ij}^k \leftarrow E_{ij}^k \cup \{y\}$ ;

$p \leftarrow p + 1$ ;

**else**

**if**  $\hat{C}_{ij}^h(z_{ij}^{h+1(k)}) > \hat{C}^k(x_k^*) + \alpha$  **then**

$y \leftarrow \frac{\hat{C}^k(x_k^*) + \alpha - B_{ij}^{s_{ij}^{h(k)}}}{A_{ij}^{s_{ij}^{h(k)}}}$ ;

$E_{ij}^k \leftarrow E_{ij}^k \cup \{y\}$ ;

$p \leftarrow p + 1$ ;

$p_{ij}^k \leftarrow p/2$ ;

**End.**

*Procedure*  $\text{Find}(A_{ij}^k)$

**Input:**  $r_{ij}^{k-1}$ ,  $p_{ij}^k$ ,  $a_{ij}^{t(k-1)}$  for  $t \in \{1, \dots, 2r_{ij}^{k-1}\}$  and  $y_{ij}^{l(k)}$  for  $l \in \{1, \dots, 2p_{ij}^k\}$ .

**Output:**  $B_{ij}^k$ ,  $r_{ij}^k$ .

**Begin**

```

 $B_{ij}^k \leftarrow \emptyset;$ 
 $r \leftarrow 0;$ 
for  $t = 1$  to  $r_{ij}^{k-1}$  do
  for  $l = 1$  to  $p_{ij}^k$  do
    if  $y_{ij}^{2l-1(k)} \geq a_{ij}^{2t-1(k-1)}$  then
      if  $y_{ij}^{2l-1(k)} \leq a_{ij}^{2t(k-1)}$  then
         $B_{ij}^k \leftarrow B_{ij}^k \cup \{y_{ij}^{2l-1(k)}\};$ 
         $r \leftarrow r + 1;$ 
      if  $y_{ij}^{2l(k)} \leq a_{ij}^{2t(k-1)}$  then
         $B_{ij}^k \leftarrow B_{ij}^k \cup \{y_{ij}^{2l(k)}\};$ 
         $r \leftarrow r + 1;$ 
    else
       $B_{ij}^k \leftarrow B_{ij}^k \cup \{a_{ij}^{2t(k-1)}\};$ 
       $r \leftarrow r + 1;$ 
  else
    if  $y_{ij}^{2l(k)} \geq a_{ij}^{2t-1(k-1)}$  then
       $B_{ij}^k \leftarrow B_{ij}^k \cup \{a_{ij}^{2t-1(k)}\};$ 
       $r \leftarrow r + 1;$ 
    if  $y_{ij}^{2l(k)} \leq a_{ij}^{2t(k-1)}$  then
       $B_{ij}^k \leftarrow B_{ij}^k \cup \{y_{ij}^{2l(k)}\};$ 
       $r \leftarrow r + 1;$ 
    else
       $B_{ij}^k \leftarrow B_{ij}^k \cup \{a_{ij}^{2t(k-1)}\};$ 
       $r \leftarrow r + 1;$ 
   $r_{ij}^k \leftarrow r/2;$ 
End.

```

### APPENDIX III

**Proof of lemma 1:** For a given edge  $(v_i, v_j)$ , a given order  $k$  ( $k \in \{1, \dots, q\}$ ) and a given iteration  $h$  ( $h \in \{1, \dots, h_{ij}^k\}$ ), step 1 requires a sorting in a set of  $q$  elements  $(C_{ij}^s(z_{ij}^{h(k)}), s \in S)$ , therefore it can be solved in  $O(q \log q)$  time. Steps 2, 5 and 6 can be solved in  $O(q)$  time, whereas steps 3 and 4 have a complexity of  $O(k)$ . Consequently, for a given edge  $(v_i, v_j)$  and a given order  $k$ , procedure  $\text{Find}(y_{ij}^{*(k)})$  can be computed in  $O((h_{ij}^k - 1)q \log q)$  since  $k \leq q$  and  $h \leq h_{ij}^k$ .

Moreover, based on Dey's theorem ([1], [10]), the number of segments of  $\hat{C}_{ij}^k$  is  $O(q(q-k)^{1/3})$ . So we can write  $O(h_{ij}^k) = O(q(q-k)^{1/3}) = O(q^{4/3})$  for all  $k \leq q$ .

The point  $y_{ij}^{*(k)}$  is the one of edge  $(v_i, v_j)$  with the lowest cost  $\hat{C}_{ij}^k$  among all  $z_{ij}^{h(k)}$  found by the procedure  $\text{Find}(y_{ij}^{*(k)})$ . Thus, it can be found in  $O(q^{7/3} \log q)$  time.  $\square$

**Proof of lemma 4:** Given an edge  $(v_i, v_j)$  and an order  $k$ , solving procedure  $\text{Find}(A_{ij}^k)$  requires  $O(r_{ij}^{k-1} \cdot p_{ij}^k)$  elementary operations.  $p_{ij}^k$  is the number of subintervals of  $[0, c_{ij}]$  given by the intersection of a straight line ( $D = \hat{C}^k(x_k^*) + \alpha$ ) with at most  $q$  lines and possibly the lines  $y_{ij} = 0$  and  $y_{ij} = c_{ij}$ . Therefore,  $p_{ij}^k \leq \frac{q+2}{2}$ . On the other hand,  $B_{ij}^k \subset (E_{ij}^k \cup B_{ij}^{k-1})$  implies that:

$$r_{ij}^k \leq p_{ij}^k + r_{ij}^{k-1} \leq \frac{q+2}{2} + r_{ij}^{k-1} \leq \dots \leq (k-1)(\frac{q+2}{2}) + r_{ij}^1$$

As  $r_{ij}^1 = 1$  and  $k \leq q$ , we get  $r_{ij}^k \leq q(\frac{q+2}{2}) + 1$ , for all  $k \in \{1, \dots, q\}$ .

Consequently,  $r_{ij}^{k-1} \cdot p_{ij}^k \leq (q(\frac{q+2}{2}) + 1) \cdot \frac{q+2}{2}$ , so  $O(r_{ij}^{k-1} \cdot p_{ij}^k) = O(q^3)$ .  $\square$

### REFERENCES

- [1] AGARWAL P.K. and SHARIR M., "Arrangements and their applications," in *Handbook of Computational Geometry*, SACK J.-R. and URRUTIA J., Eds. Elsevier Sciences, 2000, pp. 49–120.
- [2] AVERBAKH I., "Complexity of robust single facility location problems on networks with uncertain edge length," *Discrete Applied Mathematics*, vol. 127, pp. 505–522, 2003.
- [3] AVERBAKH I. and BERMAN O., "Algorithms for the robust 1-center problem on a tree," *European Journal of Operational Research*, vol. 123, pp. 292–302, 2000.
- [4] ——, "Minmax regret median location on a network under uncertainty," *INFORMS Journal on Computing*, vol. 12, no. 2, pp. 104–110, 2000.
- [5] ——, "An improved algorithm for the minmax regret median problem on a tree," *Networks*, vol. 41, no. 2, pp. 97–103, 2003.
- [6] BURKARD R.E. and DOLLANI H., "Robust location problems with pos/neg weights on a tree," *Networks*, vol. 38, pp. 102–113, 2001.
- [7] CHEN B. and LIN C.S., "Min-max regret robust 1-median location on a tree," *Networks*, vol. 31, pp. 93–103, 1998.
- [8] DASKIN M.S., *Network and Discrete Location: Models, Algorithms and Applications*. Wiley, 1995.
- [9] DASKIN M.S., HESSE S.M., and REVELLE C.S., " $\alpha$ -reliable p-minimax regret: a new model for strategic facility location modeling," *Location Science*, vol. 5, no. 4, pp. 227–246, 1997.
- [10] DEY T.K., "Improved bounds on planar k-sets and related problems," *Discrete Computational Geometry*, vol. 19, no. 3, pp. 373–382, 1998.
- [11] GRABISCH M. and PERNY P., "Agrégation multicritère," in *Logique Floue, principes, aide à la décision*, BOUCHON-MEUNIER B. and MARSALA C., Eds. Hermès-Lavoisier, 2003, pp. 81–120.
- [12] HAKIMI S.L., "Optimum locations of switching centers and the absolute centers and medians of a graph," *Operations Research*, vol. 12, pp. 450–459, 1964.

- [13] KOUVELIS P., KURAWARWALA A.A., and GUTIRREZ G.J., "Algorithms for robust single and multiple period layout planning for manufacturing systems," *European Journal of Operational Research*, vol. 63, no. 2, pp. 287–303, 1992.
- [14] KOUVELIS P. and YU G., *Robust Discrete Optimization and its Applications*, ser. Non Convex Optimization and Its Applications. Kluwer Academic Publishers, 1997.
- [15] MOULIN H., "Social welfare orderings," in *Axioms of cooperative decision making*. Cambridge University Press, 1988, pp. 30–60.
- [16] OGRYCZAK W., "On the lexicographic minimax approach to location problems," *European Journal of Operational Research*, vol. 100, pp. 566–585, 1997.
- [17] PIRLOT M. and VINCKE P., *Semiorders: Properties, Representations, Applications*. Kluwer Academic Publishers, 1997.
- [18] ROSENHEAD J., ELTON M., and GUPTA S.K., "Robustness and optimality criteria for strategic decisions," *Operational Research Quarterly*, vol. 23, no. 4, pp. 413–423, 1972.
- [19] ROY B., "A missing link in OR-DA : robustness analysis," *Foundations of computing and decision sciences*, vol. 23, no. 3, pp. 141–160, 1998.
- [20] ROY B. and VINCKE Ph., "Relational systems of preference with one or more pseudo-criteria: some new concepts and results," *Management Science*, vol. 30, no. 11, pp. 1323–1335, 1984.
- [21] SNYDER L.V. and DASKIN M.S., "Stochastic p-robust location problems," Lehigh University, Dept. of ISE, Technical report 04T-014, July 2004.
- [22] VAIKAKTARAKIS G.L. and KOUVELIS P., "Incorporation dynamic aspects and uncertainty in 1-median location problems," *Naval Research Logistics*, vol. 46, pp. 147–168, 1999.
- [23] VINCKE P., "Robust solutions and methods in decision-aid," *Journal of Multi-criteria Decision Analysis*, vol. 8, pp. 181–187, 1999.

# Models and Software for Improving the Profitability of Pharmaceutical Research

Jiun-Yu Yu\* and John Gittins†

Department of Statistics, University of Oxford

1 South Parks Road, Oxford, UK

Email: \*yu@stats.ox.ac.uk , †gittins@stats.ox.ac.uk

**Abstract**—The pharmaceutical industry is highly competitive, and the discovery and development of new drugs is extremely expensive and time consuming. This paper is a contribution to the task of improving the effectiveness of pre-clinical research. Our model investigates for any given project the number of lead series which should if necessary be optimised in the search for a development compound which is sufficiently promising to proceed to clinical trials. The numbers of scientists which should be allocated to each research stage are also investigated. Two widely-applied profitability criteria are considered. Computer software designed to implement the optimisation calculations is developed and shown to produce reasonable results.

**Keywords**—optimisation, pharmaceutical research, resource allocation.

## I. INTRODUCTION

PHARMACEUTICAL companies require great patience and enormous capital. It typically takes 12 to 15 years to successfully complete the research and development (R&D) process of a new drug, with probability of success less than 20%, while the combined cost of R&D and market introduction for a significant product today exceeds £ 700 million. (Chen, [7]) As a result, bringing research projects to an early successful conclusion gives an important competitive advantage.

There is a substantial literature, some of it specific to the pharmaceutical industry, on criteria for project selection and resource allocation in R&D. These range from simple check-lists to sophisticated pharmacoeconomic analysis. The journals *R&D Management* and *Pharmacoeconomics* are good general sources, see for example the review papers by Miller [18] and Poh *et al.* [20]. The earlier literature is reviewed by Bergman and Gittins [1].

The model discussed in this paper is a stochastic

economic model. There are three important themes with a bearing on models of this type, as follows.

- The methodology of decision analysis. This has been around since the 1960s. Key ingredients are personal probabilities, utility functions, sequential decisions expressed as a decision tree, and solution by a dynamic programming algorithm. McNamee and Celona have written a useful handbook [16], and Lindley [14] gives a good introduction to the main ideas.
- Real options. Black and Scholes [2] provided a methodology for valuing financial options. Others, notably Dixit and Pindyck [9], have pointed out that a similar analysis is possible for options to invest in specific projects.
- Pharmacoeconomics. This is the science of relating the costs and benefits, both to individuals and to society, of therapeutic regimes, including drugs. Analysis along these lines is becoming routine in the planning of clinical trials. The journal *Pharmacoeconomics* started in 1983.

The insight from financial options theory that the ability to postpone, and possibly eventually not take up, an investment opportunity can strongly influence its value has been important, see for example Burman and Senn [6], Chen [7], and Perlitz *et al.* [19]. However, the match between financial and real options is far from exact, and current economic approaches to pharmaceutical project planning tend to owe more to decision analysis than to real options theory. The papers by Stonebraker [22], Ding and Eliashberg [8], and Loch and Bode-Greuel [15] are good examples.

All these papers focus at least as much on development as on pre-clinical research, and so far as the authors are aware we have modelled the pre-clinical stages of research, in discussion with people working in the industry, in a much more detailed fashion than is to be found elsewhere. This is borne

out by the comment in [18] that there is currently very little pharmaco-economic planning in the early stages of research. Islei *et al.* [13] have written an important paper describing a planning system for those early stages. However it does not include an economic model. Further refinements of our model are in preparation.

## II. AIMS

Until the mid 1990s most pharmaceutical research projects proceeded in the following sequence. Bioscientists work out an hypothesis for the way in which a chemical intervention in the body's processes might achieve the desired result. Then bioscientists and chemists devise tests using animal tissue or live animals in order to screen compounds for relevant activity. Afterwards, chemists synthesise compounds designed in the hope of finding relevant activity. These are then subject to one or more screening tests. Compounds which were not synthesised for this specific purpose may also be screened. In recent years the initial screen has almost always been a high throughput robotic screen on tens of thousands of library compounds.

When a compound with a sufficiently high level of activity has been found, other compounds with similar chemical structures are synthesised and tested. The original active compound is termed a "lead compound". The subsequent testing of similar compounds is known as "optimising the lead". From those results which achieve promising results on the screening tests, which include tests for toxicity, a small number are selected, usually one at a time, for tests in man. The clinical trials are often referred to as the "development" phase of a project, and consequently a compound that goes on to clinical trials is called a "development compound". At most 20% of development compounds emerge from clinical trials as marketable drugs, so typically more compounds are screened while a compound is undergoing clinical trials, in order that one or more "backup compounds" may in turn be selected from them for development. In summary, the research process, excluding clinical trials, may be divided into four stages:

- 1) Before screening
- 2) Finding the first lead series
- 3) Optimising the first lead series to find the first development compound
- 4) Looking for a backup compound

Further information on this general setting is given by Bergman and Gittins [1], Boschi [3], Chen [7], Gittins [10], [12], and Spilker [21].

The probability of successful completion of each stage in a research project is denoted as  $p_i$  ( $1 - p_i = q_i$ ), where  $i = 1, 2, 3, 4$ . Based on these probabilities, the chances of successful completion of stages 1, 2, and 3, are  $p_1$ ,  $p_1 p_2$ , and  $p_1 p_2 p_3$ , respectively. In the fourth stage, we look for a number of backup development compounds once the first development compound has been discovered. We assume that each backup compound can be found with no possibility of failing. Thus,  $p_4$  is set at 1.0 in the model.

Gittins [11], [12] sets up a stochastic model based on these typical phases to investigate the relationships between the profitability and the number of scientists allocated to the different stages of a research project. An important general conclusion is that larger project teams than typical current team-sizes would be more profitable in some cases.

Note that the aforementioned research stages are not guaranteed to succeed; each stage has a probability of failure. To maximise the profitability and increase the probability of success of a research project, usually insurance or buffer lead series are identified, in case the first attempt at stage 3 fails. However, so far there has been no attempt to model this feature. One of the objectives of this paper is to study the relationships between profitability, the number of lead series to be identified, and the numbers of scientists allocated to each stage of a research project.

In this model, we have to decide the value of  $m$ , the maximum number of lead series from which we will try to find the first development compound. In cases where two or more lead series are required, if the first lead series is found at the end of stage 2, two tasks will proceed simultaneously: some scientists repeat stage 2 while others work on stage 3. This dual-task implementation will be repeated until the first development compound is found or  $m$  lead series are identified. The expected reward and the expected cost of a research project both increase as  $m$  increases. The value of  $m$  should be chosen so as to maximise the profitability of the research project.

In practice, to increase the profitability and the probability of success of a research project, usually two (or more) development compounds are found from different lead series. If we aim to find two development compounds from different lead series, we have to decide the maximum number of attempts we are prepared to make to find a development compound from a

second lead series. This maximum number of attempts is denoted as  $l$ , and the determination of its value is discussed in this paper.

Profitability Index and Internal Rate of Return are the two decision criteria which we shall employ to evaluate research projects. These are standard criteria for the evaluation of capital expenditure, see for example Brealey and Myers [5]. Computer software has been developed to find the optimal solutions, including the optimal numbers of scientists allocated to each stage, the optimal number of backup compounds found in stage 4, and the optimal values of  $m$  and  $l$ .

### III. MODELLING A RESEARCH PROJECT

#### A. Measure of Effectiveness

The time needed to complete a particular stage of a research project is not necessarily inversely proportional to the number of scientists involved in this stage. As the team-size gets larger, the difficulties of communication and coordination will eventually decrease the efficiency of the team. We define

- $e(u)$  : the effectiveness of a team of  $u$  scientists.
- $e(u)/u$  : relative efficiency of a team of  $u$  scientists.

Note that  $e(u)/u \leq 1$ .

The effectiveness is taken to be of the form

$$e(u) = \frac{bu^2}{1 + au^r},$$

where  $a > 0$ ,  $b > 0$ , and  $1 < r < 2$ . Next we define the most efficient team size

- $u_{opt}$  : most efficient team size. This is the value of  $u$  where  $e(u)/u$  is maximised, i.e.,

$$\frac{e(u_{opt})}{u_{opt}} = 1.$$

According to the definition of Gittins [10], the project team is of the most efficient size when the rate of progress per scientist is greatest. Suppose  $u_i$  scientists are allocated in stage  $i$ , ( $i = 1, 2, 3, 4$ ),  $X_i$  is the effort needed to complete stage  $i$  of the research project, and is measured in *scientist-years*. Thus,  $t_i$ , the time needed to complete stage  $i$  with  $u_i$  scientists, is given by the equation  $t_i = \frac{X_i}{e(u_i)}$ .

*DBEFF* is the relative efficiency of a team in which the team-size is double  $u_{opt}$ , i.e.,  $DBEFF = \frac{e(2u_{opt})}{2u_{opt}}$ . A numerical algorithm is called to calculate the values of  $a$ ,  $b$ , and  $r$  for given values of  $u_{opt}$  and *DBEFF*. Figure 1 shows some examples of functions defined in this way.

#### B. Discount Rate and Obsolescence Effect

Since the value of a sum of future money is lower than that of the same sum of money available immediately, the discount factor,  $\gamma$ , will be applied to calculate the expected present value of future expenditure.

- $\gamma$  : discount rate in real terms

Thus, the present value of  $\$e^{\gamma t}$  after  $t$  years is  $\$1$ .

In most cases the cash benefits resulting from the sales of a drug will themselves become lower if its launch time is delayed, due to the general tendency for better drugs to be available from competitors as time goes by. This effect is defined as *obsolescence*, which means that the exponential discount rate for future rewards of the research project,  $\gamma_1$ , is higher than  $\gamma$ , the discount rate applied to future costs. Let  $\xi\Delta t$  denote the probability of a competitor sending a development compound for clinical trials in a short interval of time  $\Delta t$ , and  $f$  be the expected fraction by which a competitor's earlier development compound reduces the value of a development compound. Gittins defines the exponential rate for discounting future rewards  $\gamma_1$  as the sum of the rate for discounting future costs and the rate due to obsolescence, namely,  $\gamma + f\xi$ .

In addition, when competitors learn of progress in our research project, there may well be an increase both in the competitive research, hence an increase in  $\xi$ , and in the similarities between the compounds screened by competitors and those screened in the project, resulting in an increase in  $f$ . To model this effect of additional *self-induced obsolescence*, a second discount rate,  $\gamma_2$ , is introduced. Let  $T$  be the period from the start of our screening process to the time when competitors become aware of our project, provided no compound has been submitted for clinical trials so far. Future rewards are discounted at the rate  $\gamma_1$  up to the time  $T$ , or until the first development compound is sent for clinical trial, if this is earlier, and from then on at the rate  $\gamma_2$ . As a result,

- $\gamma_1$  : discount rate for the future income of a development compound in real terms, *excluding* any self-induced obsolescence. Note that  $\gamma_1 = \gamma + f\xi > \gamma$ .
- $\gamma_2$  : as  $\gamma_1$ , but *includes* self-induced obsolescence. From the discussion above,  $\gamma_2 > \gamma_1$ .

#### C. Search for the First Development Compound

In the simplest model, only one lead series is identified if stage 2 is successfully completed, and this lead series is optimised directly to find the first

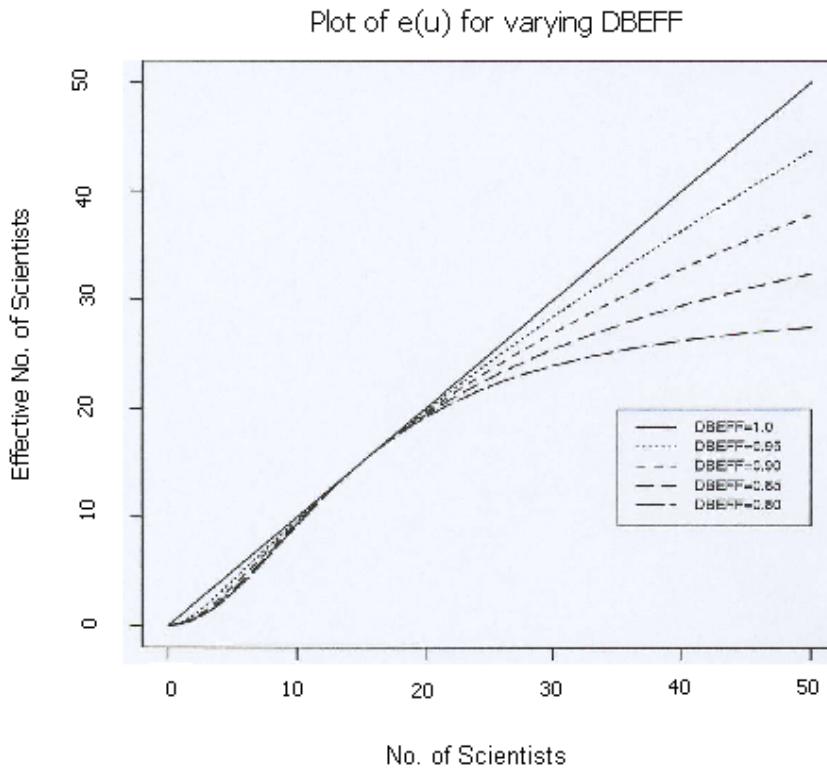


Fig. 1. Team-size effectiveness function for various  $DBEFF$  values.

development compound. If this attempt fails, then the research project is terminated without any financial return. In practice, to increase the probability of success and the future profitability of the project, it is normal to identify more than one lead series from which to look for development compounds. Let us first consider the case in which we are not only interested in finding one lead series from which we can find development compounds, but also prepared to look at more than one series in our search for the first development compound. This leads to two important questions: how many lead series should we be prepared to investigate, and how many scientists should be allocated to find these lead series?

When more than one lead series are required, scientists may be divided into two groups; the majority of them work on stage 3, the optimisation process looking for a development compound, while the minority repeat stage 2 to find additional "insurance" or "buffer" lead series. These two tasks proceed simultaneously. We shall make two assumptions about this repeated stage 2, which we shall call stage 2b. First, the effort  $X_{2b}$  needed to find an additional lead series is less than that required to find the first one, which we shall write

$X_{2a}$ . Second, the probability of success in finding an additional lead series is 1.0. Thus we will definitely find more "insurance" or "buffer" lead series once the first one has been found, provided that sufficient effort has been put in. The second assumption is obviously an oversimplification, but it is a reasonable approximation to reality.

We introduce a new decision variable,  $m$ , defined

- $m$  : maximum number of lead series from which we will try to find the first development compound.

Figure 2 is an event tree showing the possible consequences of different values of  $m$ .

Setting  $m = 1$  is the simplest policy, in which only one lead series would be optimised once it has been found, and no further investigation would be carried out to find additional lead series. If our attempt to find a development compound in this lead series succeeds, we then invest our time and effort in stage 4, in which we identify  $k$  backup compounds. If this attempt is a failure, the whole research project is terminated and there is no financial return.

When  $m = 2$ , we search for at most two lead series;

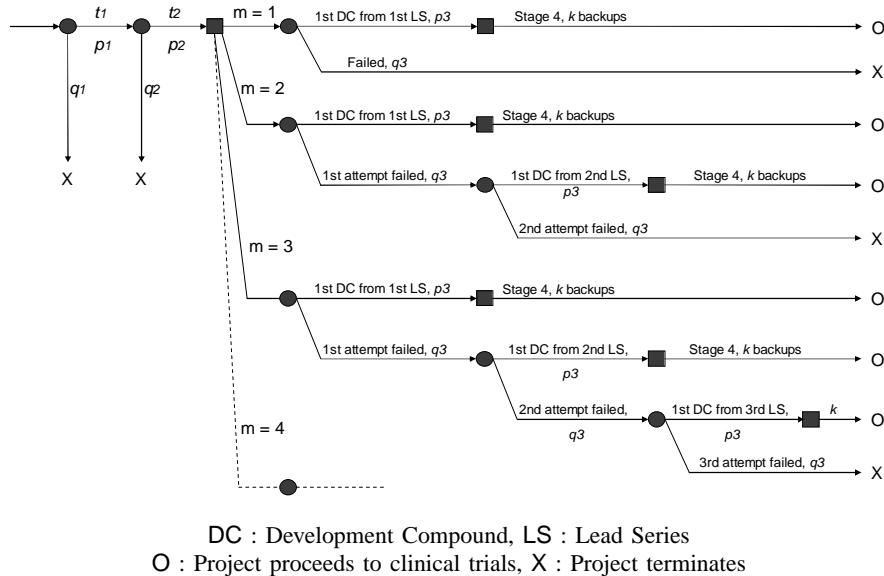


Fig. 2. Event tree for various  $m$ , the maximum number of attempts to find the first development compound.

consequently we have at most two chances to carry out stage 3 to find the first development compound. After the first lead series is obtained, the lead optimisation process to find a development compound and repetition of stage 2 occur at the same time. The number of scientists allocated to the repeated stage 2, which we call stage 2b, is expressed as  $u_{2b}$ . If the first lead series is successfully optimised and thus the first development compound is found, our investigation on the second lead series will be suspended. If on the other hand the first lead series does not produce a development compound, the second lead series will be the focus of our second attempt to find the first development compound. To ensure that the second lead series is available by the time we recognize the failure of our first optimisation attempt,  $u_{2b}$  is allocated so that the time needed to finish stage 2b,  $t_{2b}$ , is no greater than  $t_3$ :  $\frac{X_3}{e(u_3)} = t_3 \geq t_{2b} = \frac{X_{2b}}{e(u_{2b})}$ . For the sake of simplicity, in the optimisation calculations, we set  $t_3 = t_{2b}$ , so that  $u_{2b}$  can be expressed as a function of  $u_3$ :  $u_{2b} = e^{-1} \left( e(u_3) \cdot \frac{X_{2b}}{X_3} \right)$ .

Since our maximum number of attempts to find the first development compound ( $m$ ) is set at 2, no scientist is allocated to carry out stage 2b when we are looking for a development compound in the second lead series. If this second attempt turns out to be unsuccessful, the project is terminated. Thus, increasing  $m$  from 1 to 2 means that we have one more chance for lead series optimisation, and the probability of success for finding the first development compound increases from  $p_3$  to  $p_3 + q_3 p_3$ .

When  $m \geq 3$ , stage 2b will be implemented up to  $m - 1$  times. As mentioned in the previous discussion, once the lead optimisation attempt succeeds, we will go into stage 4 directly and suspend any further effort to find additional lead series.

Note that we are making the following assumptions. First, the effort needed for finding an additional lead series,  $X_{2b}$ , is assumed to remain identical for all other following lead series. In addition, the effort required for lead series optimisation,  $X_3$ , is also assumed to remain unchanged if the process has to be carried out more than once. Third, the number of backup compounds to be looked for in stage 4,  $k$ , is constrained, for the sake of simplicity, to be independent of the number of times stage 3 has been carried out in finding the first development compound.

#### D. Search for a Development Compound from a Different Lead Series

To increase the profitability and the probability of success of a research project, usually two (or more) different development compounds are found from different lead series in practice. When two different development compounds have been found from two different lead series, we shall assume that the two series are used alternately to produce backup compounds. If we are interested in finding development compounds from two different lead series, we have to decide the maximum number of attempts we are prepared to make to find the

second development compound. The variable  $l$  is defined as follows:

- $l$  : maximum number of attempts we are prepared to make to find a development compound from a lead series different from the one that produced the first development compound.

$l$ , as well as  $m$ , takes only integer values. Normally, the value of  $l$  will not be greater than that of  $m$ , because usually we are not willing to make more attempts to find the second development compound than to find the first one. Setting  $l = 0$  means no attempt will be made to search for a second development compound, as discussed in the previous section. When  $m = 1$ , since we assume that only one lead series is identified to carry out the optimisation process and no scientist is allocated to look for another lead series, the value of  $l$  can only be 0. For  $m \geq 2$ , we consider only values of  $l \leq 2$ . This is because in practice usually no more than two attempts would be made to optimise other lead series. In the following discussions,  $(m, l)$  will be used to denote the chosen values of  $m$  and  $l$ .

#### E. $(m, l) = (m, 1)$ , $m \geq 2$

To illustrate the possible consequences for  $(m, l) = (m, 1)$ ,  $m \geq 2$ , we first discuss the event tree for  $m = 2$ . If stage 3 is successful for one lead series, as shown as *Case(1.1)* in Figure 3, we then have the first development compound and a second lead series to be optimised at hand. Since  $l > 0$ , some scientists will be allocated to optimise the second lead series, while other scientists will be searching for backup compounds using the first development compound as the first seed. The probability of success for the second lead series optimisation is still  $p_3$ , and the number of scientists allocated to such work is denoted as  $u_{3b}$ . If this optimisation process for the second lead series succeeds by the time when the first backup compound is obtained, the second backup compound will be taken from the second lead series. Further backup compounds will be taken alternately from these two lead series, and the total number of backup compounds is  $k_1$ . This is *Case(1.1a)* in Figure 3. On the other hand, if the second lead series fails to produce a development compound, all backup compounds will be taken from the single successfully optimised lead series, with no further attempt to optimise an additional lead series, since  $l = 1$ . This is *Case(1.1b)* in figure 3, and the total number of backup compounds taken in this case is  $k_2$ .

If stage 3 is a failure for the first lead series, as at Node 3 in the left-hand side of Figure 3, we still

have a second lead series, from which we repeat stage 3 to look for the first development compound. Thus, the tasks to be carried out at Node 3 are the same as those at Node 1: optimise the lead series at hand and search for an additional lead series. If this second stage 3 succeeds, as illustrated as *Case(1.2)* in Figure 3, we then have the first development compound and a third lead series to be optimised. *Case(1.2a)* and *Case(1.2b)* in Figure 3 are similar to *Case(1.1a)* and *Case(1.1b)*, respectively, except that in the former two cases stage 3 is carried out twice instead of once and thus there is an additional time lag  $t_3$ . On the other hand, if the second attempt at completion of stage 3 unfortunately fails, then the research project terminates since  $m = 2$ , as shown as *Case(F)* in Figure 3.

Observing in the left-hand side of Figure 3 that branches stemming from Node 3 are the same as those from Node 1 when  $m = 2$ , we can analogously extend the event trees for  $(m, l) = (m, 1)$ ,  $m = 3, \dots$ . Furthermore, *Case(1.ia)* and *Case(1.ib)*,  $i = 2, 3, \dots, m$ , are exactly the same as *Case(1.1a)* and *Case(1.2b)*, respectively, after the point where the first development compound is found. Before *Case(1.ia)* or *Case(1.ib)* occurs, stage 3 has failed  $i - 1$  times.

#### F. $(m, l) = (m, 2)$ , $m \geq 2$

The left-hand side of Figure 4 demonstrates the event tree for  $(m, l) = (m, 2)$ ,  $m = 2$ , from which the possible consequences for  $m = 3, \dots$  can be analogously extended. To ensure that two further lead series are available for stage 3, the allocation of scientists must be modified from the point at which stage 3 is first successfully completed, as compared with the case where  $l = 1$ . This point is Node 2 in the figure. With the convention that only two tasks may be carried out at the same time, we would not immediately look for backup compounds from the first lead series from which the first development compound was just obtained. Instead, some scientists make the first attempt to find a development compound on a second lead series, while the other scientists focus on finding further lead series to ensure that the second chance for a second successful completion of stage 3 is available. If the first attempt at completion of stage 3 with a second lead series succeeds, the following series of backup compounds will use the two productive lead series, as illustrated as *Case(2.1a)* in Figure 4, and the total number of backup compounds to be identified is  $k_1$ .

If the first attempt on a second lead series fails, then a second attempt will be made immediately with

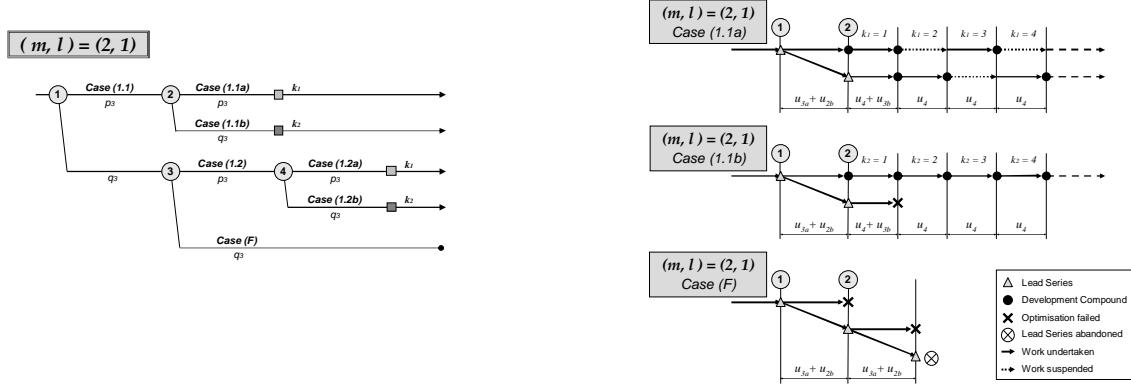


Fig. 3. Event trees and corresponding cases for various  $m$  when  $(m, l) = (m, 1)$ ,  $m \geq 2$ .

the next lead series. If a development compound is successfully obtained from the second attempt, it will be used as the second seed for backup compound searching, which will use these two seeds alternately, as depicted as *Case(2.1b)* in Figure 4. The total number of backup compounds taken in this case is  $k_2$ . However, if the second attempt for the a second development compound searching is also a failure, the following backup compound searching will be solely based on the first lead series. This is *Case(2.1c)* in Figure 4, and  $k_3$  is the total number of backup compounds taken in this case.

If stage 3 is a failure for the first lead series, as at Node 4 in the left-hand side of Figure 4, we still have a second lead series, from which we repeat stage 3 to look for the first development compound. The possible outcomes facing Node 4 are the same as Node 1, and *Case(2.2a)*, *Case(2.2b)*, and *Case(2.2c)* in the figure are similar to *Case(2.1a)*, *Case(2.1b)*, and *Case(2.1c)*, respectively, except that in the former three cases stage 3 is carried out twice instead of once and thus there is an additional time lag  $t_3$ . On the other hand, if the second attempt at completion of stage 3 unfortunately fails, then the research project terminates since  $m = 2$ , as shown as *Case(F)* in both Figures 3 and 4. As discussed in the previous section, event trees for  $(m, l) = (m, 2)$ ,  $m = 3, \dots$ , and *Case(2.1a)*, *Case(2.1b)*, and *Case(2.1c)*,  $i = 2, 3, \dots, m$ , for the  $i^{th}$  attempt at stage 3 can be analogously extended.

In both Figures 3 and 4,  $u_{3a}$  stands for the number of scientists allocated to the first attempt at completion of stage 3, and  $u_{3b}$  is used to denote the number of scientists allocated for optimising the second lead series. The effort needed for finding an additional lead series,  $X_{2b}$ , is assumed to remain identical for all other following lead series. The effort required for the lead

series optimisation,  $X_3$ , is also assumed to remain unchanged. Furthermore, the optimal number of backup compounds taken in each case,  $k_i$ ,  $i = 1, 2$  for  $l = 1$ , or  $i = 1, 2, 3$  for  $l = 2$ , is assumed to be independent of the actual number of attempts undertaken to find the first development compound.

Also note that in both Figures 3 and 4,  $u_{3b}$  scientists are allocated to look for an additional development compound while  $u_4$  scientists are allocated to carry out stage 4. This dual-task implementation is similar to the situation where stage 2b and stage 3 are carried out simultaneously. For the sake of simplicity, we assume that the result of the search for the additional development compound is known by the end of stage 4. Thus,  $\frac{X_4}{e(u_4)} = t_4 = \frac{X_3}{e(u_{3b})}$  and  $u_{3b} = e^{-1}(e(u_4) \cdot \frac{X_3}{X_4})$ .

#### G. Competition Effect between Development Compounds

The expected values for backup compounds are usually assumed to be lower than that of the first compound in a series. There are several reasons. First, each successive backup compound lags an additional time  $t_4$ , and is therefore discounted by an additional factor  $e^{-\gamma_2 t_4}$ . Second, the expected market for each backup compound discovered in this series is reduced by competition from its predecessors. This competition effect is modelled by introducing a backup factor  $\theta$  in the range  $(0, 1)$ . The expected value of the  $j^{th}$  compound in the series is obtained by multiplying the expected value of the first compound by the factor  $\theta^j e^{-j\gamma_2 t_4}$ .

When comparing two backup compounds optimised from two different lead series, the competition effect requires further modelling. The  $\theta$  mentioned in the previous paragraph is applied to backup compounds from the same series. A second factor  $\eta$  is used to model

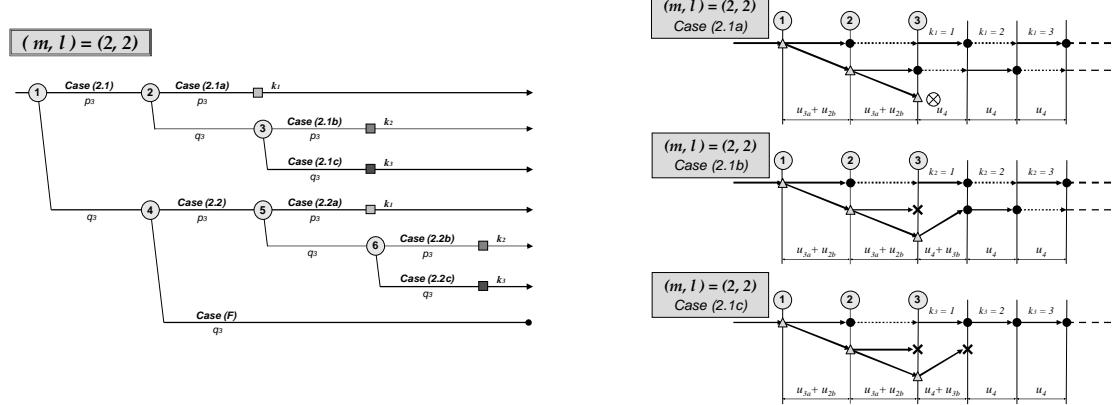


Fig. 4. Event trees and corresponding cases for various  $m$  when  $(m, l) = (m, 2)$ ,  $m \geq 2$ .

the competition between backup compounds developed from two different series. Let  $V(r, s)$  be the value of the next compound where  $r$  is the number of predecessor compounds obtained from *the same* series and  $s$  is the number of predecessor compounds obtained from *the other* series. We have

$$V(0,0) = V, \quad V(r,0) = V\theta^r, \\ V(0,s) = V\eta^s, \quad V(r,s) = V\theta^r\eta^s.$$

Normally, we assume that  $\theta \leq \eta \leq 1$  since the similarity, and hence the competition, between two backup compounds from two different series is usually less than that between two backup compounds from the same series. Furthermore, the probability of the second development compound passing successfully through clinical trials, conditional on failure of the first development compound, is lower than the probability that the first development compound is successful. This is because of the positive correlation between the properties of different compounds, a feature which is stronger if the compounds are from the same lead series. For the extreme case where the two series are identical, it is obvious that  $\theta = \eta$ , indicating that  $V(r, s) = V\theta^{r+s}$ . If the two series are totally independent, then  $\eta = 1$ , and  $V(r, s) = V\theta^r$ .

#### IV. TOTAL EXPECTED REWARD, TOTAL EXPECTED COST, AND OPTIMISATION CALCULATIONS

A. All compounds from the same lead series, i.e.  $(m, l) = (m, 0)$

To estimate the expected rewards of a pharmaceutical research project, first we define  $V$  as the expected value of a development compound available now for clinical trials. This expected value is based on the distribution

of possible cash flows resulting from a new drug. The possibility that the compound may not survive the clinical trials is also taken into account. Recall that  $T$  is the period from the start of the screening process to the time that competitors become aware of our work, assuming no compound has been submitted for clinical trials so far, and that future rewards are discounted at the rate  $\gamma_1$  up to the time  $T$ , or until the first development compound is sent for clinical trial, if this is earlier, and from then on at the rate  $\gamma_2$ .

With a maximum of  $m$  attempts for the lead series optimisation process, the probability that the  $j^{th}$  attempt takes place is  $q_3^{j-1}$  ( $q_3 = 1 - p_3$ ), and the total time spent until the end of the  $j^{th}$  attempt is  $t_2 + j \cdot t_3$ , where  $j = 1, \dots, m$ . If the time  $T$  is due during the  $(j+1)^{th}$  attempt, i.e.,  $t_2 + j \cdot t_3 \leq T < t_2 + (j+1) \cdot t_3$ , the discount rate with self-induced obsolescence,  $\gamma_2$ , should be applied after time  $T$ , and the sum of the discount factors back to the beginning of stage 2 for the last  $m-j$  attempts, starting from the  $(j+1)^{th}$  attempt to the  $m^{th}$  one, is

$$q_3^j e^{-\gamma_2[t_2+(j+1)t_3]+(\gamma_2-\gamma_1)T} \cdot \sum_{i=0}^{m-j-1} q_3^i e^{-\gamma_2 i t_3}.$$

Since the total completion time for the first  $j$  attempts is less than  $T$ , the sum of the discount factors for these first  $j$  attempts is  $\sum_{i=0}^{j-1} q_3^i e^{-\gamma_1[t_2+(1+i)t_3]}$ .

The sum of these two discount series may be written as

$$S(j, m) = e^{-\gamma_1(t_2 + t_3)} \left( \frac{1 - y_1^j}{1 - y_1} \right) + e^{(\gamma_2 - \gamma_1)T - \gamma_2[t_2 + (j+1)t_3]} q_3^j \left( \frac{1 - y_2^{m-j}}{1 - y_2} \right)$$

where  $y_1 = q_3 e^{-\gamma_1 t_3}$ ,  $y_2 = q_3 e^{-\gamma_2 t_3}$ , and

$$j = \begin{cases} 0 & \text{if } T < t_2 + t_3 \\ i & \text{if } t_2 + it_3 \leq T < t_2 + (i+1)t_3, i = 1, \dots, m-1 \\ m & \text{if } t_2 + mt_3 \leq T \end{cases}$$

According to the discussion on the competition effect in the previous section, since all backup compounds are taken from the same lead series ( $l = 0$ ), only  $\theta$  is applied to calculate their values. Thus, the total expected reward with  $k$  backup compounds includes the factor:  $\sum_{i=0}^k (\theta e^{-\gamma_2 t_4})^i$ .

It follows that the total expected reward of the project is:

$$R = V p_1 p_2 p_3 e^{-\gamma_1 t_1} S(j, m) \left( \frac{1 - (\theta e^{-\gamma_2 t_4})^{k+1}}{1 - \theta e^{-\gamma_2 t_4}} \right).$$

The expect cost of the project is measured in scientist-years, and the cost of a scientist usually involves salary, overheads, equipments, accommodation, and technical and secretarial assistance. With discount rate  $\gamma$ , the cost in scientist-year of employing  $u$  scientists for  $t$  years is  $\int_0^t ue^{-\gamma s} ds = \frac{u}{\gamma} (1 - e^{-\gamma t})$ . As a result, the expected cost for stages 1 and 2 is  $\frac{u_1}{\gamma} (1 - e^{-\gamma t_1}) + p_1 e^{-\gamma t_1} \frac{u_{2a}}{\gamma} (1 - e^{-\gamma t_2})$ .

When a maximum of  $m$  attempts for lead series optimisation are scheduled, stage 3 could be carried out at most  $m$  times. Considering the probability of occurrence of the  $j^{th}$  attempt,  $q_3^{(j-1)}$ ,  $j = 1, \dots, m$ , and the time required to complete this stage,  $t_3$ , the expected cost for stage 3 discounted to the beginning of the project is  $p_1 p_2 e^{-\gamma(t_1+t_2)} \frac{u_3}{\gamma} (1 - e^{-\gamma t_3}) \left[ \frac{1 - (q_3 e^{-\gamma t_3})^m}{1 - q_3 e^{-\gamma t_3}} \right]$ .

In addition, when  $m > 1$ , stage 2b is repeated at most  $m-1$  times since the first lead series has been found. Thus, the total discounted expected cost for stage 2b is  $p_1 p_2 e^{-\gamma(t_1+t_2)} \frac{u_{2b}}{\gamma} (1 - e^{-\gamma t_{2b}}) \left( \frac{1 - (q_3 e^{-\gamma t_3})^{m-1}}{1 - q_3 e^{-\gamma t_3}} \right)$ . Recall that in the previous discussion we suggested that time spent on stage 2b is equal to the time for stage 3. Consequently,  $t_3$ , instead of  $t_{2b}$ , will be used in the following optimisation calculations.

The total expected cost for a 4-stage research project, including  $k$  repeats of stage 4 so that a total of  $k+1$  compounds will proceed for clinical trials if the first one is successfully obtained, is listed below:

$$\begin{aligned} C = & \frac{u_1}{\gamma} (1 - e^{-\gamma t_1}) + p_1 e^{-\gamma t_1} \frac{u_{2a}}{\gamma} (1 - e^{-\gamma t_2}) \\ & + p_1 p_2 e^{-\gamma(t_1+t_2)} \frac{u_3}{\gamma} (1 - e^{-\gamma t_3}) \left( \frac{1 - y^m}{1 - y} \right) \\ & + p_1 p_2 p_3 e^{-\gamma(t_1+t_2+t_3)} \frac{u_4}{\gamma} (1 - e^{-\gamma k t_4}) \left( \frac{1 - y^m}{1 - y} \right) \\ & + p_1 p_2 e^{-\gamma(t_1+t_2)} \frac{u_{2b}}{\gamma} (1 - e^{-\gamma t_3}) \left( \frac{1 - y^{m-1}}{1 - y} \right) \end{aligned}$$

where  $y = q_3 e^{-\gamma t_3}$ .

### B. One Attempt Allowed for Second Lead Series Optimisation, $(m, l) = (m, 1)$

In Case(1.1a) in Figure 3, the first development compound is found at Node 2, from where we start stage 4. After spending time  $t_4$  to complete this stage, we successfully obtained a backup compound taken from the first development compound and a second development compound from a different lead series. This is the first completion of stage 4, making the value of  $k_1$  to be 1. According to the aforementioned competition effect, the value of the backup compound includes the factor  $\theta e^{-\gamma_2 t_4}$  and the value of the second development compound includes the factor  $\eta e^{-\gamma_2 t_4}$ . Since the second backup compound will be taken from the second lead series, its value includes the factor  $\theta \eta^2 e^{-2\gamma_2 t_4}$ . Further backup compounds will be taken alternately from the two lead series. It follows that the sum of the values of all compounds includes the factor  $1 + (\theta + \eta)e^{-\gamma_2 t_4} + \theta \eta^2 e^{-2\gamma_2 t_4} + \theta^2 \eta^2 e^{-3\gamma_2 t_4} + \dots$ , where the number of terms depends on the value of  $k_1$ . This sum can be expressed as  $1 + ERX$ , where

$$ERX = \begin{cases} (\theta + \eta)e^{-\gamma_2 t_4} + \theta \eta^2 e^{-2\gamma_2 t_4}, \\ (1 + \theta e^{-\gamma_2 t_4}) \left( \frac{1 - z_2^{k_x}}{1 - z_2} \right) \\ \quad \text{if } k_1 \text{ is odd, } k_x = \frac{k_1-1}{2}, k_x = 0, 1, \dots \\ (\theta + \eta)e^{-\gamma_2 t_4} + \theta \eta^2 e^{-2\gamma_2 t_4} \\ \quad + \theta^2 \eta^2 e^{-3\gamma_2 t_4} (1 + \eta e^{-\gamma_2 t_4}) \left( \frac{1 - z_2^{k_x}}{1 - z_2} \right) \\ \quad \text{if } k_1 \text{ is even, } k_x = \frac{k_1-2}{2}, k_x = 0, 1, \dots \end{cases}$$

with  $z_2 = \theta \eta e^{-2\gamma_2 t_4}$ .

In Case(1.1b) in Figure 3, the second lead series fails to produce a second development compound, and a total of  $k_2$  backup compounds will be taken all from the same lead series as the first development compound. Thus, the sum of the values of all these compounds includes the factor  $\frac{1 - (\theta e^{-\gamma_2 t_4})^{k_2+1}}{1 - \theta e^{-\gamma_2 t_4}}$ .

Note that in *Case(1.ia)* and *Case(1.1b)*,  $i = 2, 3, \dots, m$ , the first development compound is not found until our  $i^{th}$  attempt at stage 3. This feature has been included in the function  $S(j, m)$ . Moreover, we realise that from Node 2 in Figure 3 the probabilities for *Case(1.1a)* and *Case(1.1b)* to happen are  $p_3$  and  $q_3$ , respectively. Therefore, the total expected value for a research project when  $(m, l) = (m, 1)$ ,  $m \geq 2$ , can be expressed as:

$$R = V p_1 p_2 p_3 e^{-\gamma_1 t_1} S(j, m) \cdot \\ \left\{ p_3 (1 + ERX) + q_3 \left( \frac{1 - (\theta e^{-\gamma_2 t_4})^{k_2+1}}{1 - \theta e^{-\gamma_2 t_4}} \right) \right\},$$

where  $S(j, m)$ ,  $j$ , and  $ERX$  are as discussed above.

As for the total expected cost, from Figure 3 we realise that since  $l = 1$ , the number of scientists,  $u_{3b}$ , is allocated only once, during the first stage 4. For *Case(1.1a)* and *Case(1.1b)*, the difference between them is that  $k_1$  backup compounds are taken in *Case(1.1a)* and  $k_2$  in *Case(1.1b)*. As a consequence, the total expected cost for a 4-stage research project when  $(m, l) = (m, 1)$  can be expressed as :

$$C = \frac{u_1}{\gamma} (1 - e^{-\gamma t_1}) + p_1 e^{-\gamma t_1} \frac{u_{2a}}{\gamma} (1 - e^{-\gamma t_2}) \\ + p_1 p_2 e^{-\gamma(t_1+t_2)} \frac{u_{3a} + u_{2b}}{\gamma} (1 - e^{-\gamma t_3}) \left( \frac{1 - y^m}{1 - y} \right) \\ + p_1 p_2 p_3 e^{-\gamma(t_1+t_2+t_3)} \left( \frac{1 - y^m}{1 - y} \right) \cdot \\ \left[ \frac{u_{3b}}{\gamma} w + \frac{u_4}{\gamma} (p_3 w_1 + q_3 w_2) \right]$$

where  $y = q_3 e^{-\gamma t_3}$ ,  $w = 1 - e^{-\gamma t_4}$ ,  $w_1 = 1 - e^{-k_1 \gamma t_4}$ , and  $w_2 = 1 - e^{-k_2 \gamma t_4}$ .

#### C. Two Attempts Allowed for Second Lead Series Optimisation, $(m, l) = (m, 2)$

Based on the analysis similar to the above, the total expected reward,  $R$ , and the total expected cost,  $C$ , when  $(m, l) = (m, 2)$ ,  $m \geq 2$ , can be expressed as :

$$R = V p_1 p_2 p_3 e^{-\gamma_1 t_1} S(j, m) \cdot \\ \left\{ p_3 (1 + ERY) + q_3 p_3 (1 + e^{-\gamma_2 t_3} ERZ) \right. \\ \left. + q_3^2 \left[ 1 + e^{-\gamma_2 t_3} z_1 \left( \frac{1 - z_1^{k_3+1}}{1 - z_1} \right) \right] \right\},$$

where  $z_1 = \theta e^{-\gamma_2 t_4}$  and  $ERY$  and  $ERZ$  are listed

below, with  $z_2 = \theta \eta e^{-2\gamma_2 t_4}$  :

$$ERY = \begin{cases} \eta e^{-\gamma_2 t_3} + \theta \eta e^{-\gamma_2 t_4} \\ + \theta \eta^2 e^{-2\gamma_2 t_4} (1 + \theta e^{-\gamma_2 t_4}) \left( \frac{1 - z_2^{k_y}}{1 - z_2} \right) \\ \text{if } k_1 \text{ is odd, } k_y = \frac{k_1-1}{2}, \quad k_y = 0, 1, \dots \end{cases}$$

$$\begin{cases} \eta e^{-\gamma_2 t_3} + \theta \eta e^{-\gamma_2 t_4} (1 + \eta e^{-\gamma_2 t_4}) \left( \frac{1 - z_2^{k_y}}{1 - z_2} \right) \\ \text{if } k_1 \text{ is even, } k_y = \frac{k_1}{2}, \quad k_y = 0, 1, \dots \end{cases}$$

$$ERZ = \begin{cases} (\theta + \eta) e^{-\gamma_2 t_4} + \theta \eta^2 e^{-2\gamma_2 t_4} \cdot \\ (1 + \theta e^{-\gamma_2 t_4}) \left( \frac{1 - z_2^{k_z}}{1 - z_2} \right) \\ \text{if } k_2 \text{ is odd, } k_z = \frac{k_2-1}{2}, \quad k_z = 0, 1, \dots \end{cases}$$

$$\begin{cases} (\theta + \eta) e^{-\gamma_2 t_4} + \theta \eta^2 e^{-2\gamma_2 t_4} \\ + \theta^2 \eta^2 e^{-3\gamma_2 t_4} (1 + \eta e^{-\gamma_2 t_4}) \left( \frac{1 - z_2^{k_z}}{1 - z_2} \right) \\ \text{if } k_2 \text{ is even, } k_z = \frac{k_2-2}{2}, \quad k_z = 0, 1, \dots \end{cases}$$

$$C = \frac{u_1}{\gamma} (1 - e^{-\gamma t_1}) + p_1 e^{-\gamma t_1} \frac{u_{2a}}{\gamma} (1 - e^{-\gamma t_2}) \\ + p_1 p_2 e^{-\gamma(t_1+t_2)} \frac{u_{3a} + u_{2b}}{\gamma} (1 - e^{-\gamma t_3}) \cdot \\ \left( 1 + p_3 e^{-\gamma t_3} \right) \left( \frac{1 - y^m}{1 - y} \right) \\ + p_1 p_2 p_3 e^{-\gamma(t_1+t_2+t_3)} \left( \frac{1 - y^m}{1 - y} \right) e^{-\gamma t_3} \cdot \\ \left[ q_3 \frac{u_{3b}}{\gamma} w + \frac{u_4}{\gamma} (p_3 w_1 + q_3 p_3 w_2 + q_3^2 w_3) \right],$$

where  $y = q_3 e^{-\gamma t_3}$ ,  $w = 1 - e^{-\gamma t_4}$ ,  $w_1 = 1 - e^{-k_1 \gamma t_4}$ ,  $w_2 = 1 - e^{-k_2 \gamma t_4}$ , and  $w_3 = 1 - e^{-k_3 \gamma t_4}$ .

#### D. Optimisation Calculations with Profitability Index Criterion

One of our objectives has been to build a stochastic optimisation model to find the allocation of  $(u_1, u_2^\dagger, u_3^\dagger, u_4, m, l, k^\dagger)$ <sup>1</sup> that maximises the pharmaceutical research project's profitability index, the ratio of total expected reward and total expected cost:

$$P(u_1, u_2^\dagger, u_3^\dagger, u_4, m, l, k^\dagger) = \frac{R(u_1, u_2^\dagger, u_3^\dagger, u_4, m, l, k^\dagger)}{C(u_1, u_2^\dagger, u_3^\dagger, u_4, m, l, k^\dagger)}.$$

The profitability index is a widely used criterion for the evaluation of business activities and research projects. It represents the expected reward obtained per

<sup>1</sup>To simplify the notation,  $u_2^\dagger$  stands for  $u_{2a}$  and  $u_{2b}$ ,  $u_3^\dagger$  stands for  $u_3$  in the  $(m, l) = (m, 0)$  model, and for  $u_{3a}$  and  $u_{3b}$  in the  $(m, l) = (m, 1)$  and  $(m, l) = (m, 2)$  models. Similarly,  $k^\dagger$  means  $k$  in the  $(m, l) = (m, 0)$  model,  $k_1$  and  $k_2$  in the  $(m, l) = (m, 1)$  model, and  $k_1$ ,  $k_2$ , and  $k_3$  in the  $(m, l) = (m, 2)$  model.

unit of expected cost.

Since the Newton-Raphson algorithm is employed to maximise the profitability index  $P$ , its first partial derivatives with respect to the decision variables are required. The maximum occurs at a point at which these first partial derivatives are all equal to zero, so that

$$\frac{\partial P}{\partial x} = \frac{C \frac{\partial R}{\partial x} - R \frac{\partial C}{\partial x}}{C^2} = 0,$$

where  $x \in (u_1, u_2^\dagger, u_3^\dagger, u_4, k^\dagger)$ .

#### E. Optimisation Calculations with Internal Rate of Return

Let  $\alpha$  denote the monetary cost of a scientist-year. The internal rate of return of a project is defined to be the discount rate  $\gamma_I$  that satisfies the equation:  $G(\gamma_I, u_1, u_2^\dagger, u_3^\dagger, u_4, m, l, k^\dagger) = 0$ , where

$$\begin{aligned} & G(\gamma_I, u_1, u_2^\dagger, u_3^\dagger, u_4, m, l, k^\dagger) \\ &= R(\gamma_I, u_1, u_2^\dagger, u_3^\dagger, u_4, m, l, k^\dagger) \\ &\quad - \alpha C(\gamma_I, u_1, u_2^\dagger, u_3^\dagger, u_4, m, l, k^\dagger). \end{aligned}$$

For the internal rate of return criterion, the discount rates applied to model obsolescence are further broken down into components as follows:

$$\begin{aligned} \gamma_1 &= \gamma + \delta \\ \gamma_2 &= \gamma + \delta + \zeta. \end{aligned}$$

Here  $\delta = f\xi$  is the rate due to obsolescence, and  $\zeta$  is the rate due to the additional self-induced obsolescence effect. It is important to break the discount rates into these components because  $\delta$  and  $\zeta$  remain constant while we adjust  $\gamma$  to look for the internal rate of return of a research project.

The profitability index ranks research projects according to the expected net present value generated per unit of expenditure. In contrast, the internal rate of return of a research project can be regarded as the rate of growth of capital within the project. Our overall objective is to investigate resource allocations which improve profitability, using these two criteria as guides. Note that the profitability index and internal rate of return criteria usually lead to different 'optimal' allocations.

Since the equation  $G(\gamma_I, u_1, u_2^\dagger, u_3^\dagger, u_4, m, l, k^\dagger) = 0$  holds for all values of the parameters, it follows that  $\frac{\partial G}{\partial \gamma} d\gamma + \frac{\partial G}{\partial x} dx = 0$ , where  $x \in (u_1, u_2^\dagger, u_3^\dagger, u_4, k^\dagger)$ , and

$dx$  is a small change in  $x$  and  $d\gamma$  is the resulting change in  $\gamma$ . It follows that

$$\frac{\partial \gamma_I}{\partial x} = -\frac{\frac{\partial G}{\partial x}}{\frac{\partial G}{\partial \gamma_I}},$$

and

$$\frac{\partial^2 \gamma_I}{\partial x^2} = -\left( \frac{\frac{\partial G}{\partial \gamma_I} \cdot \frac{\partial^2 G}{\partial x^2} - \frac{\partial G}{\partial x} \cdot \frac{\partial^2 G}{\partial x \partial \gamma_I}}{\left( \frac{\partial G}{\partial \gamma_I} \right)^2} \right),$$

$$\text{where } \frac{\partial G}{\partial x} = \frac{\partial R}{\partial x} - \alpha \frac{\partial C}{\partial x}, \quad \frac{\partial^2 G}{\partial x^2} = \frac{\partial^2 R}{\partial x^2} - \alpha \frac{\partial^2 C}{\partial x^2}, \\ \frac{\partial G}{\partial \gamma_I} = \frac{\partial R}{\partial \gamma_I} - \alpha \frac{\partial C}{\partial \gamma_I}, \quad \frac{\partial^2 G}{\partial x \partial \gamma_I} = \frac{\partial^2 R}{\partial x \partial \gamma_I} - \alpha \frac{\partial^2 C}{\partial x \partial \gamma_I}.$$

To provide reasonable values for the number of scientists to be allocated to each stage, each variable  $u_i$ ,  $i = 1, 2a, 2b, 3, 3a, 3b, 4$  is subject to an upper limit, which is set to be 100 in this paper. Recall that  $u_{2b}$  and  $u_{3b}$  can be expressed as functions of  $u_{3a}$  and  $u_4$ , respectively. If either  $u_{3a}$  or  $u_{2b}$  is greater than this upper limit, the greater one is set to be 100, and the other one is determined as described in section II.C. The same procedure applies if either  $u_4$  or  $u_{3b}$  is greater than 100.

To perform these optimisation calculations, expressions for more than 200 derivatives are required. A **FORTRAN 90** programme has been written which enables users to key in input parameters to obtain the corresponding optimal solutions.

## V. PARAMETER VALUES

To keep our illustrative examples simple, the team-size effectiveness function for each stage except stage 1 was chosen to be identical. The most efficient team-size ( $u_{opt}$ ) was assumed to be 15 scientists. *DBEFF* was assumed to be 0.9, so that the team would suffer a 10% loss of efficiency if the team size were to be doubled. These two assumptions can be summarised as  $e(15) = 15$  and  $e(30) = 27$ . Usually fewer chemists are needed before the screening of compounds begins as a project at that stage tends to be in the hands of bioscientists. For this reason the most efficient team-size is set to be 7.5 for stage 1. For stage 1, *DBEFF* was again assumed to be 0.9.

The completion times for stages 1, 2, 3, and 4 are taken to be 2, 2, 2, 1 year(s) respectively, with team-size of 8, 20, 20, and 20 scientists. As for stage 2b, on

the assumption that the effort required for each of the  $2^{nd}, \dots, m^{th}$  lead series is the same and is less than that needed for the first one, two possible completion times for stage 2b are used in the following calculations, 1 and 1.5 year(s), still with a team-size of 20 scientists. These input parameters are used to obtain the effort required to complete each stage, measured in scientist-years, from the equation  $X = e(u)t$ .

The expected value of the first development compound if it is available now for clinical trials,  $V$ , is set at £40,000,000. The monetary cost of a scientist-year,  $\alpha$ , includes not only the annual salary of a scientist, but also overhead, accommodation, and administrative expenses associated with the scientist. The cost of a scientist definitely varies from scientist to scientist depending on his/her position and the work he/she does. But for illustrative purposes, we may assume that the cost of a scientist-year is around £100,000 on average.

The exponential discount rate  $\gamma$  is set at 0.09, corresponding to an effective annual interest rate of 9.4% in real terms. To determine the values of  $\gamma_1$  and  $\gamma_2$ , recall that  $\gamma_1 = \gamma + f\xi$ . Here  $\xi\Delta t$  is the probability of a competitor sending a compound for clinical trials in a short time interval  $\Delta t$ , and  $f$  is the expected fraction by which a competitor's earlier development compound reduces the value of a development compound. We see that  $\xi = 0.25$  means that other companies, in total, are generating development compounds at the rate of one in every four years. On the assumption that  $f = 0.24$ , we use  $\gamma_1 = 0.09 + 0.25 \times 0.24 = 0.15$  in our optimisation calculations. Furthermore, the exponential discount rate including self-induced obsolescence,  $\gamma_2$ , is assumed to be 0.40. Thus, calculations for the profitability index are carried out with discount rates  $(\gamma_1, \gamma_2) = (0.15, 0.40)$ . For calculations for the internal rate of return, since  $\gamma_1 = \gamma + \delta$  and  $\gamma_2 = \gamma + \delta + \zeta$ , we have  $\delta = 0.06$  and  $\zeta = 0.25$ .

The factor  $\eta$  for the competition effect between development compounds from *two different* series is assumed to be 0.9. The probability of a development compound surviving clinical trials and ultimately becoming a marketable new drug is at most 20%. On the assumption that any development compound discovered after the first marketable new drug is usually less valuable because the market would have been dominated by the first one, the probability of surviving clinical trials of 20% can be translated to  $\eta = 0.8$ . The figure of 0.9 allows for the fact that a later development compound might still be marketable,

despite a reduction in value because of drugs marketed earlier. Two alternative values, 0.8 and 0.6, are assumed for  $\theta$ , the competition effect factor for development compounds from *the same* series. These two values represent a low and a high level of similarity between compounds, respectively.

For the probabilities of success for stages 1, 2, and 3, four combinations are used: (0.3, 0.3, 0.5), (0.3, 0.5, 0.7), (0.5, 0.5, 0.5), and (0.5, 0.7, 0.7), representing unfavourable, general, average, and favourable conditions, respectively. Note that each of the three cases,  $(m, l) = (m, 0)$ ,  $(m, l) = (m, 1)$ , and  $(m, l) = (m, 2)$ , is optimised and that two values for  $t_{2b}$  are assumed. Thus, for each case, the profitability index and the internal rate of return ( $\gamma_I$ ) are calculated with  $t_{2b} = 1.0$  and  $t_{2b} = 1.5$ , respectively, for each of the 8 combinations of  $(\theta, \eta)$  and  $(p_1, p_2, p_3)$ .

## VI. RESULTS

Tables 1 and 2 show the values of  $m$ ,  $l$ , and the other control variables that produce the highest profitability index ( $PI$ ) and internal rate of return ( $IRR$ ) for each combination of  $(\theta, \eta)$  and  $(p_1, p_2, p_3)$  with  $t_{2b} = 1.0$  and  $t_{2b} = 1.5$ , respectively. An \* is attached to each of these variables in the tables to denote that they produce the best results. The first column in each table is an abbreviated code for  $(p_1, p_2, p_3)$ ; for example, 335 means  $(p_1, p_2, p_3) = (0.3, 0.3, 0.5)$ .

The most striking observation from these two tables is that when  $\theta$  is 0.6, which means there is strong competition between successive backup compounds taken from the same series, the best value of  $l$  is 1, whereas when  $\theta = 0.8$ , which means there is less competition between development compounds from the same series, the best value of  $l$  is 0. Thus, if the competition between the successive backup compounds taken from the same series is high, then it is worth searching for a second development compound from a different lead series and using these two series alternately to look for backup compounds. This outcome is as we might have expected.

We can also observe that, when  $t_{2b}$  increases from 1.0 year to 1.5 years, the suggested value of  $m$  decreases for most combinations of  $(\theta, \eta)$  and  $(p_1, p_2, p_3)$ . This is also a reasonable result since as it takes more time and effort to find an additional lead series, we may be reluctant to search for more lead series.

$(\theta, \eta) = (0.6, 0.9), t_{2b} = 1.0$ 

$p_{123}$	$m^*$	$l^*$	$u_1^*$	$u_2^*$	$u_{2b}^*$	$u_3^*$	$u_{3b}^*$	$u_4^*$	$k_1^*$	$k_2^*$	$k_3^*$	$PI$
335	5	1	9	57	32	100	100	31	4	3	0	79325.63
555	4	1	11	59	32	100	100	31	4	3	0	116023.69
357	4	1	9	54	32	100	100	31	4	3	0	133454.46
577	4	1	12	52	26	76	100	31	3	2	0	172841.00
$p_{123}$	$m^*$	$l^*$	$u_1^*$	$u_2^*$	$u_{2b}^*$	$u_3^*$	$u_{3b}^*$	$u_4^*$	$k_1^*$	$k_2^*$	$k_3^*$	$IRR$
335	5	1	8	49	32	100	100	31	4	3	0	-0.04
555	4	1	11	64	32	100	100	31	4	3	0	0.20
357	5	1	10	73	32	100	100	31	4	3	0	0.27
577	5	1	14	90	32	100	100	31	4	3	0	0.49

 $(\theta, \eta) = (0.8, 0.9), t_{2b} = 1.0$ 

$p_{123}$	$m^*$	$l^*$	$u_1^*$	$u_2^*$	$u_{2b}^*$	$u_3^*$	$u_{3b}^*$	$u_4^*$	$k_1^*$	$k_2^*$	$k_3^*$	$PI$
335	5	0	9	59	32	100	0	100	6	0	0	102743.98
555	3	0	11	57	32	100	0	83	5	0	0	149190.93
357	3	0	9	53	32	100	0	79	5	0	0	158931.36
577	2	0	11	46	24	67	0	63	4	0	0	210015.44
$p_{123}$	$m^*$	$l^*$	$u_1^*$	$u_2^*$	$u_{2b}^*$	$u_3^*$	$u_{3b}^*$	$u_4^*$	$k_1^*$	$k_2^*$	$k_3^*$	$IRR$
335	4	0	13	61	32	100	0	100	6	0	0	0.10
555	4	0	17	82	32	100	0	100	6	0	0	0.37
357	4	0	15	94	32	100	0	100	6	0	0	0.38
577	4	0	19	100	32	100	0	100	6	0	0	0.63

TABLE I  
RESULTS FOR  $t_{2b} = 1.0$ .

 $(\theta, \eta) = (0.6, 0.9), t_{2b} = 1.5$ 

$p_{123}$	$m^*$	$l^*$	$u_1^*$	$u_2^*$	$u_{2b}^*$	$u_3^*$	$u_{3b}^*$	$u_4^*$	$k_1^*$	$k_2^*$	$k_3^*$	$PI$
335	5	1	9	58	60	100	100	31	4	3	0	75507.30
555	3	1	10	56	43	69	100	31	5	3	0	107399.27
357	5	1	8	54	44	71	100	31	5	3	0	125282.33
577	2	1	12	50	37	58	100	31	3	2	0	163301.35
$p_{123}$	$m^*$	$l^*$	$u_1^*$	$u_2^*$	$u_{2b}^*$	$u_3^*$	$u_{3b}^*$	$u_4^*$	$k_1^*$	$k_2^*$	$k_3^*$	$IRR$
335	4	1	8	49	60	100	100	31	4	3	0	-0.08
555	4	1	12	64	60	100	100	31	4	3	0	0.15
357	4	1	10	70	60	100	100	31	4	3	0	0.24
577	4	1	14	86	60	100	100	31	4	3	0	0.46

 $(\theta, \eta) = (0.8, 0.9), t_{2b} = 1.5$ 

$p_{123}$	$m^*$	$l^*$	$u_1^*$	$u_2^*$	$u_{2b}^*$	$u_3^*$	$u_{3b}^*$	$u_4^*$	$k_1^*$	$k_2^*$	$k_3^*$	$PI$
335	4	0	9	60	60	100	0	100	6	0	0	98406.23
555	2	0	11	51	60	100	0	90	5	0	0	141103.80
357	2	0	9	51	60	100	0	84	5	0	0	152204.60
577	2	0	11	47	39	61	0	65	4	0	0	200466.04
$p_{123}$	$m^*$	$l^*$	$u_1^*$	$u_2^*$	$u_{2b}^*$	$u_3^*$	$u_{3b}^*$	$u_4^*$	$k_1^*$	$k_2^*$	$k_3^*$	$IRR$
335	4	0	13	60	60	100	0	100	6	0	0	0.07
555	3	0	17	78	60	100	0	100	6	0	0	0.34
357	3	0	15	91	60	100	0	100	6	0	0	0.36
577	2	0	19	100	60	100	0	100	6	0	0	0.60

TABLE II  
RESULTS FOR  $t_{2b} = 1.5$ .

Note that in cases where  $l = 1$ ,  $k_1$  is always greater than  $k_2$ . Recall that  $k_1$  is the number of backup compounds taken when our attempt at the search for a second development compound is successful while  $k_2$  is the counterpart when the attempt fails. The result that  $k_1 > k_2$  indicates that, to increase the profitability of the research project, it is worth taking more backup

compounds when we find two development compounds from different lead series.

Most of the optimal numbers of scientists allocated to each stage are much greater than 50. In fact, many of them reach the upper limit we set in the models (100), indicating that the unconstrained optima for these

variables are even greater. These big values must be treated with great caution because they are obtained from the extrapolation of the team-size effectiveness function well beyond the range of values at which it was fitted, namely, 15 and 30. Furthermore, since our model of the research process is simpler than the real process, it is unwise to accept the resulting optimal allocation as definitive.

## VII. CONCLUSIONS

In this paper, a mathematical model for improving the profitability of a pharmaceutical research project is developed using both the profitability index and the internal rate of return criteria. Maximum numbers of attempts to look for the first development compound and another one from a different lead series, termed  $m$  and  $l$ , respectively, are introduced. The optimal numbers of scientists to be allocated to each research stage and the optimal number of backup compounds to be identified are also calculated.

The purposes of this paper are to set out an extended version of the model described in [12] and to show that this model gives reasonable results for some illustrative examples. There is scope for further refinement of our model to make it more realistic, such as variations of the event trees currently used in our model and use of a team-size effectiveness function based on a  $K - fold$  increase in team size.

## REFERENCES

- [1] S. W. Bergman and J. C. Gittins. *Statistical Methods for Pharmaceutical Research Planning*. Marcel Dekker, New York, 1985.
- [2] F. Black and M. Scholes. The pricing of options and corporate liabilities. *Journal of Political Economics*, 81:637–659, 1973.
- [3] R. A. A. Boschi. Modelling exploratory research. *European Journal of Operations Research*, 10:250–259, 1982.
- [4] E. H. Bowman and G. T. Moskowitz. Real options analysis and strategic decision making. *Organization Science*, 12:772–777, 2001.
- [5] R. Brealey and S. Myers. *Principles of Corporate Finance*. McGraw-Hill, 2000.
- [6] C.-F. Burman and S. Senn. Examples of option values in drug development. *Pharmaceutical Statistics*, 2:113–125, 2003.
- [7] B. P. K. Chen. *Prioritization of Research Projects in the Pharmaceutical Industry*. DPhil. thesis, Department of Statistics, University of Oxford, Oxford, UK, 2004.
- [8] M. Ding and J. Eliashberg. Structuring the new product development pipeline. *Management Science*, 48:3:343–363, 2002.
- [9] A. K. Dixit and R. S. Pindyck. *Investment Under Uncertainty*. Princeton University Press, Princeton, New Jersey, 1994.
- [10] J. C. Gittins. Quantitative methods in the planning of pharmaceutical research. *Drug Information Journal*, 30:479–487, 1996.
- [11] J. C. Gittins. Algorithms for allocating resources to multi-stage pharmaceutical research projects. In A. H. Christer, S. Osaki, and L. C. Thomas, editors, *Stochastic Modelling in Innovative Manufacturing*, pages 123–139. Springer, 1997.
- [12] J. C. Gittins. Why crash pharmaceutical research? *R&D Management*, 27:79–85, 1997.
- [13] G. Islei et al. Modeling strategic decision making and performance measurements at ICI Pharmaceuticals. *Interfaces*, 21:4–22, 1991.
- [14] D. V. Lindley. *Making Decisions*. John Wiley & Sons, London, 1991.
- [15] C. H. Loch and K. Bode-Greuel. Evaluating growth options as sources of value for pharmaceutical research projects. *R&D Management*, 31:231–248, 2001.
- [16] P. McNamee and J. Celona. *Decision analysis with SuperTree*. The Scientific Press, San Francisco, 1990.
- [17] M. Metcalf and J. Reid. *Fortran 90/95 Explained*. Oxford University Press, Oxford, 1996.
- [18] P. Miller. Role of pharmacoeconomic analysis in R&D decision making. *Pharmacoeconomics*, 23:1–12, 2005.
- [19] M. Perlitz, T. Peske, and R. Schrank. Real options valuation: the new frontier in R&D project evaluation? *R&D Management*, 29:255–269, 1999.
- [20] K. L. Poh, B. W. Ang, and F. Bai. A comparative analysis of R&D project evaluation methods. *R&D Management*, 31:63–75, 2001.
- [21] B. Spilker. *Multinational Drug Companies – Issues in Drug Discovery and Development*. Raven Press, New York, 1989.
- [22] J. S. Stonebraker. How Bayer makes decisions to develop new drugs. *Interfaces*, 32:77–90, 2002.
- [23] W. T. Vetterling, W. H. Press, S. A. Teukolsky and B. P. Flannery. *Numerical recipes in FORTRAN 90 : the art of parallel scientific computing*. Cambridge University Press, Cambridge, 1992.
- [24] J.-Y. Yu. *Software for Improving the Profitability of Pharmaceutical Research*. MSc. thesis, Department of Statistics, University of Oxford, Oxford, UK, September 2002.
- [25] J.-Y. Yu. *Models and Software for Improving the Profitability of Pharmaceutical Research*. DPhil. transfer thesis, Department of Statistics, University of Oxford, Oxford, UK, December 2004.

# Application of U-Lines principles to the Assembly Line Worker Assignment and Balancing Problem (UALWABP). A model and a solving procedure.

Cristóbal Miralles<sup>\*</sup>, José Pedro García<sup>†</sup> and Carlos Andrés<sup>‡</sup>

Depto. Organización de Empresas - UPV  
Camí de Vera s/n, 46071  
Valencia, Spain

<sup>\*</sup> [cmiralles@omp.upv.es](mailto:cmiralles@omp.upv.es)

<sup>†</sup> [jpgarcia@omp.upv.es](mailto:jpgarcia@omp.upv.es)

<sup>‡</sup> [candres@omp.upv.es](mailto:candres@omp.upv.es)

**Abstract**—The Assembly Line Worker Assignment and Balancing Problem arises in those assembly lines where we have certain limited resources available in which the operation time for every task is different depending on who executes the task, and where there are also some task-worker incompatibilities defined. This is quite usual in Sheltered Work Centres for disabled where each worker has different limitations and operation times. This paper deals with the extension of this problem that suppose the application of U-line principles in this environment. The problem has been named as UALWAB and both a valid IP model and a Laser Search method for the problem are presented and tested against experimental and real data of a Sheltered Work Centre; whose application can suppose its growth, providing more jobs for more disabled, but always considering the specific limitations that these workers have. In this sense this paper shows one of the real applications where OR can help not only to get economic and productive benefits but also certain social aims.

**Keywords**—Assembly Line Balancing, U-lines

## I. INTRODUCTION

There are about 386 million disabled people between the age of 16 and 64, normally with very high unemployment rates from 13% to even 80% in certain countries. Current practices for the treatment of the physically and/or mentally handicapped prescribe meaningful job activity as a mean towards a more fulfilling life and societal integration [2]. In many countries, these practices have facilitated the development of many Sheltered Work centres for Disabled (from now on SWD) where fortunately disabled people can get a job in the same way as any other person. This model tries to get away from the traditional stereotype that

considers disabled people as not able to develop a continuous professional work. Just as any other firm, a SWD compete in real markets and must be flexible and efficient enough to adapt to market variations. The only difference is that the SWD is a Not-For-Profit organisation. Thus, the potential benefit that may be obtained from being efficient usually improves the growth of the SWD. This means: more jobs for disabled people, which is in fact the real primary aim of every SWD. Therefore different key policies issues, like proposed in [15], are desirable in order to achieve this primary aim.

One of the policies that can provide more efficiency in the social and labour integration of disabled is the adoption of assembly lines as productive configuration in these centres. As it is well known, in an ordinary work environment this configuration can be very harmful for the workers implied, if no more considerations of job rotation, work enrichment or other techniques are taken into account [12]. But in the SWD specific environment, assembly lines can provide many advantages, since the traditional division of work in single tasks may become a perfect tool for making certain worker disabilities invisible. In fact, an appropriate task assignment can even become a good therapeutic method for certain disabilities rehabilitation. However, some specific constraints relative to time variability arise in this environment, mainly due to the great difference among the mean operation times for each task depending on which person executes it.

Assembly Line literature is mainly based on fixed operation times. This simplification is only justified

in those cases where operation time variation is small enough, but is not valid in SWD where some workers can be very slow, or even incapable, when executing certain tasks, but very efficient when developing some others.

#### A. The Assembly Line Worker Assignment and Balancing Problem

This real situation was the source of inspiration for defining the new problem called *Assembly Line Worker Assignment and Balancing Problem* (ALWABP), which in [14] is mathematically modeled and where different solving approaches are proposed.

This problem arises in those assembly lines where we have certain limited resources available (normally workers) in which the operation time for every task is different depending on who executes the task, and where there are also some task-worker incompatibilities defined. In its basic form, an assembly line consists of a finite set of work elements or single tasks, each having an operation processing time and a set of precedence relations, which specify the permissible orderings of the tasks. The fundamental assembly line balancing problem is to assign the tasks to an ordered sequence of stations, such that the precedence relations are satisfied and some measure of effectiveness is optimized [7]. However, the problem here not only consists of assigning tasks to stations, but also available workers to stations; always respecting the incompatibilities when assigning tasks to workers.

Some other references also face a similar double assignment of tasks and resources to stations. For example some cost-oriented models assume that the equipment of the stations is given and that the production process is fixed, then the total cost must be minimized by optimally integrating design (selecting the machine type to locate at each activated station) and operating issues (assigning tasks to observe precedence relationships and cycle time restrictions). When these decisions are connected, the term *Assembly Line Design Problem* [1] or *Assembly System Design Problem* [18] are frequently used in the literature.

Although ALWABP can be included in this kind of problems, it is not a cost problem in which there are alternative machines with different costs and the total cost has to be minimized. Furthermore, in ALWABP the available resources are constrained: there are unique workers that can only be assigned once. In some cases there exist workers with similar characteristics, but even in these cases there is not an infinite number of workers available, as assumed in most ASDP problems.

SWD are a prototypical environment for human diversity, but ALWABP is not exclusive of this environment. Results obtained in [14] are extensible to ordinary assembly lines with non disabled workers, since diversity and some other considered real features are also present.

#### B. Objectives and structure of the paper

A very interesting extension of ALWAB problem that can help to satisfy SWD requirements is the use of U-shaped layouts with crossover workers. This would enable more available combinations when the manager has to cope with assigning tasks to workers, since the U-type lines have great assignment flexibility and balancing efficiency. In this sense the aim of this paper is to propose an extension to ALWABP that will be named as UALWABP. This extension is mathematically modeled, previously establishing certain basic assumptions, and a Laser Search solving procedure is designed and tested against a set of pseudo-benchmark problems in order to check its efficiency.

The paper is organized as follows: first we review the related U-shape assembly line balancing approaches in the literature. Then, we model the UALWAB problem proposed, previously highlighting the basic assumptions for the problem. This model is inspired on Urban's model presented in [24], which is the only integer programming model in the literature for U-lines. After formulating a mathematical model for this new problem, a Laser Search Branch and Bound-based procedure is defined in section IV, and compared against the IP model in section V. The experimental study is carried out with a set of self made problems generated through a two-level three-factor full factorial development. This enables to obtain valid conclusions that are also reported. Finally an application to a real case of both, the IP model and the procedures described, is presented and some conclusions and further research lines are exposed.

## II. STATE OF THE ART OF THE U-LINES

The assembly line balancing problem has been studied extensively since the very first publication of [19]. Simple Assembly Line Balancing Problem (SALBP) [1] is known to be NP-hard [10], and SALBP is a special case of ALWABP where every task has a fixed duration. Therefore, ALWABP is also NP-hard. When we talk about SALBP the aim can be to minimize the number of stations given a desired cycle time (SALBP-1) or to minimize the cycle time given a fixed number of workstations (SALBP-2), and several techniques have been proposed for the solution of both problems with

different considerations of tasks times and configurations (see the reviews of [6], [7] or [22]).

Recently, U-type layouts have been utilized in many production lines in place of the traditional straight-line configuration due to the use of just-in-time production principles. In 1994, [13] presented a new problem derived from the traditional ALB problem where production lines are arranged as U-type lines instead of straight lines. The U-type assembly line is an attractive alternative for assembly production systems since operators become multiskilled by performing tasks located on different parts of assembly line [9]. Stations can be arranged so that during the same cycle two workpieces at different positions on the line can be handled. Hence, the difference to SALBP is that a station  $k$  can contain not only tasks whose predecessors are assigned to one of the stations 1,...,  $k$ , but also tasks whose predecessors will be finished until the product returns to station  $k$  for the second time[21].

We can define two problem versions of U line balancing problems regarding to SALBP: UALBP-1: Given the cycle time  $C$ , minimize the number of stations  $m$ ; and UALBP-2: Given the number of stations  $m$ , minimize the cycle time  $C$ .

After Miltenburg first approach, in 1997 [17] developed bounds and approximations for the cycle time when task times are random variables and the number of stations is given (stochastic UALBP-2). Later, [24] gives an integer programming formulation for UALBP-1 and solves problem instances with CPLEX, and [23] develop the procedure ULINO (U-Line Optimizer) and apply it to all versions of UALBP distinguished above.

[5] present a simulated annealing algorithm for the problem, and [25] present a chance-constrained, piecewise linear integer program for UALBP-1 with stochastic task times, which can be solved with CPLEX for small problems. [3] propose a hybrid heuristic composed of priority rule based procedures and an improvement step.

More recently, [9] present a shortest route formulation for the simple U line balancing problem (SULB) inspired in the shortest route model developed by [10] for the traditional SALBP. Lastly [8] propose a goal programming model based both on the Urban integer programming formulation [24] and the goal model of [4], resulting in the first multi-criteria decision making approach to the U-line.

### III. ADAPTATION OF URBAN'S INTEGER PROGRAMING FORMULATION TO UALWABP

As presented in the above state of the art, there is a small and growing literature on ULB problem.

Since the SULB problem was first modeled by [13], the only one integer programming model in the literature has been the Urban's model presented in [24]. This reference can be a good baseline for adapting the ALWABP mixed integer programming model presented in [14] to a U-shaped scenario.

The distinguishing characteristic of the ULB problem is that it must allow for the forward and backward assignment of tasks to workstations; for example, the first and the last task of an assembly line can be placed in the same workstation on a U-line, but not on a traditional line [8]. Urban accomplished this in [24] by establishing a "Phantom" network and appending it to the original precedence network. Then, starting in the middle of this extended network, assignments to the workstations can be made forward through the original network, backward through the phantom network, or simultaneously in both directions. Thus, before presenting the notation and the IP model created, certain basic assumptions must be stated in order to completely define our U-problem:

- 1) Tasks processing times and precedence relationships are known deterministically.
- 2) A single product is assembled on the line.
- 3) We define a U paced line where buffers are not considered.
- 4) There are certain workers available, where task processing time can be different depending on which one of the workers executes the task, including some incompatibilities in the task times matrix
- 5) Every worker is assigned to only one workstation, but crossover workstations are possible due to the U-shape of the line.
- 6) Every task is assigned to only one workstation, provided that the worker selected for that station is capable of performing the task.

With these basic assumptions, the IP model for this problem will be defined using this notation:

According to this notation we have the following MIP formulation:

TABLE I - NOTATION USED FOR UALWABP

$i, j$	Task
$h$	Worker
$s$	Workstation
$N$	Set of tasks
$H$	Set of available workers
$S$	Set of Workstations
$A$	Set of assignments a priori ( $i, h$ ) task-worker
$C$	Cycle Time
$m$	Number of worksations
$p_{hi}$	Processing time for task $i$ when worker $h$ executes it
$D_j$	Set of tasks immediately preceding task $j$ in the precedence network
$x_{shi}$	Binary variable equal to 1 only if task $i$ is assigned to worker $h$ in station $s$
$z_{shi}$	Binary variable equal to 1 only if task $i$ is assigned to worker $h$ in station $s$ in the phantom precedence network
$y_{sh}$	Binary variable equal to 1 only when worker $h$ is assigned to station $s$

$$\text{Min } z = C \quad (1)$$

subject to:

$$\sum_{h \in H} \sum_{s \in S} (x_{shi} + z_{shi}) = 1; \quad \forall i \in N \quad (2)$$

$$\sum_{s \in S} y_{sh} \leq 1; \quad \forall h \in H \quad (3)$$

$$\sum_{h \in H} y_{sh} \leq 1; \quad \forall s \in S \quad (4)$$

$$\sum_{h \in H} \sum_{s \in S} s \cdot x_{shi} \leq \sum_{h \in H} \sum_{s \in S} s \cdot x_{shj} \quad \forall i, j / i \in D_j \quad (5)$$

$$\sum_{h \in H} \sum_{s \in S} s \cdot z_{shi} \geq \sum_{h \in H} \sum_{s \in S} s \cdot z_{shj} \quad \forall i, j / i \in D_j \quad (6)$$

$$\sum_{i \in N} p_{hi} \cdot (x_{shi} + z_{shi}) \leq C; \quad \forall h \in H; \forall s \in S \quad (7)$$

$$\sum_{i \in N} (x_{shi} + z_{shi}) \leq M \cdot y_{sh}; \quad \forall h \in H; \forall s \in S \quad (8)$$

$$\sum_{s \in S} x_{shi} = 1; \quad \forall (i, h) \in A \quad (9)$$

with:

$$M > \sum_{h \in H} \sum_{i \in N} p_{hi}$$

As it is showed, it is necessary to define parallel variable  $x_{shi}$  and  $z_{shi}$  so that the assignments can be done forward in the precedence network or backwards in the phantom precedence network. Therefore we have that:

- Objective function (1) minimizes the cycle time.
- With constraints set (2) every task  $i$  is assigned

to a single station  $s$  and worker  $h$  on the real or in the phantom network.

- (3) and (4) ensure that every worker can be assigned to an only one station, and that in every station there is only one worker.
- Constraints (5) and (6) imply that the precedence constraints are not violated on the original network and phantom network.
- (7) and (8) ensure that every worker  $h$  assigned to station  $s$  can have more than one task, whenever given cycle time  $C$  is not overcome (including tasks assigned both from original and from phantom network). As Cycle Time  $C$  and  $y_{sh}$  are both variables, (7) and (8) are defined separately in order to maintain the model linearity.
- (9) model those situations where certain task-worker assignments (A) must be considered a priori by therapeutic or other specific reasons.

In analogy with UALBP-2, where the aim is to minimize the cycle time given a set of workstations, this problem is named as UALWABP-2, modeling the most typical situation in SWD: given certain unique workers, to minimize the cycle time.

In UALBP-1, the aim is to minimize the number of stations given a target cycle time, and then an UALWABP-1 problem can be easily formulated just by modifying the exposed model. As this situation is not so usual in this environment, for the sake of brevity, we prefer to focus this paper just in the UALWABP-2 problem, although the procedure presented in the next section has a modular design that enables the resolution of both problem types.

#### IV. LAU: LASER SEARCH DESIGNED FOR UALWABP

In the last decades, many Branch and Bound approaches have been proposed in the literature for solving different combinatorial problems, including assembly line balancing. The Branch and Bound here proposed is a Depth First Search procedure that works with a LASer Search strategy ([20], [22]) and has been named as **LAU**.

Most of SALBP-2 resolution procedures in the literature are search methods based on repeatedly solving instances by SALBP-1 procedures [22]. The same way, the procedure developed for solving UALWABP-2 is somehow based on a UALWABP-1 approach. In this sense, the procedure always explores the solution space trying to find the assignment with less number of stations. The first attempt is done with a starting cycle time  $C$ , and while  $C$  is unfeasible (when  $C$  is too low because more than the available workers are needed) it is iteratively increased by one. The first time that  $C$  is

possible, the assignment will normally include all workers available, and this will be the optimal solution for the UALWABP-2 with a minimum cycle time.

#### A. Starting bound for cycle time

In this sense, the starting cycle time for all the procedures is set to  $C = \max(C_1, C_2)$ , since cycle time should accomplish the following statements:

- A cycle time may never be less than the minimum task time, since tasks are indivisible. In this case, in the best assignment that could be achieved, every task would be assigned to the worker that performs it fastest. Then, being  $lowp_i$  the lowest processing time for every task  $i$ , we have:

$$C \geq C_1 = \max(lowp_i) \quad \forall i \in N \quad (10)$$

- On the other hand, if we relax the problem by ignoring the precedence constraints and by expecting a perfect assignment (in which every task is assigned to the worker with the lowest processing time) we can adapt the bound defined in [2] for SALBP, which is obtained by using an analogy with the Bin Packing Problem. In our case, this bound is:

$$C \geq C_2 = \left\lceil \frac{\sum_{i \in N} lowp_i}{|H|} \right\rceil \quad (11)$$

#### B. Branching scheme of LAU

Regarding the branching, with the actual cycle time, there are two different situations to manage throughout the process of creating the solution tree:

- Starting in node 0, when a new station is opened, one node for every combination of workers and task available is created. Excluding the incompatibilities, the feasible tasks will be: on one hand the tasks that have no predecessors (or its predecessors have been already assigned) developed by the workers still not assigned; on the other hand the tasks in the phantom network that have no successors (or they have been already assigned) developed by the still available workers.
- Once inside a station, only feasible tasks for the worker actually assigned to that station are selected, and one node is created for each one of these tasks (with the same double consideration of precedence depending on if we are in the phantom network or not).

Therefore, according to a Laser Search strategy, in every iteration only one descending node is built and

developed at a time until:

- a leaf node is reached
- no leaf node is reached, but the all workers are already assigned.
- the current node is fathomed.

In the three cases on its way back to the root, the search follows the first possible alternative branch, but each node is completely developed before its ancestor nodes are revisited [20].

If all the possible paths have been explored and no solution is provided, this means that the tried cycle time is not feasible with the available workers, and must be increased by one; resetting then the algorithm and starting again. The procedure iterates this way making all necessary returns to ancestor nodes in order to guarantee an optimal solution.

## V. EXPERIMENTAL STUDY

#### A. Experimental study for the procedures developed

Since UALWABP is a new problem, there is no standard set of benchmark problems available for testing. So we have constructed a two-level three factors full factorial experimental study based on the *Mitchell* problem, from the classical collection of SALBP problems of [11]. From this standard problem the original precedence network was preserved. The original task time was used for first worker and new workers tasks times were randomly generated from first worker ones. From our experience in SWD involved in our R&D project, the range for randomly generating these times should not be greater than three times the original task time. When a worker is over this range for a certain task, we will assume that this task shouldn't be assigned to him/her. The time for that task when developed by this worker will be then infinite (which means: task not assignable to this worker). In fact, different percentages of incompatibilities in the tasks-workers matrix were also defined for this full factorial study. The problems were generated according to the following three parameters:

- **NrW:** The relation between the number of tasks and the number of workers (size of the matrix).
- **Var:** Variability of task times for the different workers.
- **Inc:** The percentage incompatibilities defined a priori in the task-worker matrix.

For the first factor two levels, *high* (number of tasks 3 times higher than number of workers) and *low* (number of tasks 6 times higher than number of workers), were defined. The different tasks times for

every task  $i$  were randomly generated from a uniform distribution with range selected according to the original time  $t_i$ . The two levels defined for the task times variability used the distributions  $U[1, t_i]$  and  $U[1, 3t_i]$  for *low* and *high* variability. And finally, the *low* and *high* percentage of incompatibilities in the tasks-workers matrix was set to 10% and 20% approximately.

40 problems were generated following this outline in order to compare the behaviour of LAU procedure against the Integer Programming approach when facing different kinds of problem. This structured way of experimenting let us obtain valid partial and global conclusions that will be exposed. The IP model of UALWABP-2 was translated into MPL language so that every problem was run with CPLEX 9.0, with the following results.

### B. Results

In this section we summarize the relative behavior of both methods by comparing the mean values of the computational time (in seconds) needed by IP and by Laser Search procedure (**IP** and **LAU** in next figures). In the table II we observe the ANOVA (analysis of variance) obtained and how the main significant factor is the number of workers.

TABLE II – ANOVA SUMMARY OF RESULTS

Analysis of Variance for Comp Time - Type III Sums of Squares

Source	Sum of Squares	Df	Mean Square	F-Ratio	P-Value
<b>MAIN EFFECTS</b>					
A:Method	3,7698E7	1	3,7698E7	48,09	0,0000
B:NrW	1,15294E7	1	1,15294E7	14,71	0,0003
C:Var	2,18416E6	1	2,18416E6	2,79	0,0996
D:Inc	7,0758E6	1	7,0758E6	9,03	0,0037
<b>INTERACTIONS</b>					
AB	5,26851E7	1	5,26851E7	67,20	0,0000
AC	1,48766E6	1	1,48766E6	1,90	0,1728
AD	1,38049E6	1	1,38049E6	1,76	0,1889
BC	259210,0	1	259210,0	0,33	0,5672
BD	107531,0	1	107531,0	0,14	0,7123
CD	3,26555E6	1	3,26555E6	4,17	0,0451
<b>RESIDUAL</b>	5,40933E7	69	783961,0		
<b>TOTAL (CORRECTED)</b>	1,98519E8	80			

As it is shown in the next figure an overall better behaviour of LAU is achieved, with important differences in some of the instances. The next figure indicates this difference between both methods:

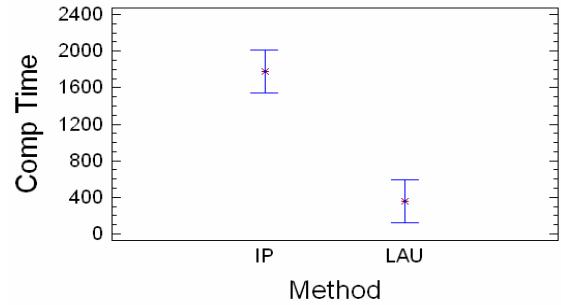


Fig.1. Means and 95.0 % LSD Intervals Graphic for IP model and LAU

About the double interactions we only notice one significant interaction between the **Method** applied (A in the ANOVA table) and the number of workers **NrW** (B in the ANOVA table). It has been noticed a quite important difference when facing problems with more or with less workers available (larger or smaller problems). For smaller problems (**Low** number of workers) the **IP** model has better behaviour; however **LAU** is much faster for finding an optimal solution when problems are bigger:

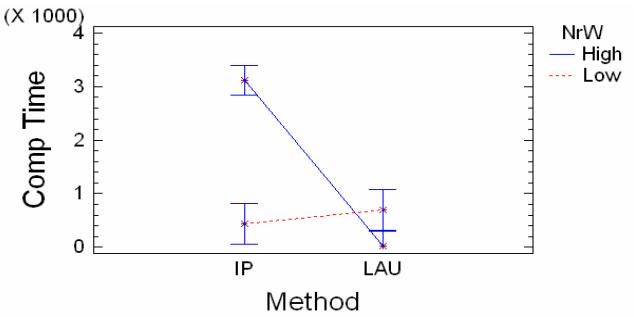


Fig.2. Interaction Method vs. NrW 95.0 % LSD Intervals

IP model is more affected by upper values of **NrW** probably because it needs to generate many more constraints as the task-times matrix raise. Apart from this, no more significant interactions have been founded between the **Method** applied and the variability of task times for the different workers (**Var**), or the percentage of task-worker incompatibilities defined a priori (**Inc**).

### VI. APPLICATION TO A REAL CASE

An application to a real case of both the IP model and the procedure described is now presented in order to clarify the potential real benefits of UALWABP. With this aim we used the data of the ALWABP case study presented in [15]. In that reference some resolution proposals are successfully applied in a SWD whose main industrial activities are related to assembling of electronic components. In the section of the SWD where this research was

carried out, a prototype assembly line for the product with highest demand was designed and successfully implemented.

Seven workers were selected for this reengineering process. After a long process of task times definition, the final data file with the operation times for the 18 tasks and seven workers (**h1** to **h7**) implied was determined. This task-workers matrix with the existent incompatibilities (**Inf**) is shown:

The IP model for ALWABP was fed with this

TABLE III - TASK TIMES MATRIX DEFINED

Workers \ Tasks	<b>h1</b>	<b>h2</b>	<b>h3</b>	<b>h4</b>	<b>h5</b>	<b>h6</b>	<b>h7</b>
1	31	31	31	26	26	31	31
2	84	84	80	53	51	71	80
3	115	115	146	73	66	113	110
4	115	115	141	94	73	110	110
5	80	80	84	74	65	75	80
6	119	114	141	98	86	139	119
7	84	84	80	61	73	75	80
8	93	81	124	90	87	94	93
9	26	27	53	44	30	35	35
10	31	44	Inf	42	38	56	44
11	39	42	Inf	42	39	55	44
12	15	14	Inf	15	15	16	15
13	70	90	Inf	80	59	88	80
14	81	101	Inf	92	71	81	66
15	97	97	Inf	98	60	97	63
16	106	106	168	82	84	98	85
17	Inf	Inf	181	Inf	Inf	124	Inf
18	Inf	Inf	26	Inf	Inf	26	Inf

data and run with CPLEX 9.0 and the solution provided, in a traditional straight-line approach, is the one showed in figure 3. In this solution we can see how, from the seven workers implied, the worker **h1** placed in the fifth station is the bottleneck and the sum of his tasks (task 9, 13 and 14) is  $177 \text{ min}^{-3}$ :

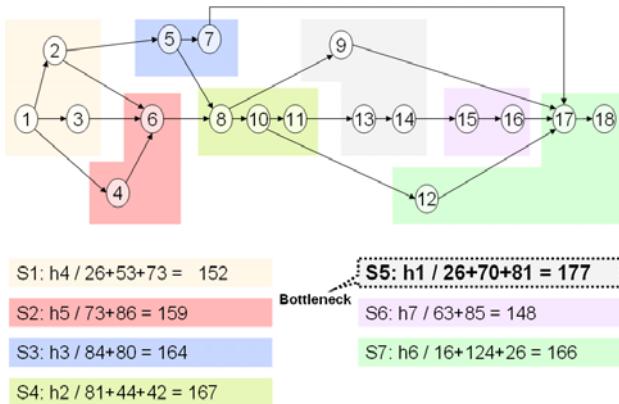


Fig.3. ALWABP-2 case resolution

The same data file was used in this research in order to validate the U-line concepts presented in this paper as an extension of the problem. Both the UALWABP IP modeling approach and the Laser Search approach gave the same final solution which

showed a better cycle time performance:

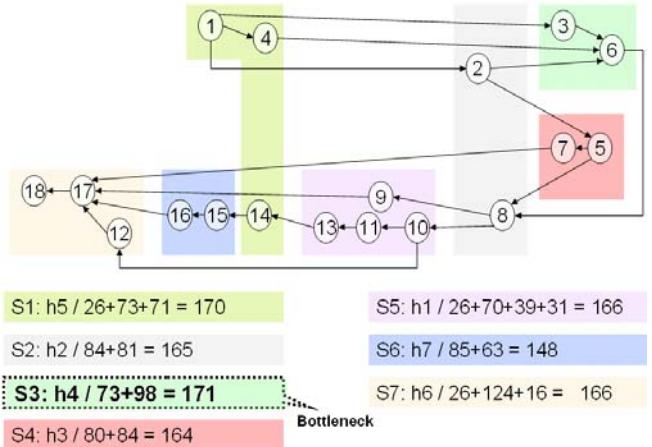


Fig.4. UALWABP-2 case resolution

As can be observed, the bottleneck is now the worker **h4** and the sum of his tasks (task 3 and 6) is  $171 \text{ min}^{-3}$ . In this solution, two workers (h5 and h2) are placed in crossover workstations, which enables to achieve a better cycle time and more combinations of feasible assignments for job rotation procedures (which is a further research line that we are already actually exploring).

In this case the IP model run with CPLEX 9.0 needed 47397 seconds to solve the problem, while LAU algorithm needed only 261 seconds; both in a Pentium IV 1GHz (and also the previous tests). This confirms some of the conclusions obtained in the previous section: for big problems, the LAU procedure achieves much better results.

## VII. CONCLUSIONS

The assembly lines are very useful in SWD since the division of work in single tasks can make some disabilities disappear, just by finding a proper assignment of tasks to workers and workers to stations. Although the ALWAB problem models the main features of the assembly lines in this environment, new approaches are necessary to increase the feasible assignments.

In this paper an extension of this problem, named as UALWABP, has been presented and modeled. This model proposes a U-shaped solution where crossovers stations are allowed and some specific constraints must be considered. LAU, a Branch and Bound procedure, that uses Laser Search as strategy, has been presented and tested against the IP model through an experimental study. The main conclusion of this study has been the better performance of LAU procedure when the number of workers, and then the possible combinations, increase.

A sample of application of these procedures with real data of a SWD, whose main industrial activities are related to assembling of electronic components, has been useful for validating the proposal.

Further research includes two main topics that would widen the scope of the problem. The first one is considering parallel workstations, which would enable even more combinations of assignments. Both modelling and design of solving procedures should be faced in this new scenario. The other main research line will be the design of efficient job rotation procedures, taking advantage of the new combinations that U-lines and parallel stations would provide. Job rotation is always desirable but here is even more important, since it can contribute to the improvement, if not simply fundamental maintenance, of certain worker abilities. As the task times are different depending on the worker, these procedures are not as obvious as in ordinary assembly lines.

SWD should be the first who benefit from these research lines. The continuous improvement of resolution methods for this environment is very important: although SWD receive some institutional help, these centres have to survive in real markets and, therefore, then need to run efficiently considering also the limitations of the individuals that work there. Only through efficiency can reach their primary aim: to grow in order to provide more jobs for more disabled people which is, in fact, the final aim of this research.

#### ACKNOWLEDGEMENTS

This work was developed under research projects DELIMER (DPI2004-03472) and GESCOFLOW (DPI2004-02598) both supported by the Spanish National Science&Technology Commission CICYT. The authors would also like to acknowledge the SWD involved in this research for their collaboration.

#### REFERENCES

- [1] Baybars, I. (1986), "A survey of exact algorithms for the simple assembly line balancing problem", *Management Science* 32, 909-932.
- [2] Bellamy, G.T., Horner R.H., Inman D.P., (1979), "Vocational Habilitation of Severely Retarded Adults: A direct Service Technology", University Press, Baltimore
- [3] Chiang, W.-C., Urban, T.L., (2002). A hybrid heuristic for the stochastic U-line balancing problem, Working Paper, University of Tulsa, Oklahoma, USA.
- [4] Deckro, Rangachari, (1990), "A goal approach to assembly line balancing", *Computers and Operations Research* 17, 509-521.
- [5] Erel, E., Sabuncuoglu, I., Aksu, B.A. (2001), "Balancing of U-type assembly systems using simulated annealing", *International Journal of Production Research* 39, 3003-3015.
- [6] Erel, E., Sarin S. (1998) "A survey of the assembly line balancing procedures". *Production Planning and Control* 9, No 5, 414-434.
- [7] Ghosh, S., Gagnon R.J. (1989), "A comprehensive Literature Review and Analysis of the Design, Balancing and Scheduling of Assembly Systems", *International Journal of Production Research* 27, 637-670
- [8] Gökçen, H., Kürsad A. (2004), "A goal Programming approach to simple U-line balancing problem". Accepted in *European Journal of Operational Research*, to be published.
- [9] Gökçen, H., Kürsat A., Gencer C., Kizilkaya, E. (2004), "A shortest route formulation of simple U-type assembly line balancing problem". Accepted in *Applied Mathematical Modelling*, to be published.
- [10] Gutjahr, A.L., Nemhauser G.L. (1964), "An algorithm for the line balancing problem", *Management Science* 11 (2) (1964) 308-315.
- [11] Hoffmann, T.R. (1990), "Assembly line balancing: A set of challenging problems", *International Journal of Production Research* 28, 1807-1815.
- [12] Marin, J. A., M. Pardo and T. Bonavia (2002), "The Impact of Training and Ad Hoc Teams in Industrial Settings", 6th International Workshop on Teamworking (IWOT 6). Malmö (Sweden).
- [13] Miltenburg, J., Wijngaard, J. (1994), "The U-line balancing problem", *Management Science* 40, 1378-1388.
- [14] Miralles C., García-Sabater, J.P., Andrés C., Cardós M. (2005): "Branch and Bound Procedures for solving the Assembly Line Worker Assignment and Balancing Problem. Application to Sheltered Work Centres for Disabled". Accepted in *Discrete Applied Mathematics*.
- [15] Miralles C., García-Sabater, J.P., Andrés C., Cardós M. (2005), "Advantages of assembly lines in sheltered work centres for Disabled. A Case Study", Proceedings of the International Conference on Production Research ICPR 2005 in Salerno (Italy).
- [16] Miralles C., Andrés C., García-Sabater, J.P. (2005), "Model and heuristic approach for the Assembly Line Worker Assignment and Balancing Problem. Application to Sheltered Work Centres for Disabled" International Conference on Industrial Engineering and Systems Management IESM May 2005, Marrakech (Morocco).
- [17] Nakade, K., Ohno, K., Shanthikumar, J.G. (1997), "Bounds and approximations for cycle times of a U-shaped production line", *Operations Research Letters* 21, 191-200.
- [18] Pinnoi, A., Wilhelm W.E. (1997), "A family of hierarchical models for assembly system design", *International Journal of Production Research* 35, 253-280
- [19] Salveson, M.E. (1955): The assembly line balancing problem. *Journal of Industrial Engineering* 6, No 3, 18-25
- [20] Scholl, A. (1999), "Balancing and sequencing assembly lines", 2nd edition, Physica, Heidelberg
- [21] Becker, C., Scholl, A. (2003) "A survey on problems and methods in generalized assembly line balancing", *European Journal of Operational Research* (to be published, available online).
- [22] Scholl, A. and Becker, C. (2004) "State-of-the-art exact and heuristic solution procedures for simple assembly line balancing". *European Journal of Operational Research* (to be published, available online).
- [23] Scholl, A., Klein, R. (1999), "ULINO: Optimally balancing U-shaped JIT assembly lines", *International Journal of Production Research* 37, 721-736.
- [24] Urban (1998), "Optimal balancing of U-shaped assembly lines", *Management Science* 44(5) 738-741
- [25] Urban, T.L., Chiang, W.-C. (2002), "Piecewise-linear optimization of the U-line balancing problem with stochastic task times" *European Journal of Operational Research*.

# Modelling and Forecasting Spanish Mortality

Ana Debón Aucejo\* and Francisco Puig Blanco†

\*Universidad Politécnica de Valencia /Dept. de Estadística e I. O. Aplicadas y Calidad.  
Camino de Vera, s/n. 46022-Valencia (Spain)

Email: andeau@eio.upv.es

†Universitat de València/Dept. de Direcció d'Empreses. "Juan J. Renau Piquerias"  
Av. Tarongers, s/n. 46022-Valencia (Spain)  
Email: francisco.puig@uv.es

**Abstract**—Experience shows that standard life tables overestimate death probabilities. This fact has a negative consequence for insurance companies because premiums are lower than they actually should be otherwise. The reason for this overestimation is that standard life tables, as they are computed for a specific period of time, cannot take into account the decreasing mortality trend over time. Dynamic life tables overcome this problem by incorporating the influence of the calendar when graduating mortality. Recent papers on the topic look for the development of new methods to deal with this dynamism.

The majority of methods used in dynamic tables are parametric, these methods apply traditional mortality laws and then analyze the evolution of estimated parameters with time series techniques.

Our contribution consists in applying Lee-Carter methods to Spanish mortality data, exploring residuals and future trends.

**Keywords**—Standard Life Tables, Dynamic Life Tables, Lee-Carter, Forecasting.

## I. INTRODUCTION

THE most recent actuarial literature recognises the fact that mortality evolves over time. Mortality experiences that correspond to different periods display different death probabilities for the same age. This is supported by the fact that mortality has been seen to gradually decline over time, although the decrease is not necessarily uniform across age groups. Therefore, it is important to be able to measure mortality changes over time accurately, as life insurance policy gains depend on survival. If the standard table used to calculate annuities and reserves predicts higher mortality probabilities than is actually the case among policy-holders, the latter will have been undercharged and the insurance company will make a loss.

The concept of a dynamic table seeks to solve this problem by jointly analyzing mortality data corresponding to a series of consecutive years. This approach allows the calendar effect's influence on mortality to be studied.

A compilation of a sample of dynamic models and their classification can be found in [1], [10] and [13].

We aim to use Lee-Carter model to obtain Spanish dynamic tables. By determining this structure, we will be able to make predictions by applying the model.

The current time-dependent mortality measurements are, probability of death,  $q_{xt}$ , the force of mortality,  $\mu_{xt}$ , and the central mortality rates,  $m_{xt}$ , all described as being at age  $x$  and year  $t$ .

The following article is structured as follows: Section 2 briefly presents the Lee-Carter's methodology to be used. Section 3 presents the results of the application of the method to the analysis of Spanish mortality data, corresponding to the period 1980-1999 and also the results of the prediction of specific mortality rates in the year 2000 ( $q_{x,2000}$ ), obtained by means of the adjusted models.

Section 4 contains the conclusions drawn from the results presented in the previous section.

## II. ADJUSTMENT AND PREDICTION OF $q_{xt}$

The Lee-Carter Model, developed in [7], consists in adjusting the following function to the central mortality rates,

$$m_{xt} = \exp(a_x + b_x k_t) + \epsilon_{xt}$$

or, equally, the function

$$\ln(m_{xt}) = a_x + b_x k_t + \epsilon'_{xt}, \quad (1)$$

applied to its logarithm matrix. In the previous two expressions, the double subscript refers to the age,  $x$ , and to the year or unit of time,  $t$ ,  $a_x$  and  $b_x$  are age-dependent parameters and  $k_t$  is a specific mortality index for each year or unit of time. The errors  $\epsilon_{xt}$ , with a zero average and variance  $\sigma_\epsilon^2$ , reflect the historical influences of each specific age that are not captured by the model.

In [6] the author remarks that nothing ensures that equation (1) will not exceed unity, while this possibility could be avoided by modelling the logit death rates. We

are going to apply this model to logit death probability  $q_{xt}$ ,

$$\ln \left( \frac{q_{xt}}{1 - q_{xt}} \right) = a_x + b_x k_t + \epsilon'_{xt}, \quad (2)$$

or, equally, the function,

$$q_{xt} = \frac{\exp(a_x + b_x k_t)}{1 + \exp(a_x + b_x k_t)} + \epsilon_{xt}$$

#### A. Estimation

In order for the model to have only one solution, parameters must be normalised,  $\sum_x b_x = 1$  and  $\sum_t k_t = 0$ .

Given that the structure is invariant under either of the parameter transformations,  $(a_x, b_x/c, ck_t)$  or  $(a_x + cb_x, b_x, k_t - c)$ , for any constant  $c$ .

##### 1) Lee-Carter:

- a) The estimate  $a_x$  as,

$$\hat{a}_x^1 = \frac{\sum_t \ln \left( \frac{q_{xt}}{1 - q_{xt}} \right)}{T},$$

with  $T$  number of years.

- b) The values  $b_x$  and  $k_t$  are estimated by singular value decomposition (SVD) applied to the matrix,  $\ln \left( \frac{q_{xt}}{1 - q_{xt}} \right) - \hat{a}_x^1$ . Then  $\hat{k}_t^{SVD}$  and  $\hat{b}_x^{SVD}$  are the respective first right and first left singular vectors.

The solution with  $\sum_x \hat{b}_x = 0$  is  $\hat{b}_x = \frac{\hat{b}_x^{SVD}}{\sum_x \hat{b}_x^{SVD}}$  then  $\hat{k}_t^1 = \hat{k}_t^{SVD} \sum_x \hat{b}_x^{SVD}$ .

- c) Later,  $k_t$  is reestimated to fit the total number of deaths observed and estimated, by means of the equation

$$D_t = \sum_x \left( N_{xt} \frac{\exp(\hat{a}_x^1 + k_t \hat{b}_x)}{1 + \exp(\hat{a}_x^1 + k_t \hat{b}_x)} \right), \quad (3)$$

where  $D_{xt}$ ,  $N_{xt}$  are number of death and the population at age  $x$  in year  $t$  and  $D_t = \sum_x D_{xt}$  is total deaths in year  $t$ . After applying equation 3, we obtain  $\hat{k}_t^2$  and transform  $\hat{k}_t^2 - \text{mean}(\hat{k}_t^2) = \hat{k}_t$  and  $\hat{a}_x^1 + \hat{b}_x \text{mean}(\hat{k}_t^2) = \hat{a}_x$ .

Finally, the solution is  $(\hat{a}_x, \hat{b}_x, \hat{k}_t)$  that we could improve by conditional generalized linear models (GLM), as [3] applied them to force of mortality.

2) Lee-Carter-GLM: In this paper we will use GLM applied to probability of death,  $q_{xt}$ , because the number of deaths  $D_{xt} \sim Bi(E_{xt}, q_{xt})$ , where  $E_{xt}$  are initial exposed to risk,

- d) for each  $x$  with

$$\log \left( \frac{q_{xt}}{1 - q_{xt}} \right) = \text{offset}(\hat{a}_x) + b_x \hat{k}_t,$$

we obtain  $\hat{b}_x^{GLM(1)}$  with  $\sum_x \hat{b}_x^{GLM(1)} = S \neq 1$ ,

- e) then for each  $t$  with

$$\log \left( \frac{q_{xt}}{1 - q_{xt}} \right) = \text{offset}(\hat{a}_x) + \hat{b}_x^{GLM(1)} k_t,$$

we obtain  $\hat{k}_t^{GLM(1)}$  with  $\sum_x \hat{k}_t^{GLM(1)} \neq 0$ ,

- f)

$$\begin{aligned} \hat{b}_x^{GLM} &= \frac{\hat{b}_x^{GLM(1)}}{S} \\ \hat{k}_t^{GLM} &= \hat{k}_t^{GLM(1)} S - \text{mean}(\hat{k}_t^{GLM(1)} S) \\ \hat{a}_x^{GLM} &= \hat{a}_x + \hat{b}_x^{GLM} \text{mean}(\hat{k}_t^{GLM(1)} S) \end{aligned}$$

Now, the solution is  $(\hat{a}_x^{GLM}, \hat{b}_x^{GLM}, \hat{k}_t^{GLM})$  which satisfies the constraints.

#### B. Residuals

Little attention appears to have been paid to the definition and analysis of residuals in the literature; [11] is an exception. This work focuses on SVD residuals,

$$\hat{\epsilon}'_{xt} = \ln \left( \frac{q_{xt}}{1 - q_{xt}} \right) - (\hat{a}_x + \hat{k}_t \hat{b}_x)$$

standardized by dividing by

$$\sqrt{\frac{\sum \hat{\epsilon}'_{xt}^2}{\nu}},$$

where  $\nu = (n - 1)(T - 2)$  are degrees of freedom with  $n$  number of ages.

As an alternative, [12] used Pearson residuals in the context of Poisson distribution associated with  $\mu_{xt}$ . In this context, we could use these residuals for  $q_{xt}$  with Binomial distribution,

$$z_{xt} = \frac{D_{xt} - N_{xt} q_{xt}}{\sqrt{N_{xt} q_{xt} (1 - q_{xt})}}.$$

In this paper, we study SVD residuals with the usual graphs. We add a Mean Absolute Percentage Error (MAPE) to analyse the goodness-of-fit for each of the years, criteria used by [5].

#### C. Prediction

Once the series of values  $\hat{k}_t$  have been estimated, the last step of the Lee-Carter model consists in finding a model for the series using Box-Jenkins methodology. In many of these applications, a good model for the  $k_t$  is

$$\hat{k}_t = c + \hat{k}_{t-1} + u_t.$$

where  $c$  is constant and  $u_t$  is white noise. With this model, the prediction of  $k_t$  varies in a linear way and

each death rate predicted varies at a constant exponential rate.

Some authors have made suggestions and modified this method, including [2], [14] and [6] himself, who compares his method to other alternatives, such as that of [8], [9].

### III. ANALYSIS OF MORTALITY DATA IN SPAIN

#### A. Data

The models described in Section 3 have been used to adjust mortality data for men and women separately. The data have been collected in Spain in the period 1980–1999 and correspond to the range of ages from 0 to 96. The population of the country studied was 40,847,371 in the last census carried out in 2001.

The crude estimates of  $q_{xt}$  that the models require, have been obtained by means of the procedure used by the Instituto Nacional de Estadística (Spanish National Institute of Statistics),

$$\dot{q}_{xt} = \frac{1/2(D_{xt} + D_{x(t+1)})}{P_{xt} + 1/2D_{xt}},$$

where  $D_{xt}$  are the deaths in year  $t$  at age  $x$ ,  $D_{x(t+1)}$  are the deaths in year  $t+1$  at age  $x$ , and  $P_{xt}$  is the population that on December 31 of year  $t$  were aged  $x$ . The formula can be applied to all ages, except for zero, due to the concentration of deaths in the first few months of life. This expression has been used for this age,

$$\dot{q}_{0t} = \frac{0.85D_{0t} + 0.15D_{0(t+1)}}{P_{0t} + 0.85D_{0t}}.$$

#### B. Model Adjustment

The high number of parameters estimated in this model,  $97 \times 2 + 20 = 194$  for men and a similar number for women, cannot be fully presented in a paper of this extent. Instead, we preferred to present them in the form of a graph in Figure 1. By comparing parameter  $a_x$  for both sexes, it is verified that mortality among women is lower than among men. Parameter  $b_x$  displays negative values for ages from 24 to 40 for men, for ages from 28 to 32 for women and for both sexes at advanced ages (more than 92,93), which indicates that mortality in these age groups does increase over time.

The differences between Lee-Carter and Lee-Carter-GLM are very small in  $a_x$  values for both sexes. In the case of  $b_x$ , the differences between Lee-Carter and Lee-Carter-GLM are greater among men than among women.

Estimate values for  $k_t$  are displayed in Figure 1(c). The graph clearly shows that the decrease in mortality for

women is more marked. The differences between Lee-Carter and Lee-Carter-GLM are greater among men than among women.

[5] use the Heligman-Pollards laws to research the way in which the calendar time affects mortality patterns in the Spanish population, and how this information can be used to elaborate predictions. They remark that the evolution has differed by gender and age ranges. Men's main ages are as follows,

- i) Ages from 0 to 10: clear decreasing trend.
- ii) Ages from 11 to 40: the accident hump has developed throughout the period to include these ages.
- iii) Ages from 41 to 90: clear decreasing trend with maximum differences around age 70, minimum at the range extremes.

By contrast, women's age ranges are as follows,

- i) Ages from 0 to 15: clear decreasing trend.
- ii) Ages from 16 to 35: no trend, it is the accident hump effect.
- iii) Ages from 36 to 90: again clear decreasing trend, reaching their maximum differences at age 75, minimum at the range extremes.

We confirm this evolution in Figures 2 and 3, which is similar in both models, and then, we explore the behavior for ages greater than 90 years.

In recent times, mortality statistics have improved the study of mortality shape at very old ages. [10] remarks that the mortality at very old ages is slowly increasing; Figure 4 shows this fact for the Spanish mortality.

#### C. Goodness-of-fit

Residuals plots in Figures 5 and 6. The plots show unsatisfactory behavior of residuals with different variances and the mean value which could be not zero.

A more detailed examination of the residuals shows autocorrelations in each year for both models (Figure 7). Nearly all the residuals are autocorrelated and show a definitive pattern.

In order to measure the goodness-of-fit of each year for both models and compare them with [5] results, we have included Table I, which shows MAPE. Analyzing the MAPE, we can conclude that the model Lee-Carter-GLM improves the goodness-of-fit measures more in the case of men than women.

#### D. Forecasting

As far as the modelling of  $k_t$  by means of temporal series is concerned, the series of first differences is seen

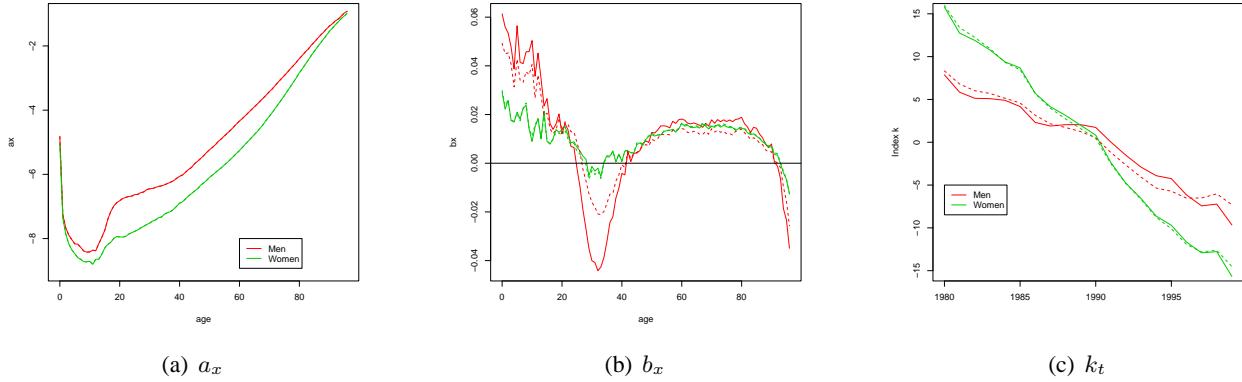


Fig. 1. Estimated values according to Lee-Carter (solid line) and Lee-Carter-GLM (dotted line).

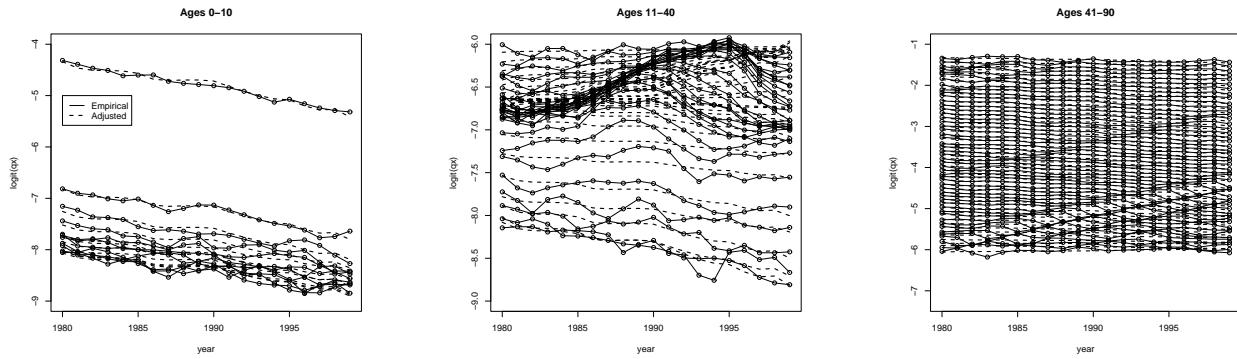


Fig. 2. Logit rates versus year (1980-1999) for men by age.

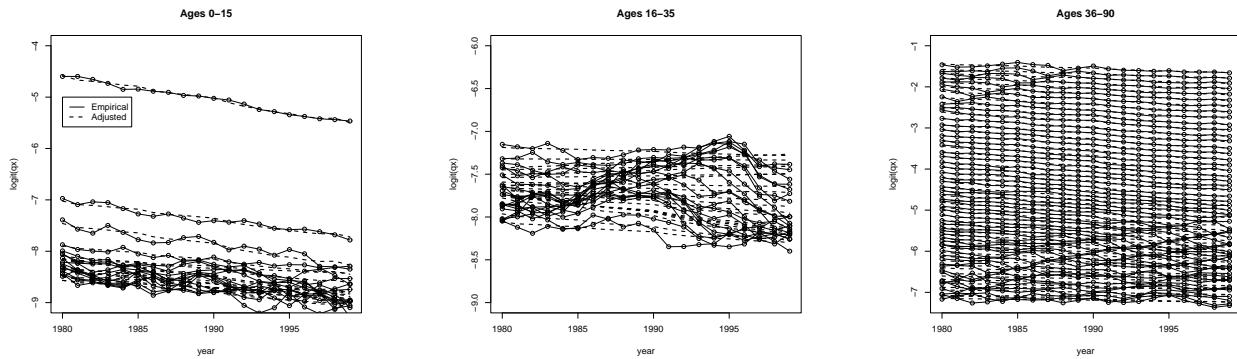


Fig. 3. Logit rates versus year (1980-1999) for women by age.

to be stationary serie for both sexes. The adjusted models are

$k_t = -0,922584 + k_{t-1} + u_t$ , and  $k_t = -1,65504 + k_{t-1} + u_t$  for men and women respectively in the case of Lee-Carter model, where  $u_t$  is white noise.

The adjusted models are

$k_t = -0,822322 + k_{t-1} + u_t$ , and  $k_t = -1,60528 + k_{t-1} + u_t$

for men and women respectively in the case of Lee-Carter-GLM model, where  $u_t$  is white noise.

The forecasted values of  $k_t$  are shown in Figures 8, 9, and in Table II for the period 2000-2011 for men and women.

#### E. Predictions for 2000

We focus on the projected rates of death in order to measure goodness of model to make predictions. The

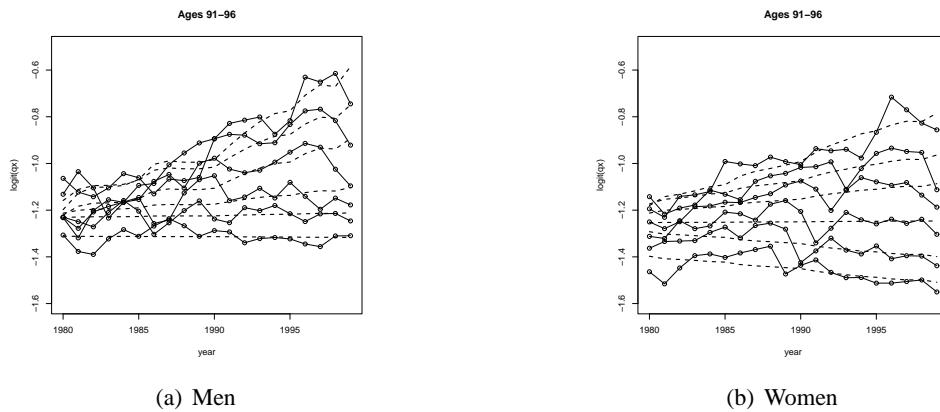


Fig. 4. Logit rates versus year (1980-1999) by old age.

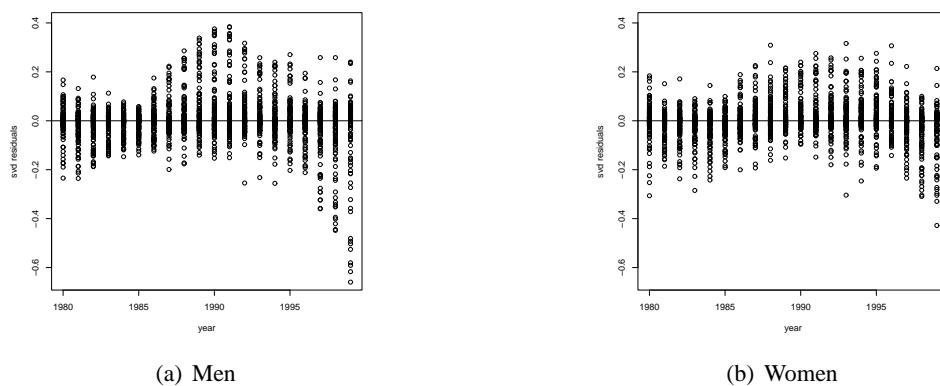


Fig. 5. Residuals SVD versus year

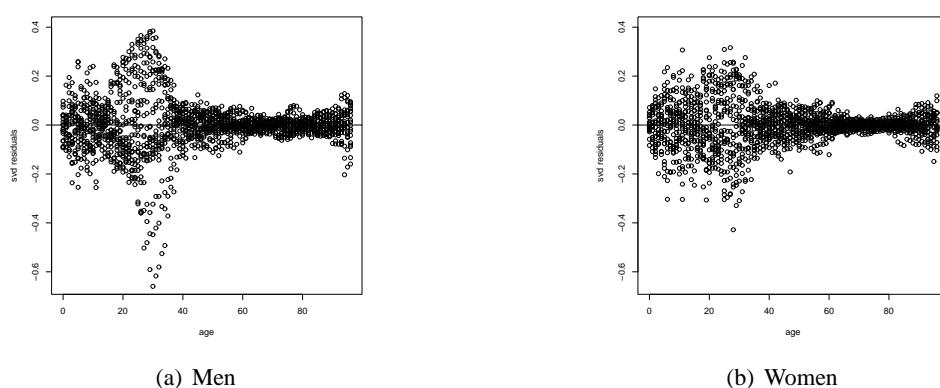


Fig. 6. Residuals SVD versus age

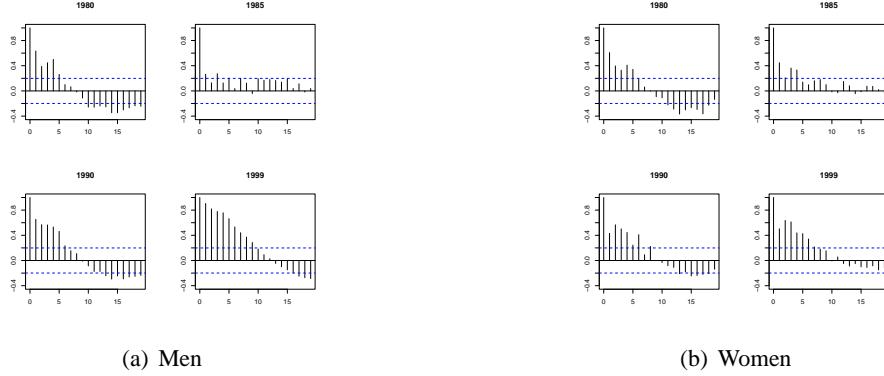
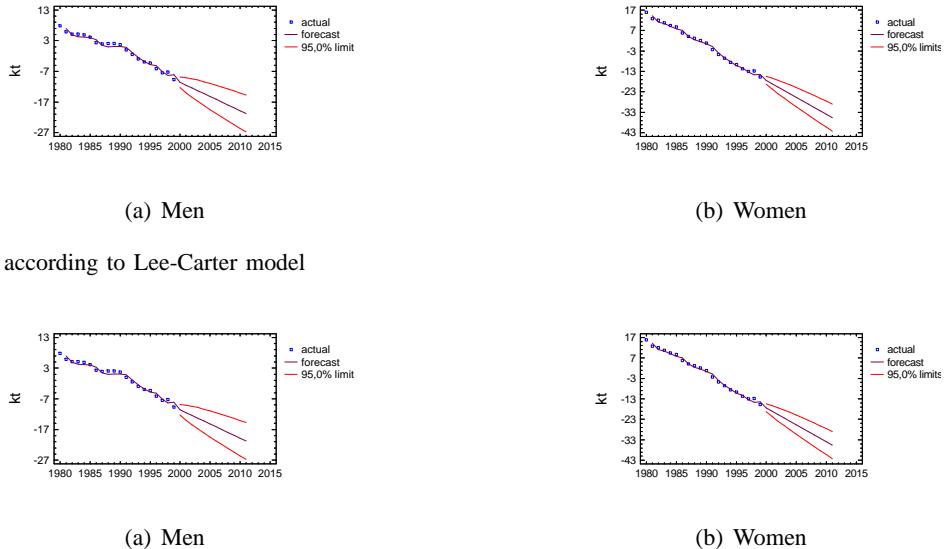


Fig. 7. Autocorrelations of residuals SVD versus year

Fig. 8. Forecast of  $k_t$  according to Lee-Carter modelFig. 9. Forecast of  $k_t$  according to Lee-Carter-GLM model

predictions of  $q_{x,2000}$  corresponding to 2000 have been calculated. The crude estimates are available to us as we also have mortality data for the year 2001.

The  $MSE$  and  $MAPE$  are a numerical summary of the accuracy of the predictions, which are displayed in Table III.

#### IV. CONCLUSIONS

In Table III,  $MAPE$  and the  $MSE$  show that the predictions made by both models in the case of women are better than men.

The Heligman and Pollard second law was used by [5] but with a maximum age of 90 and for the period of 1975 to 1993. By contrast, the maximum age in this analysis is 96 and the period of 1980 to 1999. In this respect, it is interesting to compare our with those obtained by the authors mentioned. They achieve a slightly better fitting

for men, but the fitting for women presents worse results than ours.

As far as the Lee-Carter-GLM method is concerned, the decrease in  $MSE$  is 40.05% for men and 6.05% for women. These percentages are 9.87% and -6.24% respectively, when compared  $MAPE$ .

This work pays attention to SVD residuals, variance and autocorrelations. The modelling of residuals by means of spatio-temporal methods, such as those developed by [4] would supposedly better capture the momentary progress of the phenomenon and could improve the results obtained up to now.

#### REFERENCES

- [1] B. Benjamin and J. Pollard, *The Analysis of Mortality and Other Actuarial Statistics*, 6th ed. London: Butterworth-Heinemann, 1992.

Year	Lee-Carter		Lee-Carter-GLM	
	Men	Women	Men	Women
1980	4.37	5.26	6.14	5.06
1981	5.03	4.07	7.27	3.87
1982	5.27	4.99	7.15	4.73
1983	3.98	4.74	5.66	4.43
1984	6.69	5.25	5.14	5.00
1985	3.12	4.36	4.35	4.17
1986	4.39	4.47	4.63	4.43
1987	4.78	4.82	4.76	4.85
1988	6.13	4.86	5.40	4.90
1989	7.59	5.10	6.65	5.12
1990	8.20	5.00	7.25	4.99
1991	7.61	5.04	7.13	4.98
1992	5.71	5.30	5.88	5.20
1993	5.52	5.51	6.09	5.39
1994	5.66	5.78	6.23	5.77
1995	4.95	6.08	6.19	6.03
1996	5.68	4.75	5.54	4.77
1997	8.21	5.32	6.66	5.74
1998	10.38	7.12	8.62	7.67
1999	13.26	7.69	9.50	8.24

TABLE I

MEAN ABSOLUTE PERCENTAGE ERROR FOR EACH YEAR BY GENDER

- [2] L. Carter and R. Lee, "Modeling and forecasting U. S. sex differentials in mortality," *International Journal of Forecasting*, vol. 8, no. 3, pp. 393–411, November 1992.
- [3] I. Currie, J. Kirkby, M. Durban, and P. Eilers, "Smooth Lee-Carter models and beyond," in *Workshop on Lee-Carter Methods*, 2004.
- [4] A. Debón, , F. Martínez-Ruiz, and F. Montes, "Dynamic life tables: a geostatistical approach," in *8th International Congress on Insurance: Mathematics & Economics*, Rome, Italy, Junio 2004.
- [5] A. Felipe, M. Guillén, and A. Pérez-Marín, "Recent mortality trends in the spanish population," *British Actuarial Journal*, vol. 8, no. 4, pp. 757–786, 2002.
- [6] R. Lee, "The lee-carter method for forecasting mortality, with various extensions and applications," *North American Actuarial Journal*, vol. 4, no. 1, pp. 80–91, 2000.
- [7] R. Lee and L. Carter, "Modelling and forecasting U. S. mortality," *Journal of the American Statistical Association*, vol. 87, no. 419, pp. 659–671, Septiembre 1992.
- [8] R. McNown and A. Rogers, "Forecasting cause-specific mortality using time series methods," *International Journal of Forecasting*, vol. 8, no. 3, pp. 413–432, 1992.
- [9] ——, "Forecasting mortality: A parametrized time series aproach," *Demography*, vol. 26, no. 4, pp. 645–660, 1989.
- [10] E. Pitacco, "Survival models in dynamic context: a survey," *Insurance: Mathematics and Economics*, vol. 35, no. 2, pp. 279–298, 2004.
- [11] A. Renshaw and S. Haberman, "Lee-Carter mortality forecasting: a parallel generalized linear modelling aproach for England and Wales mortality projections," *Journal of the Royal Statistical Society C*, vol. 52, no. 1, pp. 119–137, 2003.
- [12] ——, "On the forecasting or mortality reduction factors," *Actuarial Research Report*, vol. 135, Febrero 2001.

	Lee-Carter			
	Men		Women	
Pred.	95% Interval	Pred.	95% Interval	
2000	-10,58	-12,32	-8,84	-17,32
2001	-11,50	-13,96	-9,05	-18,98
2002	-12,43	-15,44	-9,41	-20,64
2003	-13,35	-16,83	-9,87	-22,29
2004	-14,27	-18,16	-10,38	-23,94
2005	-15,19	-19,45	-10,94	-25,60
2006	-16,12	-20,71	-11,52	-27,25
2007	-17,04	-21,95	-12,12	-28,91
2008	-17,96	-23,17	-12,75	-30,56
2009	-18,88	-24,38	-13,39	-32,22
2010	-19,81	-25,57	-14,04	-33,87
2011	-20,73	-26,75	-14,71	-35,53

	Lee-Carter-GLM			
	Men		Women	
Pred.	95% Interval	Pred.	95% Interval	
2000	-8,11	-9,35	-6,88	-16,12
2001	-8,94	-10,68	-7,19	-17,73
2002	-9,76	-11,89	-7,63	-19,33
2003	-10,58	-13,05	-8,12	-20,94
2004	-11,40	-14,16	-8,65	-22,54
2005	-12,23	-15,24	-9,21	-24,15
2006	-13,05	-16,31	-9,79	-25,76
2007	-13,87	-17,35	-10,39	-27,36
2008	-14,69	-18,39	-10,99	-28,97
2009	-15,51	-19,41	-11,62	-30,57
2010	-16,34	-20,42	-12,25	-32,18
2011	-17,16	-21,43	-12,89	-33,782

TABLE II  
PREDICTIONS OF  $k_t$  FOR MEN AND WOMEN

	Lee-Carter		Lee-Carter-GLM	
	MSE	MAPE	MSE	MAPE
Men	0.01301	11.85	0.00788	10.69
Women	0.00893	7.69	0.00839	8.17

TABLE III  
MSE AND MAPE FOR PREDICTIONS FOR EACH GENDER.

- [13] E. Tabeau, A. Van den Berg Jeths, and C. H. (Eds), *A Review of Demographic Forecasting Models for Mortality. Forecasting in Developed Countries: From description to explanation*. Kluwer Academic Publishers, 2001.
- [14] J. Wilmoth, "Computational methods for fitting and extrapolating the Lee-Carter model of mortality change," *Technical Report, Departament of Demography, University of California, Berkeley*, 1993.



# Planning holidays and working time under annualised hours

Amaia Lusa, Albert Corominas and Rafael Pastor  
 Universitat Politècnica de Catalunya/Research Institute IOC  
 Av. Diagonal 647, p11, 08028 Barcelona, Spain  
 Email: {[albert.corominas@amaia.lusa@rafael.pastor@upc.edu](mailto:albert.corominas@amaia.lusa@rafael.pastor@upc.edu)}

**Abstract**—Annualising working hours (AH) is a means of achieving flexibility in the use of human resources to face the seasonal nature of demand. There are few papers dealing with the problem of planning staff working hours under an annualised hours agreement. The proposed planning procedures are able to minimise costs due to overtime and temporary workers but, due to the difficulty of solving the problem, it is normally assumed both that the holiday weeks are fixed beforehand and that workers from different categories who are able to perform a specific type of task have the same efficiency. In the present paper, those assumptions are relaxed and a more general problem is solved. The computational experiment leads to the conclusion that MILP is a technique suited to dealing with the problem.

**Keywords**—manpower planning, annualised hours, service industry, integer programming.

## I. INTRODUCTION

ANNUALISING working hours (AH)—i.e., the possibility of irregularly distributing the total number of staff working hours over the course of a year—is a means of achieving flexibility, because AH allows production capacity to be adapted to fluctuations in demand, thus reducing costs (overtime, temporary workers and inventory costs).

AH gives rise to new problems that have hitherto been given little attention in the literature. For instance, in [5], [6], [7] and [1] it is emphasised that the concept of annualised hours is surprisingly absent from the literature on planning and scheduling. A significant difficulty to be faced is that the diversity of production systems means that the problems that AH entails vary greatly; in [4], the characteristics of the planning problem are discussed and a classification scheme is proposed, giving rise to thousands of different cases; moreover, AH often implies the need to solve a complicated working time planning problem. Some authors deal with different versions of the problem (see, for example, [5], [6], [11], and [1]-[3]), but most papers (for example, [8]-[10]) discuss AH only from a qualitative point of view.

In [3], a MILP (Mixed Integer Linear Programming) model is used to solve the problem of

planning staff working hours with an annual horizon. Two hierarchical categories of workers are considered and the costs of overtime and of employing temporary workers are minimised. In the aforementioned paper, the following is assumed to ease its resolution: (i) the holiday weeks are fixed a priori; and (ii) the workers from different categories who are able to perform a specific type of task have the same efficiency.

Actually, although workers from different categories may be able to perform a specific type of task, obviously certain categories frequently require more time than others do. In addition, the allocation of holiday weeks may be a decision variable of the model with the objective both of minimising costs and helping in the bargaining process: computing the difference of costs between a situation in which holidays are fixed a priori and one in which those are decision variables, the company knows the maximum amount of money that could offer to workers in exchange of being able to fix their holidays in the best moment. Therefore, in this paper, assumptions in aforementioned paper are relaxed and a more general problem is solved.

The main aims are to approach the planning of working hours and holiday weeks over the course of a year in services that employ cross-trained workers who have different relative efficiencies, to show that MILP is an appropriate tool for this aim, and of course to verify that the possibility of determining holiday weeks with the model provides better results. The rest of the article is organised as follows: the section 2 introduces the problem and two MILP models for planning AH over a year; section 3 include the results of the computational experiment; section 4 show how results could be used to help in the bargaining process; and, finally, section 5 exposes the conclusions.

## II. TWO MILP MODELS TO PLAN HOLIDAYS AND WORKING TIME UNDER AH

Solving the planning problem involves determining the number of weekly working hours and holiday weeks for each member of staff.

A service system that is carried out on an individual basis is considered (so working hours for each worker may be different). Different types of tasks are involved, the product is not storable and the company forecasts the seasonal demand.

The production capacity in any given week must be greater than or equal to that which is needed and, if the staff does not provide entirely this capacity, temporary workers will be hired for the number of hours required. Overtime is admitted, but its total amount is bounded; overtime hours are classified into two blocks and the cost of an hour belonging to the second block is greater than that of an hour of the first. From the outset, the objective function is the cost of overtime plus the cost of employing temporary workers; it is possible to break the tie between optimal solutions by considering the penalties associated with the assignment of different types of tasks to categories of employees (adding this function to the first one with a small weight).

Workers from different categories may frequently be able to perform a specific type of task, although certain categories may require more time than others may. Therefore, cross-trained workers are considered: certain categories can perform different types of tasks and can have different relative efficiencies associated with them (for example, a value of 0.9 means that a worker in that category needs to work 1/0.9 hours to serve a demand that a worker with a relative efficiency equal to 1 would serve in 1 hour).

The conditions to be fulfilled by the solution are the following:

- i) the total of annual working hours is fixed;
- ii) the weekly number of working hours must fall within an interval defined by a lower and upper bound;
- iii) the average weekly working hours for any set of twelve consecutive weeks is upper bounded;
- iv) if the average weekly working hours over a specified number of consecutive weeks (“week-block”) is greater than a certain value, then over a given number of weeks immediately succeeding the week-block, the number of working hours must not be greater than a certain value;
- v) if “strong” and “weak” weeks are defined as those in which the number of working hours is respectively greater or less than certain specified values, there is an upper bound for the number of strong weeks and a lower bound for the number of weak weeks.

Below, we introduce the two models to be tested.

In *M1*, holiday weeks are determined by the model but, in *M2*, these are fixed a priori (in both cases, in the computational experiment, two

consecutive holiday weeks in winter and four consecutive holiday weeks in summer are assumed).

We use the following notation:

*Data*

$T$	Weeks in the planning horizon (in general, 52)
$C$	Set of categories of workers
$F$	Set of types of tasks
$E$	Set of members of staff
$\rho_{jk}$	Relative efficiency associated with the workers in category $j$ in the accomplishment of tasks of type $k$ ( $j=1,\dots, C $ ; $k=1,\dots, F $ ); $0 \leq \rho_{jk} \leq 1$ . If $\rho_{jk}=0$ , workers in category $j$ are not able to perform tasks of type $k$ .
$\hat{C}_k$	Sets of categories of workers that can be assigned to tasks of type $k$ ( $\hat{C}_k = j \in C   \rho_{jk} > 0$ )
$\hat{F}_j$	Sets of types of tasks which can be performed by employees in category $j$ ( $\hat{F}_j = k \in F   \rho_{jk} > 0$ )
$p_{jk}$	Penalty associated with an hour of work in a task of type $k$ of a staff member in category $j$ ( $\forall k \in F; \forall j \in \hat{C}_k$ )
$\lambda$	Parameter to weigh the penalties to establish the trade-off between these and the monetary costs of the solution.
$\hat{E}_j$	Set of employees in category $j$ ( $j=1,\dots, C $ )
$r_{tk}$	Required working hours for tasks of type $k$ in week $t$ ( $t=1,\dots,T$ ; $k=1,\dots, F $ )
$H_i$	Stipulated ordinary annual working hours of employee $i$ ( $\forall i \in E$ )
$\alpha_1, \alpha_2$	Maximum proportions, over the annual amount of ordinary working hours, of overtime corresponding to blocks 1 and 2 respectively.
$\beta l_i, \beta 2_i$	Respectively, the cost of an hour of overtime for block 1 and block 2 for employee $i$ ( $\forall i \in E$ ), with $\beta l_i < \beta 2_i$
$hm_{it}, hM_{it}$	Lower and upper bounds of the number of working hours for worker $i$ in week $t$ ( $\forall i \in E; t=1,\dots,T$ ); $hm_{it} < hM_{it}$
$L, h_L$	$L$ is the maximum number of consecutive weeks in which the average weekly working hours cannot be greater than $h_L$
$B, b, h_b, h_b$	$b$ is the minimum number of weeks, after a week-block of $B$ consecutive weeks with a weekly average of working hours greater than $h_B$ , in which the number of weekly hours cannot be greater than $h_b$
$N_S, h_S$	$N_S$ is the maximum number of “strong” weeks, i.e., weeks with a number of working hours greater than $h_S$
$N_W, h_W$	$N_W$ is the minimum number of “weak”

	weeks, i.e., weeks with a number of working hours equal or less than $h_w$
$hw1_i, hw2_i$	Number of holiday weeks in the first and second holiday periods respectively for worker $i$ ( $\forall i \in E$ )
$t1_i, t2_i$	First and last week respectively in which worker $i$ can take holidays in the first holiday period ( $\forall i \in E$ )
$t3_i, t4_i$	First and last week respectively in which worker $i$ can take holidays in the second holiday period ( $\forall i \in E$ )
$\gamma_k$	Cost of an hour for tasks of type $k$ performed by a worker who is not a member of staff ( $\gamma_k > \beta2_i, \forall i \in \hat{E}_j   j \in \hat{C}_k$ )

*Variables*

$x_{it}$	Working hours of employee $i$ in week $t$ ( $\forall i \in E; t = 1, \dots, T$ ).
$y_{tjk}$	Working hours of employees in category $j$ dedicated to tasks of type $k$ in week $t$ ( $\forall k \in F; \forall j \in \hat{C}_k; t = 1, \dots, T$ ).
$d_{tk}$	Working hours corresponding to tasks of type $k$ to be supplied in week $t$ by workers who are not members of staff ( $\forall k \in F; t = 1, \dots, T$ ).
$v1_i, v2_i$	Overtime corresponding respectively to blocks 1 and 2 of employee $i$ ( $\forall i \in E$ ).
$vc1_{it} \in \{0,1\}$	Indicates whether employee $i$ starts his or her first holiday period in week $t$ ( $\forall i \in E, t = t1_i, \dots, t2_i - hw1_i + 1$ ).
$vc2_{it} \in \{0,1\}$	Indicates whether employee $i$ starts his or her second holiday period in week $t$ ( $\forall i \in E, t = t3_i, \dots, t4_i - hw2_i + 1$ ).
$\delta_{it} \in \{0,1\}$	Indicates whether the average working hours of employee $i$ , in a week-block of $B$ weeks that ends with week $\tau$ , is (or is not) greater than $h_B$ hours ( $\forall i \in E; \tau = B, \dots, T - b$ ).
$s_{it} \in \{0,1\}$	Indicates whether employee $i$ has a planned number of working hours greater than $h_S$ hours for week $t$ ( $\forall i \in E; t = 1, \dots, T$ ).
$w_{it} \in \{0,1\}$	Indicates whether employee $i$ has a planned number of working hours equal to or less than $h_W$ hours for week $t$ ( $\forall i \in E; t = 1, \dots, T$ ).

All the non-binary variables are real and non-negative.

Now, the two models can be formalised.

*MODEL 1 (M1)*

$$\begin{aligned} [MIN] z = & \sum_{i \in E} \beta1_i \cdot v1_i + \sum_{i \in E} \beta2_i \cdot v2_i + \\ & \sum_{k \in F} \sum_{t=1}^T \gamma_k \cdot d_{tk} + \lambda \cdot \sum_{t=1}^T \sum_{k \in F} \sum_{j \in \hat{C}_k} p_{jk} \cdot y_{tjk} \end{aligned} \quad (1)$$

$$\sum_{t=1}^T x_{it} = H_i + v1_i + v2_i \quad \forall i \in E \quad (2)$$

$$v1_i \leq \alpha_1 \cdot H_i \quad \forall i \in E \quad (3)$$

$$v2_i \leq \alpha_2 \cdot H_i \quad \forall i \in E \quad (4)$$

$$\sum_{i \in \hat{E}_j} x_{it} = \sum_{k \in \hat{C}_j} y_{tjk} \quad t = 1, \dots, T; \forall j \in C \quad (5)$$

$$\sum_{j \in \hat{C}_k} \rho_{jk} \cdot y_{tjk} + d_{tk} \geq r_{ik} \quad t = 1, \dots, T; \forall k \in F \quad (6)$$

$$\sum_{t=t-L+1}^{\tau} x_{it} \leq L \cdot h_L \quad \tau = L, \dots, T; \forall i \in E \quad (7)$$

$$\sum_{t=\tau-B+1}^{\tau} x_{it} \leq B \cdot h_B + \left( \sum_{t=\tau-B+1}^{\tau} hM_{it} - B \cdot h_B \right) \cdot \delta_{it} \quad \tau = B, \dots, T-b; \forall i \in E \quad (8)$$

$$\sum_{t=\tau-B+1}^{\tau} x_{it} \leq B \cdot h_B \quad \tau = T-b+1, \dots, T; \forall i \in E \quad (9)$$

$$x_{i,\tau+l} \leq hM_{i,\tau+l} - (hM_{i,\tau+l} - h_b) \cdot \delta_{it} \quad \forall i \in E; \tau = B, \dots, T-b; l = 1, \dots, b \quad (10)$$

$$x_{it} \leq hS + (hM_{it} - h_S) \cdot s_{it} \quad \forall i \in E; t = 1, \dots, T \quad (11)$$

$$x_{it} \leq hM_{it} - (hM_{it} - h_W) \cdot w_{it} \quad \forall i \in E; t = 1, \dots, T \quad (12)$$

$$\sum_{t=1}^T s_{it} \leq N_S \quad \forall i \in E \quad (13)$$

$$\sum_{t=1}^T w_{it} \geq N_W \quad \forall i \in E \quad (14)$$

$$\sum_{t=t1_i}^{t2_i - hw1_i + 1} vc1_{it} = 1 \quad \forall i \in E \quad (15)$$

$$\sum_{t=t3_i}^{t4_i - hw2_i + 1} vc2_{it} = 1 \quad \forall i \in E \quad (16)$$

$$x_{it} \leq hM_{it} \quad \forall i \in E; t \notin ([t1_i, \dots, t2_i] \cup [t3_i, \dots, t4_i]) \quad (17)$$

$$x_{it} \geq hm_{it} \quad \forall i \in E; t \notin ([t1_i, \dots, t2_i] \cup [t3_i, \dots, t4_i]) \quad (18)$$

$$x_{it} \leq hM_{it} \cdot \left( 1 - \sum_{\tau=\max(t1_i, t-hw1_i+1)}^{\min(t, t2_i-hw1_i+1)} vc1_{i\tau} \right) \quad \forall i \in E; t = t1_i, \dots, t2_i \quad (19)$$

$$x_{it} \geq hm_{it} \cdot \left( 1 - \sum_{\tau=\max(t1_i, t-hw1_i+1)}^{\min(t, t2_i-hw1_i+1)} vc1_{i\tau} \right) \quad \forall i \in E; t = t1_i, \dots, t2_i \quad (20)$$

$$x_{it} \leq hM_{it} \cdot \left( 1 - \sum_{\tau=\max(t3_i, t-hw2_i+1)}^{\min(t, t4_i-hw2_i+1)} vc2_{i\tau} \right) \quad \forall i \in E; t = t3_i, \dots, t4_i \quad (21)$$

$$x_{it} \geq hm_{it} \cdot \left( 1 - \sum_{\tau=\max(t3_i, t-hw2_i+1)}^{\min(t, t4_i-hw2_i+1)} vc2_{i\tau} \right) \quad \forall i \in E; t = t3_i, \dots, t4_i \quad (22)$$

$$\delta_{it} \in \{0,1\} \quad \forall i \in E; \tau = B, \dots, T-b \quad (23)$$

$$s_{it}, w_{it} \in \{0,1\} \quad \forall i \in E; t = 1, \dots, T \quad (24)$$

$$vc1_{it} \in \{0,1\} \quad \forall i \in E; t = t1_i, \dots, t2_i - hw1_i + 1 \quad (25)$$

$$vc2_{it} \in \{0,1\} \quad \forall i \in E; t = t3_i, \dots, t4_i - hw2_i + 1 \quad (26)$$

$$v1_i, v2_i \geq 0 \quad \forall i \in E \quad (27)$$

$$y_{tjk} \geq 0 \quad t = 1, \dots, T; \quad \forall k \in F; \quad \forall j \in \hat{C}_k \quad (28)$$

$$d_{tk} \geq 0 \quad t = 1, \dots, T; \quad \forall k \in F \quad (29)$$

(1) is the objective function, which includes the cost of overtime plus that of employing external workers and the (weighted) penalties associated with the assignment of tasks to the types of employees on the staff; (2) imposes that the total number of worked hours should be equal to the ordinary annual hours stipulated plus overtime, if applicable; (3) and (4) stipulates that the overtime for each of the two blocks should not exceed their respective upper bounds; (5) is the balance between the hours provided by specific types of workers of the staff and the hours assigned to different types of tasks; (6) expresses that the hours assigned to a type of task that are to be carried out by members of staff plus, if applicable, the hours provided by external workers for that same type of task must not be less than the number of hours required; (7) imposes the upper bound on the average weekly working hours for any subset of  $L$  consecutive weeks; (8) implies that variable  $\delta_{it}$  is equal to 1 if the average number of working hours in a week-block of  $B$  weeks is greater than  $h_B$ ; (9) prevents the average hours worked from being greater than  $h_B$  in the last weeks of the year, when after the week-block of  $B$  weeks there are no longer  $b$  weeks to “compensate”; (10) implies that, if variable  $\delta_{it}$  is equal to 1, the upper bound of the number of working hours is  $h_b$ ; (11) imposes that, if the number of working hours is greater than  $h_S$ , then variable  $s_{it}$  is equal to 1; (12) states that, if the number of working hours is greater than  $h_W$ , then variable  $w_{it}$  is equal to 0; (13) and (14) stipulate that the number of “strong” and “weak” weeks cannot be greater than  $N_S$  and less than  $N_W$  respectively; (15) and (16) establish that the worker must start his or her holidays in a one and only one week; (17) and (18) set the lower and upper bounds of the number of weekly working hours in non-holiday weeks; (19), (20), (21) and (22) set the lower and upper bounds of the number of weekly working hours for possible holiday weeks; (23), (24), (25) and (26) express the binary character of the corresponding variables; and (27), (28) and (29) show the non-negative character of the rest of the non-binary variables.

#### MODEL 2 (M2)

*M2* can be obtained by deleting the variables  $vc1_{it}$  and  $vc2_{it}$  and their associated constraints (15, 16 and 19 to 22, 25 and 26) from model *M1* and making several minor modifications to equations (2), (5), (7)-(14), (17), (18), (23) and (24). First of all, since

holidays are fixed a priori, additional data has to be defined and the range of some data and variables has to be redefined to take into account holiday weeks.

$S_i$  Set of working weeks for employee  $i$  ( $\forall i \in E$ )

$hm_{it}$ ,  $hM_{it}$ ,  $x_{it}$ ,  $s_{it}$  and  $w_{it}$  are defined  $\forall i \in E; \forall t \in S_i$

$\delta_{it}$  it is considered now that the constraint only applies to blocks of  $B$  consecutive working weeks. Thus, this variable is defined  $\forall i \in E; \tau = B, \dots, T-b | [\tau-B+1.. \tau] \in S_i$

The model is formulated as follows:

$$[MIN] z = \sum_{i \in E} \beta 1_i \cdot v1_i + \sum_{i \in E} \beta 2_i \cdot v2_i + \sum_{k \in F} \sum_{t=1}^T \gamma_k \cdot d_{tk} + \lambda \cdot \sum_{t=1}^T \sum_{k \in F} \sum_{j \in \hat{C}_k} p_{jk} \cdot y_{tjk} \quad (1)$$

$$\sum_{\forall t \in S_i} x_{it} = H_i + v1_i + v2_i \quad \forall i \in E \quad (30)$$

$$v1_i \leq \alpha_1 \cdot H_i \quad \forall i \in E \quad (3)$$

$$v2_i \leq \alpha_2 \cdot H_i \quad \forall i \in E \quad (4)$$

$$\sum_{i \in \hat{E}_j | t \in S_i} x_{it} = \sum_{k \in \hat{F}_j} y_{tjk} \quad t = 1, \dots, T; \quad \forall j \in C \quad (31)$$

$$\sum_{j \in \hat{C}_k} \rho_{jk} \cdot y_{tjk} + d_{tk} \geq r_{ik} \quad t = 1, \dots, T; \quad \forall k \in F \quad (6)$$

$$\sum_{t=\tau-L+1}^{\tau} x_{it} \leq L \cdot h_L \quad (32)$$

$$\forall i \in E; \quad \tau = L, \dots, T | [\tau-L+1.. \tau] \in S_i$$

$$\sum_{t=\tau-B+1}^{\tau} x_{it} \leq B \cdot h_B + \left( \sum_{t=\tau-B+1}^{\tau} hM_{it} - B \cdot h_B \right) \cdot \delta_{it} \quad (33)$$

$$\forall i \in E; \quad \tau = B, \dots, T-b | [\tau-B+1.. \tau] \in S_i$$

$$\sum_{t=\tau-B+1}^{\tau} x_{it} \leq B \cdot h_B \quad (34)$$

$$\forall i \in E; \quad \tau = T-b+1, \dots, T | [\tau-B+1.. \tau] \in S_i$$

$$x_{i,\tau+l} \leq hM_{i,\tau+l} - (hM_{i,\tau+l} - h_b) \cdot \delta_{it} \quad \forall i \in E; \quad \tau = B, \dots, T-b; \quad l = 1, \dots, b | [\tau \in S_i \wedge (\tau+l) \in S_i] \quad (35)$$

$$x_{it} \leq h_S + (hM_{it} - h_S) \cdot s_{it} \quad \forall i \in E; \quad \forall t \in S_i \quad (36)$$

$$x_{it} \leq hM_{it} - (hM_{it} - h_W) \cdot w_{it} \quad \forall i \in E; \quad \forall t \in S_i \quad (37)$$

$$\sum_{t \in S_i} s_{it} \leq N_S \quad \forall i \in E \quad (38)$$

$$\sum_{t \in S_i} w_{it} \geq N_W \quad \forall i \in E \quad (39)$$

$$x_{it} \leq hM_{it} \quad \forall i \in E; \quad \forall t \in S_i \quad (40)$$

$$x_{it} \geq hm_{it} \quad \forall i \in E; \quad \forall t \in S_i \quad (41)$$

$$\delta_{it} \in \{0,1\} \quad \forall i \in E; \quad \tau = B, \dots, T-b | [\tau-B+1.. \tau] \in S_i \quad (42)$$

$$s_{it}, w_{it} \in \{0,1\} \quad \forall i \in E; \quad \forall t \in S_i \quad (43)$$

$$v1_i, v2_i \geq 0 \quad \forall i \in E \quad (27)$$

$$y_{tjk} \geq 0 \quad t = 1, \dots, T; \quad \forall k \in F; \quad \forall j \in \hat{C}_k \quad (28)$$

$$d_{tk} \geq 0 \quad t = 1, \dots, T; \quad \forall k \in F \quad (29)$$

### III. COMPUTATIONAL EXPERIMENT

A large-scale computational experiment was performed to evaluate the effectiveness (in terms of computing time and the quality of the solutions) of the models. Overall, the results were very satisfactory.

The basic data used for the experiment are as follows:

- Two MILP models: *M1* and *M2*.
- 10, 40, 70, 100 and 250 staff workers.
- A time horizon of 52 weeks (46 working weeks and 6 holiday weeks).
- The holiday weeks for each worker are distributed into two uninterrupted periods, including two weeks in winter and four weeks in summer. In *M2*, the temporary allocation of holidays was fixed for each worker at random.
- There are three categories and three types of tasks. There are two patterns of relative efficiency (and penalty). Table I and Table II show the relative efficiency (and the penalty) values for each pattern.

TABLE I

RELATIVE EFFICIENCY AND PENALTY VALUES, PATTERN 1

	Task 1	Task 2	Task 3
Category 1	1 (1)	0.9 (2)	0
Category 2	0	1 (1)	0.9 (2)
Category 3	0	0	1 (1)

TABLE II

RELATIVE EFFICIENCY AND PENALTY VALUES, PATTERN 2

	Task 1	Task 2	Task 3
Category 1	1 (1)	0	0
Category 2	0.9 (2)	1 (1)	0
Category 3	0.8 (2)	0	1 (1)

- The capacity (in working hours) required over the year follows three different patterns. Demand Type 1 corresponds to a non-seasonal capacity pattern with noise (Figure I). Demand Type 2 corresponds to a seasonality pattern with one peak (Figure II), with noise. Demand Type 3 corresponds to a seasonality pattern with two peaks, with noise (Figure III). In each case, the total demand is equal to the total capacity multiplied by 0.99.

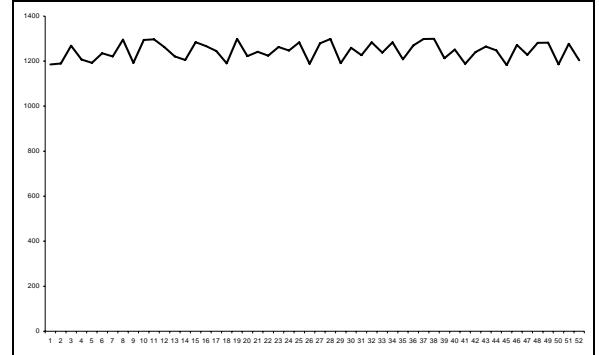


Fig. 1. Type of demand 1 (no seasonality)

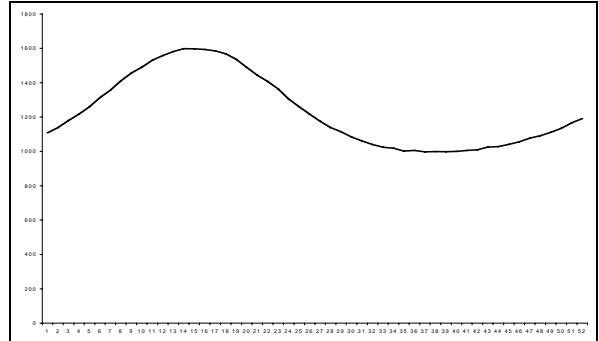


Fig. 2. Type of demand 2 (one peak)

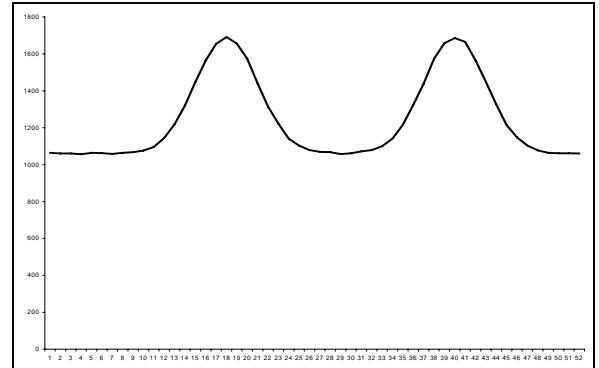


Fig. 3. Type of demand 3 (two peaks)

For every combination of models, number of staff workers, type of demand and pattern of relative efficiency (and penalty), 20 instances were generated (varying demand noise and, in *M2*, holiday weeks at random), which gave 1,200 instances.

In spite of the dimension of the models, which may be considered large (for example, on average, for 250 workers *M1* has 55,377 variables and 90,315 constraints and *M2* has 42,689 variables and 56,572 constraints), they were solved to optimality using an ILOG CPLEX 8.1 optimiser and a Pentium IV PC at 1.8 GHz with 512 Mb of RAM. The absolute and relative MIP gap tolerances were set to 0.01. The maximum computing time for all instances was set to 1,800 seconds.

Feasible solutions were always obtained and most of these were optimal solutions. Model *M2* gave

always an optimal solution, whilst for *M1* optimal solutions were obtained in a 50.83%, 52.5%, 94.17%, 99.17% and 100% for 10, 40, 70, 100 and 250 workers, respectively.

The maximum computing times are very reasonable considering the problem to be solved (the aim of the models is to establish an annual plan) and its maximum size (two hundred and fifty workers, which is a large enough number, since we are supposed to be dealing with a production system of services or a part of this system). Model *M1* was harder than *M2* to solve, as expected, given that this model include more constraints and binary variables than *M2*.

The experiments provided satisfactory results regarding the quality of the solutions of the models. The possibility of determining holiday weeks with model *M1*, whilst observing a set of legal constraints or constraints imposed by a collective bargaining agreement between the management and the workers (two uninterrupted weeks in winter and four in summer in this case), provides very good solutions and an average saving of more than 90% (for all number of workers). These values also show how the capacity of the staff can be adapted to demand by determining the holiday weeks of the staff (this is also due to the flexibility provided by the annualisation of working time).

Another computational experiment was performed with the following new data: total demand is equal to total capacity multiplied by 1.05; for each combination, 5 instances were generated (giving 300 new instances).

The results show that if the system is not adequately sized (total capacity is less than total demand), the solution is a little more difficult (and the number of optimal/feasible solutions obtained decreases); the results, nevertheless, can be considered very good.

#### IV. A TOOL FOR A BARGAINING PROCESS

In most countries companies cannot introduce irregular working hours if workers do not agree, so the question is whether workers will really accept an increase in flexibility (and also their holidays being planned by the company). Besides the convincing argument of conserving their jobs even in periods of low demand, companies should offer some kind of compensation that will lead workers to accept more or less flexibility. Hence, it is essential to have a tool to help in the bargaining.

Planning working time under different AH scenarios provides the company and the workers with quantitative information that can be very useful for the bargaining process in order to adopt an annual hours scheme. These scenarios may be characterised, for example, by the weekly flexibility accepted by workers, the total amount of annual

working hours (the company could eventually reduce the annual working time), the maximum overtime, the conditions related to the strong and weak weeks and, of course, the possibility of, some rules provided, planning the holiday weeks. For each scenario, the model (*M1* if holidays can be planned and *M2* otherwise) would give the cost of the solution and the company and the workers could agree to satisfactory conditions for both. Obviously, doing this implies solving several instances of the model. Hence, this would be possible only if solving the model requires a reasonable time, which is the case of the models presented in this paper.

Table III shows the results of a specific case in which scenarios are characterised by the total amount of annual hours (first column) and the weekly flexibility (first row). For each scenario, the first and second values correspond to the cost obtained by *M1* –holidays fixed by the model– and *M2* –holidays fixed a priori–, respectively. Note that *K* is the cost obtained in a situation without flexibility, without a reduction in working time and with holidays fixed a priori. It can be seen how the cost diminishes when flexibility is high, even when reducing working time.

Two options for reducing the cost by implementing annualised hours might be as follows: (1) by increasing weekly flexibility and reducing working time as a compensation for the workers; or (2), by increasing flexibility and not reducing working time but instead offering financial compensation to the workers. As it is shown in Table III, in both cases the cost can be further reduced if workers' holidays are planned by the model.

TABLE III  
COST OF DIFFERENT SCENARIOS (ANNUAL HOURS, WEEKLY FLEXIBILITY AND PLANNING HOLIDAYS)

	[40,40]	[40, 50]	[30, 45]	[25, 50]
1,840	0.64· <i>K</i> <i>K</i>	0.52· <i>K</i> 0.86· <i>K</i>	0.16· <i>K</i> 0.52· <i>K</i>	0 0.21· <i>K</i>
1,748	-	-	0.16· <i>K</i> 0.51· <i>K</i>	0 0.21· <i>K</i>
1,610	-	-	0.45· <i>K</i> 0.58· <i>K</i>	0 0.21· <i>K</i>

#### V. CONCLUSIONS

Annualising working hours (AH) is a means of obtaining flexibility in the use of human resources to face the seasonal nature of demand. There are few papers dealing with the problem of planning staff working hours under an annualised hours agreement; moreover, most of them include tough assumptions. For example, in [3], a MILP model is used to solve the following AH problem: the costs of overtime and employing temporary workers are

minimised. To facilitate the solving of the model, however, the following is assumed: (i) the holiday weeks are fixed a priori; and (ii) the workers from different categories who are able to perform a specific type of task have the same efficiency.

In this paper, these assumptions are relaxed and a more general problem is solved: planning the working hours and holiday weeks of cross-trained workers who have different relative efficiencies over the course of a year in the service sector. Our computational experiment leads us to conclude that MILP is a technique suited to dealing with the problem in many real situations and, as is obvious, that better results are obtained when the holiday weeks are determined by the model. Finally, it has been shown how the MILP models could be a useful tool for helping in the bargaining process carried out before the adoption of an annual hours scheme.

#### ACKNOWLEDGEMENTS

Supported by the Spanish MCyT projects DPI2001-2176 and DPI2004-05797, co-financed by FEDER.

#### REFERENCES

- [1] Azmat C and Widmer M (2004). A case study of single shift planning and scheduling under annualized hours: A simple three step approach. *European Journal of Operational Research* 153 (1), 148-175.
- [2] Azmat C, Hürlimann T and Widmer M (2004). Mixed Integer Programming to Schedule a Single-Shift Workforce under Annualized Hours. *Annals of Operation Research* 128, 199-215.
- [3] Corominas A, Lusa A and Pastor R (2002). Using MILP to plan annualised hours. *Journal of the Operational Research Society* 53, 1101-1108.
- [4] Corominas A, Lusa A and Pastor R (2004). Characteristics and classification of annualised working hours planning problems. *International Journal of Services Technology and Management* 5/6, 435-447.
- [5] Hung R (1999). Scheduling a workforce under annualized hours. *International Journal of Production Research* 37 (11), 2419-2427.
- [6] Hung R (1999). A multiple-shift workforce scheduling model under annualized hours. *Naval Research Logistic* 46 (6), 726-736.
- [7] Grabot B and Letouzey A (2000). Short-term manpower management in manufacturing systems: new requirements and DSS prototyping. *Comp Ind* 3 (1), 11-29.
- [8] Lynch P (1995). Annual Hours: An idea whose time has come. *Personnel Management* November, 46-50.
- [9] MacMeeking J (1995). Why Tesco's new composite distribution needed annual hours. *International Journal Retail Distribution Management* 23 (9), 36-38.
- [10] Mazur L (1995). Coming: the annual workweeks. *Across the Board* 32 (4), 42-45.
- [11] Vila GFE and Astorino JM (2001). Annualized hours as a capacity planning tool in make-to-order or assemble-to-order environment: an agricultural implements company case. *Production Planning & Control* 12, (4), 388-398.



# Soft computing-based aggregation methods for human resource management

Lourdes Canós \* and Vicente Liern†,

\*Universitat Politècnica de València. Dep. Organización de Empresas, Eco. Fin. y Cont.

Camino de Vera, s/n. 46022-Valencia (Spain)

Email: loucada@omp.upv.es

†Universitat de València. Dep. Matemàtica Econòmica-Empresarial

Av. Tarongers, s/n. 46071-Valencia (Spain)

Email: Vicente.Liern@uv.es

**Abstract**—The main idea of this paper is to help managers in their decision making function by providing a flexible decision support system. In considering personnel selection, OWA aggregation operators allow us to assign different weights to different selection criteria to simulate expert valuation. The flexibility of these techniques allows us to state a ranking of the applicants for a job. Besides, we show an aggregation model based on effience analysis to order the candidates.

**Keywords**—Personnel selection, Fuzzy sets, OWA operators, Efficient aggregation.

## I. INTRODUCTION

MAKING right decisions about human resources policies can determine the success in companies. Personnel selection is the process for selecting a person from among a set of applicants. An accurate selection, taking into account the company circumstances, allows managers to optimize production costs and to achieve corporative goals [1]. This process is complicated because of the human nature, and implies focusing in some concepts like validity, trust and criteria fixing. The main goal of managers is to obtain a ranking of candidates which have been valued according to different competences. Therefore, the development of efficient and flexible information aggregation methods has become a main issue in information access methods.

Fuzzy set theory considers some elements that are essential to deal with economic, social and technological situations: the uncertainty in data, and the modeler or manager capacity to add any additional information.

Weighted aggregation has been widely used in fuzzy decision making, where a set of weights is used to represent the relative importance that the decision maker gives to different decision criteria [8]. Classical aggregation operators are the arithmetic mean and quasi-arithmetic means, for instance, the geometric, harmonic

and quadratic means [2], and the well known OWA operators [15, 17]. Jaquet-Lagrèze and Siskos review in [10] some useful methods to establish a ranking based in the importance of every competence [16]. However, in this paper we analyze the case in which the competences are not important one by one, but the case in which bad and good valuations can be compensated.

On the other hand, we can do an aggregation based in parametric weights. Concretelly, we consider the weights as functions depending on a parameter  $\alpha$  that represents the satisfaction level. In some cases, an important attribute with a low satisfaction value can be penalized by means of its weight, to render the given attribute less significant in the overall evaluation. There are three main approaches to choose the set of weights [12]: the indifference trade-off method, the direct weighting method and the probabilistic equivalence technique.

In this paper we present two personnel selection models. In Section III we deal with some OWA operators and we use them to imitate the experts' opinion in the selection process. In Section IV, we introduce a parametric aggregation model whose performance is shown on a numerical example.

## II. FUZZY OPTIMIZATION

A classical set (*crisp set*)  $A \subseteq X$  is defined as a collection of elements  $x \in X$  where each single element can either belong to or not belong to it. We can express this by using the characteristic function of  $A$ , i. e.,

$$\chi_A(x) = \begin{cases} 1 & \text{if } x \in A \\ 0 & \text{if } x \notin A \end{cases} \quad (1)$$

This binary function can be generalized by defining a characteristic function such that  $\chi_A(x) \in [0, 1]$ . Formally, we can express this idea by means of the following definition:

**Definition 1** Let  $X$  be a universal (crisp) set. A fuzzy set  $\tilde{A}$  in  $X$  is a set of ordered pairs:

$$\tilde{A} = \{(x, \mu_{\tilde{A}}(x)), \quad x \in X\}, \quad (2)$$

where  $\mu_{\tilde{A}}(x)$  is the membership function or degree of membership of  $x$  in  $A$ .

In this framework, a decision problem can be formulated as [19]:

**Definition 2** Assume that we are given a fuzzy goal  $\tilde{G}$  and a fuzzy constraint  $\tilde{C}$  in a space of alternatives  $X$ . Then  $\tilde{G}$  and  $\tilde{C}$  combine to form a decision  $\tilde{D}$ , which is a fuzzy set resulting from intersection of  $\tilde{G}$  and  $\tilde{C}$ , i.e.,  $\tilde{D} = \tilde{G} \cap \tilde{C}$ , and

$$\mu_{\tilde{D}} = \min\{\mu_{\tilde{G}}, \mu_{\tilde{C}}\}.$$

Note that the intersection of fuzzy sets is defined in the probabilistic sense by the min-operator. This approach makes the problem symmetric, i.e., the degree of improvement of the goal is considered as important as the degree of feasibility of the solution. If we want a final crisp decision, we look for a solution where  $\mu_{\tilde{D}}$  is maximum.

### III. PERSONNEL SELECTION BY USING ORDERED WEIGHTED AVERAGE (OWA) OPERATORS

We have  $n$  candidates  $\{P_i\}_{i=1}^n$  to fill a vacancy. They have been evaluated in  $R$  competences  $\{x_j\}_{j=1}^R$ . Let us assume that the experts have given a global valuation independent of previous valuations by using their intuition and experience for  $\{P_i\}_{i=1}^L$  candidates ( $L < n$ ), see TABLE I. Clearly, the company would be interested in the experts' opinion for the remaining candidates [3]. However, when a personnel selection involves many candidates, the evaluation process performed by external experts would be too long and expensive. The OWA operators can be very useful to our aim.

The Ordered Weighted Average operator was introduced by Yager in 1988 [17]. An OWA operator of dimension  $n$  is a mapping  $F : \mathbb{R}^n \rightarrow \mathbb{R}$ , characterized by an  $n$ -dimensional vector,  $w = (w_1, \dots, w_n)^T$ , called the weighting vector such that

$$w_i \in [0, 1], \quad 1 \leq i \leq n,$$

$$\sum_{i=1}^n w_i = 1.$$

where

$$F(x_1, x_2, \dots, x_n) = \sum_{k=1}^n w_k x_{j_k},$$

being  $x_{j_k}$  the  $k$ -th largest element of the collection  $\{x_1, x_2, \dots, x_n\}$ .

By construction, a fundamental aspect of an OWA operator is the re-ordering step. An aggregate  $x_i$  is not associated with a particular weight  $w_j$ , but rather a weight is associated with a particular ordered position  $j$  of the arguments. In fact, this ordering introduces the non-linearity into the aggregation process [6]. As the function value  $F = (x_1, x_2, \dots, x_n)$  determines the aggregated value of arguments  $x_1, x_2, \dots, x_n$ , in particular, for the vectors

$$\begin{aligned} W^1 &= [1, 0, \dots, 0]^T, \\ W^2 &= [0, 1, \dots, 0]^T, \\ W^3 &= [1/n, 1/n, \dots, 1/n]^T \end{aligned}$$

we obtain the max, min, and arithmetic mean operators, respectively [9].

TABLE I  
PARTIAL AND GLOBAL VALUATION OF THE CANDIDATES

	$x_1$	$x_2$	...	$x_R$	GEV*
$P_1$	$a_{11}$	$a_{12}$	...	$a_{1R}$	$v_1$
$P_2$	$a_{21}$	$a_{22}$	...	$a_{2R}$	$v_2$
$\vdots$	$\vdots$	$\vdots$		$\vdots$	$\vdots$
$P_L$	$a_{L1}$	$a_{L2}$	...	$a_{LR}$	$v_L$
$P_{L+1}$	$a_{(L+1)1}$	$a_{(L+1)2}$	...	$a_{(L+1)R}$	
$\vdots$	$\vdots$	$\vdots$		$\vdots$	
$P_n$	$a_{n1}$	$a_{n2}$	...	$a_{nR}$	

\* GEV means Global Expert Valuation.

In order to make a decision about the selection process, the human resources department should aggregate all the information about every candidate. The first step consists in trying to assign, at least approximately, a global valuation to every candidate. In many situations, it is unrealistic to assign a fixed weight to every competence because low levels in some competences can be compensated by high levels in some others, this idea is assumed when the experts give a global valuation. With this aim, for each candidate we order their competence values from major to minor and we express the results by means of a matrix.

$$A = \begin{pmatrix} a_{1j_1} & a_{1j_2} & \cdots & a_{1j_R} \\ a_{2j_1} & a_{2j_2} & \cdots & a_{2j_R} \\ \vdots & \vdots & \ddots & \vdots \\ a_{Lj_1} & a_{Lj_2} & \cdots & a_{Lj_R} \\ a_{(L+1)j_1} & a_{(L+1)j_2} & \cdots & a_{(L+1)j_R} \\ \vdots & \vdots & \ddots & \vdots \\ a_{nj_1} & a_{nj_2} & \cdots & a_{nj_R} \end{pmatrix} \quad (3)$$

where  $a_{ij_k}$  represents the  $k$ -th greatest score of the candidate  $P_i$ .

To obtain the weights that the experts have 'intuitively' used, we solve the next quadratic programming problem [8]:

$$\begin{aligned}
 (\text{Ow}) \quad & \text{Min} \quad \sum_{i=1}^L \left( \sum_{k=1}^R a_{ijk} w_k - v_i \right)^2 \\
 \text{s. t.} \quad & \sum_{k=1}^R w_k = 1 \\
 & w_k \geq 0 \\
 & k = 1, 2, \dots, R
 \end{aligned} \tag{4}$$

The solution of the program (Ow) gives a vector of weights  $W = [w_1^*, w_2^*, \dots, w_R^*]^T$  that allows the valuation of all the candidates by using the following expression:

$$v_i^* = \sum_{k=1}^R w_k^* a_{ijk}$$

Now, we can sort the candidates by using the values  $v_1^*, v_2^*, \dots, v_n^*$ , and we have solved our problem. If there is a draw, we must break the tie with some criteria such as the arithmetic average or the greatest score in some specific competences [3].

The procedures involved in the model above can be performed by using MS Excel. We summarize our proposal in the following algorithm.

#### ALGORITHM 1

##### Inputs:

Global valuation of some candidates:

$$v_1, v_2, \dots, v_L, \quad L < n$$

Valuation of every candidate in every competence.

STEP 1: Order valuations from every candidate in a decreasing way.

$$A = (a_{ijk}), \quad 1 \leq i \leq n, \quad 1 \leq k \leq R$$

STEP 2: Calculate the weights,  $w_1^*, w_2^*, \dots, w_R^*$ , of every competence by using the programming model (Ow).

STEP 3: Calculate the global valuation of each candidate:

$$v_i^* = \sum_{k=1}^R w_k^* a_{ijk}, \quad i = 1, \dots, R.$$

STEP 4: Order the candidates according to  $v_i^*$ .

**Output:** Selection.

#### A. OWA operators and tolerance intervals

In practice is more comfortable and realistic to value the competences by using tolerance intervals,

$$I_{ij} = [b_{ij}, d_{ij}],$$

because it is closer to the human thinking. In this case, we express the valuations as in Table II.

TABLE II  
PARTIAL AND GLOBAL VALUATION WITH INTERVALS

	$x_1$	$x_2$	...	$x_R$	GEV
$P_1$	$I_{11}$	$I_{12}$	...	$I_{1R}$	$v_1$
$P_2$	$I_{21}$	$I_{22}$	...	$I_{2R}$	$v_2$
$\vdots$	$\vdots$	$\vdots$		$\vdots$	$\vdots$
$P_L$	$a_{L1}$	$a_{L2}$	...	$a_{LR}$	$v_L$
$P_{L+1}$	$I_{(L+1)1}$	$I_{(L+1)2}$	...	$I_{(L+1)R}$	
$\vdots$	$\vdots$	$\vdots$		$\vdots$	
$P_n$	$I_{n1}$	$I_{n2}$	...	$I_{nR}$	

As we exposed before, a fundamental aspect is the re-ordering of the valuations. Since the valuations are expressed as intervals, it is necessary to introduce an ordering relation in the set of intervals. For this aim, we can use the next definition [7]:

**Definition 3** Let  $A = [a_1, a_2]$ ,  $B = [b_1, b_2] \subset \mathbb{R}$  be two intervals.  $A$  is larger than  $B$ ,  $A \succ B$  if and only if

$$\begin{cases} \frac{k_1 a_1 + k_2 a_2}{k_1 + k_2} > \frac{k_1 b_1 + k_2 b_2}{k_1 + k_2}, & k_1 a_1 + k_2 a_2 \neq k_1 b_1 + k_2 b_2 \\ a_1 > b_1, & k_1 a_1 + k_2 a_2 = k_1 b_1 + k_2 b_2 \end{cases}$$

where  $k_1$  and  $k_2$  are two positive constants fixed a priori.

Notice that if we change  $a_1 > b_1$  by  $a_2 > b_2$  the order would be less pessimistic.

By applying Definition 3, we construct the matrix of intervals  $A(I)$ .

$$A(I) = \begin{pmatrix} I_{1j_1} & I_{1j_2} & \cdots & I_{1j_R} \\ I_{2j_1} & I_{2j_2} & \cdots & I_{2j_R} \\ \vdots & \vdots & \ddots & \vdots \\ I_{Lj_1} & I_{Lj_2} & \cdots & I_{Lj_R} \\ I_{(L+1)j_1} & I_{(L+1)j_2} & \cdots & I_{(L+1)j_R} \\ \vdots & \vdots & \ddots & \vdots \\ I_{nj_1} & I_{nj_2} & \cdots & I_{nj_R} \end{pmatrix} \tag{5}$$

where  $I_{ij_r}, r = 1, \dots, R$  represents the  $r$ -th greatest interval of valuation for the candidate  $P_i$ .

We can distinguish two cases: that in which we suppose that any possible increasing or decreasing of the competences is balanced and that in which we can not assume this hypothesis. Let us show two methods to deal with both situations.

### A.1 Assuming trade-off

If we accept that there is a trade-off in the competences of all candidates, we can write the elements of matrix  $A(I)$  as intervals depending on a parameter:

$$I_{ij_r} \rightarrow b_{ij_r} + (d_{ij_r} - b_{ij_r})\alpha, \quad \alpha \in [0, 1]. \quad (6)$$

We denote these intervals as

$$a_{ij_r}(\alpha) = b_{ij_r} + (d_{ij_r} - b_{ij_r})\alpha, \quad \alpha \in [0, 1] \quad (7)$$

and we solve the following programming model (see [5]):

$$\begin{aligned} (\text{Ow2}) \quad \text{Min} \quad & \sum_{i=1}^L \left( \sum_{r=1}^R a_{ij_r}(\alpha) w_r - v_i \right)^2 \\ \text{s. t.} \quad & \sum_{r=1}^R w_r = 1 \\ & \alpha \in [0, 1] \\ & w_r \geq 0 \\ & r = 1, 2, \dots, R \end{aligned} \quad (8)$$

(Ow2) provides the results for the weights,  $w_r^*$ , and the parameter  $\alpha^*$  that best fits the external experts' opinion. With this information, we evaluate all the candidates by using the following expression.

$$v_i^* = \sum_{k=1}^R w_k^* a_{ij_k}(\alpha^*).$$

### A.2 Without a perfect trade-off

In this case, we can not parametrize the intervals as before. So, we choose a representant of the elements of matrix  $A(I)$  (see [7]). According to Definition 3, we calculate

$$a_{ij_r} = \frac{k_1 b_{ij_r} + k_2 d_{ij_r}}{k_1 + k_2}, \quad 1 \leq i \leq n, 1 \leq r \leq R. \quad (9)$$

For this  $a_{ij_r}$  we solve the programming model (Ow) and, finally, we calculate the global valuation of each candidate.

*Remark 1.* By construction,  $a_{ij_r}$  represents the  $r$ -th greatest defuzzyfied value for the candidate  $P_i$ .

*Remark 2.* Taking into account expression (9), the solution of case A. 2 is a particular case of A. 1 making

$$\alpha = \frac{k_2}{k_1 + k_2}.$$

## IV. PARAMETRIC AGGREGATION TECHNIQUES

Following Kao and Liu ideas [11] to obtain efficiencies depending on a parameter,  $\alpha$ , we have designed a flexible model to calculate intervals in which the weights of every competence can fluctuate.

Let  $n$  be the number of candidates to be evaluated in  $R$  competences. For the  $i$ -th candidate we suppose that the relative importance of the  $j$ -th competence is given by the fuzzy number:

$$\tilde{W}_{ij} = \left\{ (w_{ij}, \mu_{\tilde{W}_{ij}}(w_{ij})) : w_{ij} \in W_{ij} \right\}, \quad 1 \leq j \leq R, \quad (10)$$

where  $W_{ij}$  is the universal crisp set of the  $j$ -th weight for the  $i$ -th candidate and  $\mu_{\tilde{W}_{ij}}$  is the membership function of  $\tilde{W}_{ij}$ . On the other hand, we assume that the competences of the  $i$ -th candidate are given by the fuzzy number:

$$\tilde{C}_{ij} = \left\{ (c_{ij}, \mu_{\tilde{C}_{ij}}(c_{ij})) : c_{ij} \in C_{ij} \right\}, \quad 1 \leq j \leq R, \quad (11)$$

where  $C_{ij}$  is the universal crisp set of the  $j$ -th competence for the  $i$ -th candidate and  $\mu_{\tilde{C}_{ij}}$  is a membership function.

Similarly to the crisp weighted average, we can define the fuzzy weighted average as

$$\tilde{Y}_i = \frac{\sum_{j=1}^R \tilde{w}_j \tilde{c}_j}{\sum_{j=1}^R \tilde{w}_j}, \quad 1 \leq i \leq n. \quad (12)$$

This fuzzy quantity is computed for each candidate and provides us with a global measure of their corresponding adaptation for the vacancy.

According to the extension principle [18], the membership function of  $\tilde{Y}_i$  is:

$$\mu_{\tilde{Y}_i}(y_i) = \sup \min_{c,w} \left\{ \mu_{\tilde{W}_{ij}}(w_{ij}), \mu_{\tilde{C}_{ij}}(c_{ij}), \quad 1 \leq j \leq R \right\},$$

$$\text{where } y_i = \sum_{j=1}^R w_{ij} c_{ij} / \sum_{i=1}^R w_{ij}.$$

In practice, we can obtain  $\mu_{\tilde{Y}_i}$  by solving the next mathematical programming model:

$$\begin{aligned} \mu_{\tilde{Y}_i}(y_i) = & \text{ Max. } z \\ \text{s. t. } & z \leq \mu_{\tilde{C}_{ij}}(c_{ij}), \quad 1 \leq j \leq R, \\ & z \leq \mu_{\tilde{W}_{ij}}(w_{ij}), \quad 1 \leq j \leq R, \\ & y_i = \sum_{j=1}^R w_{ij} c_{ij} / \sum_{i=1}^R w_{ij}, \\ & w_{ij} \in W_{ij}, c_{ij} \in C_{ij}, \quad 1 \leq j \leq R. \end{aligned}$$

We express  $\alpha$ -cuts of  $\tilde{W}_{ij}$  and  $\tilde{C}_{ij}$  as follows:

$$\begin{aligned}\tilde{w}_{ij}(\alpha) &= \{w_{ij} \in W_{ij} : \mu_{\tilde{W}_{ij}}(w_{ij}) \geq \alpha\}, \quad 1 \leq j \leq R \\ \tilde{c}_{ij}(\alpha) &= \{c_{ij} \in C_{ij} : \mu_{\tilde{C}_{ij}}(c_{ij}) \geq \alpha\}, \quad 1 \leq j \leq R.\end{aligned}$$

and we assume that all  $\alpha$ -cuts are given by the pair of intervals [11]:

$$\begin{aligned}\tilde{w}_{ij}(\alpha) &= [\min_{w_{ij}} \{w_{ij} \in W_{ij} : \mu_{\tilde{W}_{ij}}(w_{ij}) \geq \alpha\}, \\ &\quad \max_{w_{ij}} \{w_{ij} \in W_{ij} : \mu_{\tilde{W}_{ij}}(w_{ij}) \geq \alpha\}] \\ &= [w_{ij}(\alpha)^L, w_{ij}(\alpha)^U] \\ \tilde{c}_{ij}(\alpha) &= [\min_{c_{ij}} \{c_{ij} \in C_{ij} : \mu_{\tilde{C}_{ij}}(c_{ij}) \geq \alpha\}, \\ &\quad \max_{c_{ij}} \{c_{ij} \in C_{ij} : \mu_{\tilde{C}_{ij}}(c_{ij}) \geq \alpha\}] \\ &= [c_{ij}(\alpha)^L, c_{ij}(\alpha)^U]\end{aligned}$$

Kao and Liu proposed to solve the following linear fractional programming problems:

$$\begin{aligned}y_i(\alpha)^L &= \min y_i = \frac{\sum_{j=1}^R w_{ij} c_{ij}(\alpha)}{\sum_{j=1}^R w_{ij}} \\ \text{s. t. } &w_{ij}(\alpha)^L \leq w_{ij} \leq w_{ij}(\alpha)^U, \\ &j = 1, \dots, R, \\ &c_{ij}(\alpha)^L \leq c_{ij} \leq c_{ij}(\alpha)^U, \\ &j = 1, \dots, R,\end{aligned}\tag{13}$$

$$\begin{aligned}y_i(\alpha)^U &= \max y_i = \frac{\sum_{j=1}^R w_{ij} c_{ij}(\alpha)}{\sum_{j=1}^R w_{ij}} \\ \text{s. t. } &w_{ij}(\alpha)^L \leq w_{ij} \leq w_{ij}(\alpha)^U, \\ &j = 1, \dots, R, \\ &c_{ij}(\alpha)^L \leq c_{ij} \leq c_{ij}(\alpha)^U, \\ &j = 1, \dots, R,\end{aligned}\tag{14}$$

If we make the change  $t_j = 1 / \sum_{j=1}^R w_{ij}$ ,  $v_{ij} = t_j w_{ij}$ ,  $j = 1, \dots, R$ , the previous models are transformed in the following linear models:

$$\begin{aligned}y_i(\alpha)^L &= \min y_i = \sum_{j=1}^R v_{ij} c_{ij}(\alpha) \\ \text{s. t. } &t_j w_{ij}(\alpha)^L \leq v_{ij} \leq t_j w_{ij}(\alpha)^U, \\ &j = 1, \dots, R, \\ &\sum_{j=1}^R v_{ij} = 1, \\ &t_j \geq 0,\end{aligned}\tag{15}$$

$$\begin{aligned}y_i(\alpha)^U &= \min y_i = \sum_{j=1}^R v_{ij} c_{ij}(\alpha)^U \\ \text{s. t. } &t_j w_{ij}(\alpha)^L \leq v_{ij} \leq t_j w_{ij}(\alpha)^U, \\ &j = 1, \dots, R, \\ &\sum_{j=1}^R v_{ij} = 1, \\ &t_j \geq 0.\end{aligned}\tag{16}$$

The  $\alpha$ -cut of  $\tilde{Y}_i$  is the crisp interval

$$E_i(\alpha) := [y_i(\alpha)^L, y_i(\alpha)^U],\tag{17}$$

obtained from the previous model. The membership function  $\mu_{\tilde{Y}_i}$  can be constructed by enumerating different  $\alpha$  values.

Expressed in an algorithmic form, our proposal is the following:

ALGORITHM 2																															
STEP 1: State membership functions for competence valuations and weights.																															
STEP 2: Calculate the points $y_j(\alpha)^L$ , $y_j(\alpha)^U$ for every candidate and for $\alpha \in \{0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1\}$																															
<table border="1"> <thead> <tr> <th><math>\alpha</math></th> <th><math>P_1</math></th> <th><math>\dots</math></th> <th><math>P_n</math></th> </tr> </thead> <tbody> <tr> <td>0</td> <td><math>[y_1(0)^L, y_1(0)^U]</math></td> <td><math>\dots</math></td> <td><math>[y_n(0)^L, y_n(0)^U]</math></td> </tr> <tr> <td>0.1</td> <td><math>[y_1(0.1)^L, y_1(0.1)^U]</math></td> <td><math>\dots</math></td> <td><math>[y_n(0.1)^L, y_n(0.1)^U]</math></td> </tr> <tr> <td>0.2</td> <td><math>[y_1(0.2)^L, y_1(0.2)^U]</math></td> <td><math>\dots</math></td> <td><math>[y_n(0.2)^L, y_n(0.2)^U]</math></td> </tr> <tr> <td><math>\vdots</math></td> <td><math>\vdots</math></td> <td><math>\vdots</math></td> <td><math>\vdots</math></td> </tr> <tr> <td>0.9</td> <td><math>[y_1(0.9)^L, y_1(0.9)^U]</math></td> <td><math>\dots</math></td> <td><math>[y_n(0.9)^L, y_n(0.9)^U]</math></td> </tr> <tr> <td>1</td> <td><math>[y_1(1)^L, y_1(1)^U]</math></td> <td><math>\dots</math></td> <td><math>[y_n(1)^L, y_n(1)^U]</math></td> </tr> </tbody> </table>				$\alpha$	$P_1$	$\dots$	$P_n$	0	$[y_1(0)^L, y_1(0)^U]$	$\dots$	$[y_n(0)^L, y_n(0)^U]$	0.1	$[y_1(0.1)^L, y_1(0.1)^U]$	$\dots$	$[y_n(0.1)^L, y_n(0.1)^U]$	0.2	$[y_1(0.2)^L, y_1(0.2)^U]$	$\dots$	$[y_n(0.2)^L, y_n(0.2)^U]$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	0.9	$[y_1(0.9)^L, y_1(0.9)^U]$	$\dots$	$[y_n(0.9)^L, y_n(0.9)^U]$	1	$[y_1(1)^L, y_1(1)^U]$	$\dots$	$[y_n(1)^L, y_n(1)^U]$
$\alpha$	$P_1$	$\dots$	$P_n$																												
0	$[y_1(0)^L, y_1(0)^U]$	$\dots$	$[y_n(0)^L, y_n(0)^U]$																												
0.1	$[y_1(0.1)^L, y_1(0.1)^U]$	$\dots$	$[y_n(0.1)^L, y_n(0.1)^U]$																												
0.2	$[y_1(0.2)^L, y_1(0.2)^U]$	$\dots$	$[y_n(0.2)^L, y_n(0.2)^U]$																												
$\vdots$	$\vdots$	$\vdots$	$\vdots$																												
0.9	$[y_1(0.9)^L, y_1(0.9)^U]$	$\dots$	$[y_n(0.9)^L, y_n(0.9)^U]$																												
1	$[y_1(1)^L, y_1(1)^U]$	$\dots$	$[y_n(1)^L, y_n(1)^U]$																												
STEP 3: Order the intervals for every value of $\alpha$																															
$E_i(\alpha) = [y_i(\alpha)^L, y_i(\alpha)^U]$																															
$E_{k_1}(\alpha), \prec E_{k_2}(\alpha), \prec \dots, \prec E_{k_n}(\alpha)$																															
STEP 4: Select $E_{k_n}(\alpha)$ .																															

## V. COMPUTATIONAL RESULTS

Suppose that we have five candidates,  $P_1, \dots, P_5$  and we must choose just one of them. For this aim we have considered six competences,  $c_1, \dots, c_6$ . Let us show how the proposed models perform.

### A. Example 1: OWA techniques.

Let us assume that the valuation of the competences of every candidate are those appearing in TABLE III.

TABLE III  
VALUATION OF THE COMPETENCES OF EACH CANDIDATE

	$c_1$	$c_2$	$c_3$	$c_4$	
$P_1$	[0.3, 0.7]	0.1	[0.3, 0.6]	[0.4, 0.9]	
$P_2$	[0.1, 0.3]	[0.7, 0.9]	0	[0.6, 0.8]	
$P_3$	0.3	[0.6, 0.9]	[0.4, 0.7]	[0.8, 0.9]	
$P_4$	[0.4, 0.6]	[0.3, 0.5]	[0.4, 0.8]	[0.5, 0.7]	
$P_5$	[0.2, 0.5]	[0.7, 1]	[0.4, 0.7]	[0.6, 0.9]	
	$c_5$	$c_6$		GEV	
$P_1$	[0.5, 0.8]	[0.8, 0.9]		0.6	
$P_2$	[0.4, 0.7]	0.8		0.5	
$P_3$	[0.3, 0.7]	[0.5, 0.7]		0.7	
$P_4$	1	[0.3, 0.7]			
$P_5$	[0.4, 0.6]	0.8			

### A. 1 Assuming trade-off

TABLE IV  
RESULTS OF (OW2) MODEL

$\alpha^*$	$w_1^*$	$w_2^*$	$w_3^*$	$w_4^*$	$w_5^*$	$w_6^*$
0.87	0.4277	0	0	0.0866	0.2513	0.2342

If we solve the (Ow2) model by using data from TABLE III, we obtain the results:

TABLE V  
GEV RESULTS

Candidate	Valuation
$P_4$	0.7392
$P_5$	0.7205

Therefore, the order of the valuations of the candidates is

$$P_2 \prec P_1 \prec P_3 \prec P_5 \prec P_4,$$

and consequently we select the candidate  $P_4$ .

### A. 2 Without trade-off

Firstly, according to expression (9) we choose a representant of every interval,

$$A = \begin{bmatrix} 0.53 & 0.1 & 0.47 & 0.69 & 0.67 & 0.86 \\ 0.22 & 0.82 & 0 & 0.72 & 0.57 & 0.8 \\ 0.3 & 0.77 & 0.57 & 0.86 & 0.53 & 0.62 \\ 0.52 & 0.42 & 0.63 & 0.62 & 1 & 0.53 \\ 0.37 & 0.87 & 0.57 & 0.77 & 0.52 & 0.8 \end{bmatrix}$$

The optimal solution of the corresponding quadratic problem (OW) is

$$w_2 = w_3 = w_6 = 0, w_1 = 0.40, w_4 = 0.28, w_5 = 0.32$$

and the optimal value of the objective function is 0.0009259. Then, we can imitate the valuations made by the experts and evaluate all the candidates, GEV, as we see in TABLE VI.

TABLE VI  
GEV RESULTS

Candidate	Valuation
$P_4$	0.7148
$P_5$	0.6766

Finally, we order the valuations of the candidates,

$$P_2 \prec P_1 \prec P_5 \prec P_3 \prec P_4.$$

We can conclude that candidate  $P_4$  is the most prepared to do the job.

### B. Example 2: Parametric weights.

Let us suppose that the fuzzy weights associated to each competence  $\tilde{W}_1, \dots, \tilde{W}_6$  are triangular fuzzy numbers whose  $\alpha$ -cuts are as follows:

$$\begin{aligned} w_1 &= [-0.25(1-\alpha), 0.1 + 0.15(1-\alpha)] \\ w_2 &= [-0.3(1-\alpha), 0.1 + 0.2(1-\alpha)] \\ w_3 &= [-0.1(1-\alpha), 0.1 + 0.35(1-\alpha)] \\ w_4 &= [0.3 - 0.2(1-\alpha), 0.8 + 0.1(1-\alpha)] \\ w_5 &= [0.1 - 0.05(1-\alpha), 0.15 + 0.1(1-\alpha)] \\ w_6 &= [0.1 - 0.1(1-\alpha), 0.2 + 0.2(1-\alpha)] \end{aligned} \quad (18)$$

Besides, if we assume that the membership functions of the competences are also triangular fuzzy numbers, according to TABLE II, the parametric valuation of competences appears in TABLE VII.

We use GAMS or MS Excel to calculate the intervals that define the objective function depending on the parameter values from 0 to 1.

After choosing a tolerance level, we obtain a set of intervals, one for each candidate, as we see in TABLE VIII. In case that we are interested in defuzzifying the

TABLE VII  
PARAMETRIC VALUATION OF COMPETENCES

	$c_1$	$c_2$	$c_3$
$P_1$	[0.3-0.1(1- $\alpha$ ), 0.7+0.15(1- $\alpha$ )]	0.1	[0.3-0.2(1- $\alpha$ ), 0.6+0.15(1- $\alpha$ )]
$P_2$	[0.1-0.05(1- $\alpha$ ), 0.3+0.3(1- $\alpha$ )]	[0.7-0.25(1- $\alpha$ ), 0.9+0.05(1- $\alpha$ )]	0
$P_3$	0.3	[0.6-0.3(1- $\alpha$ ), 0.9+0.1(1- $\alpha$ )]	[0.4-0.2(1- $\alpha$ ), 0.7+0.2(1- $\alpha$ )]
$P_4$	[0.4-0.05(1- $\alpha$ ), 0.6+0.2(1- $\alpha$ )]	[0.3-0.3(1- $\alpha$ ), 0.5+0.25(1- $\alpha$ )]	[0.4-0.3(1- $\alpha$ ), 0.8+0.1(1- $\alpha$ )]
$P_5$	[0.2-0.2(1- $\alpha$ ), 0.5+0.3(1- $\alpha$ )]	[0.7-0.3(1- $\alpha$ ), 1]	[0.4-0.1(1- $\alpha$ ), 0.7+0.2(1- $\alpha$ )]
	$c_4$	$c_5$	$c_6$
$P_1$	[0.4-0.15(1- $\alpha$ ), 0.9+0.1(1- $\alpha$ )]	[0.5-0.25(1- $\alpha$ ), 0.8+0.1(1- $\alpha$ )]	[0.8-0.3(1- $\alpha$ ), 0.9+0.1(1- $\alpha$ )]
$P_2$	[0.6-0.35(1- $\alpha$ ), 0.8+0.15(1- $\alpha$ )]	[0.4-0.2(1- $\alpha$ ), 0.7+0.1(1- $\alpha$ )]	0.8
$P_3$	[0.8-0.1(1- $\alpha$ ), 0.9+0.1(1- $\alpha$ )]	[0.3-0.25(1- $\alpha$ ), 0.7+0.3(1- $\alpha$ )]	[0.5-0.15(1- $\alpha$ ), 0.7+0.15(1- $\alpha$ )]
$P_4$	[0.5-0.35(1- $\alpha$ ), 0.7+0.25(1- $\alpha$ )]	1	[0.3-0.1(1- $\alpha$ ), 0.7+0.3(1- $\alpha$ )]
$P_5$	[0.6-0.3(1- $\alpha$ ), 0.9+0.05(1- $\alpha$ )]	[0.4-0.3(1- $\alpha$ ), 0.6+0.2(1- $\alpha$ )]	0.8

TABLE VIII  
RESULTS

$\alpha$	$P_1$	$P_2$	$P_3$
0	[0.125, 0.9963]	[0.0559, 0.944]	[0.2156, 1]
0.1	[0.1478, 0.9858]	[0.0805, 0.9297]	[0.2524, 0.9879]
0.2	[0.1716, 0.9754]	[0.1078, 0.9154]	[0.2891, 0.9754]
0.3	[0.1967, 0.9649]	[0.1379, 0.9011]	[0.3202, 0.9627]
0.4	[0.223, 0.9544]	[0.1710, 0.8867]	[0.3494, 0.9496]
0.5	[0.2506, 0.9439]	[0.2073, 0.8723]	[0.3792, 0.9362]
0.6	[0.2766, 0.9333]	[0.2468, 0.8579]	[0.4097, 0.9224]
0.7	[0.3043, 0.9228]	[0.29, 0.8435]	[0.4409, 0.9083]
0.8	[0.3339, 0.9122]	[0.3368, 0.829]	[0.4725, 0.8938]
0.9	[0.3657, 0.9016]	[0.3871, 0.8145]	[0.503, 0.8789]
1	[0.4, 0.8909]	[0.44, 0.8]	[0.5353, 0.8636]
$\alpha$	$P_4$	$P_5$	
0	[0.1222, 0.9933]	[0.075, 0.9667]	
0.1	[0.161, 0.9749]	[0.1287, 0.9555]	
0.2	[0.1995, 0.961]	[0.1744, 0.9442]	
0.3	[0.2358, 0.9446]	[0.2193, 0.9341]	
0.4	[0.2669, 0.926]	[0.2639, 0.9262]	
0.5	[0.2988, 0.9056]	[0.308, 0.9181]	
0.6	[0.3316, 0.8834]	[0.3517, 0.9097]	
0.7	[0.364, 0.8598]	[0.389, 0.9009]	
0.8	[0.3991, 0.8349]	[0.4259, 0.8919]	
0.9	[0.4319, 0.8091]	[0.4651, 0.8825]	
1	[0.4667, 0.7846]	[0.5067, 0.8727]	

results we can compute the midpoints of the intervals, see TABLE IX for  $\alpha = 0.5$  and  $\alpha = 1$ .

The greatest average values correspond to candidate  $P_3$  in both cases. So, he or she is chosen to fill the vacancy. Although the results are the same, this is not always true as it depends on the tolerance level.

TABLE IX  
RESULTS FOR A GIVEN TOLERANCE LEVEL

$\alpha$	$P_1$	$P_2$	$P_3$	$P_4$	$P_5$
0.5	0.59725	0.5398	0.6577	0.6022	0.61305
1	0.64545	0.62	0.69945	0.62565	0.6897

Notice that by using this model the result can differ from the one obtained by using OWA operators, described in the previous section.

## VI. CONCLUSIONS

When mathematical models help decision-making there are some advantages such as quick and clear solutions that are easy to understand. On the other hand, difficulties appear because, in a general way, mathematical models make magnitudes to become objective and quantified. To avoid this, we use models developed in the fuzzy set theory, to add uncertainty and subjectivity to the problem. Representing a phenomenon that occurs in real life without any deformation is a difficult task. Fuzzy logic does not increase the difficulty of traditional mathematics and it is closer to human thinking. Also, it allows thinking on future policies to avoid the rigidity requirements that makes the model non-sense and prevents us from ignoring other solutions that could be useful.

In personnel selection, an inflexible treatment of the candidate valuations can obstruct the ordering process because not all the requirements are considered, while global valuation neutralizes the positive valuation of competences with the negatives, and is unfair, etc. In this paper we show a way to simulate the experts valuation because in a personnel selection that involves many candidates an evaluation process performed by external experts would be too long and expensive. The techniques explained can be used in other scenarios: hiring, training, promotion, etc., and also can be modelled by means of fuzzy sets. In this framework, fuzzy mathematical methods are a powerful tool for the decision-making process.

OWA operators provide flexibility to model a wide variety of aggregators because of its definition is based in a vector of weights and not in a unique parameter. A different proposal is to assume that the weights are

functions depending on a parameter  $\alpha$  that represents the satisfaction level. We can calculate intervals associated with weights for every  $\alpha$  by means of an efficiency analysis. Therefore, in a personnel selection process we get an order of the candidates to fill a vacancy.

Finally, it is worth remarking that the described models can be solved easily by using MS Excel, this is an added advantage because the program is almost universally available.

#### ACKNOWLEDGEMENTS

The authors would like to thank the referees for their comments and suggestions.

This paper has been partially supported by the Ministerio de Ciencia y Tecnología of Spain, TIC2002-04242-C03-03, and Generalitat Valenciana, GV04B-090.

#### REFERENCES

- [1] J.E. Butler, G.R. Ferris and N.K. Napier, *Strategy and Human Resources Management*. South Western Publishing CO, 1991.
- [2] T. Calvo and R. Mesiar, "Weighted triangular norms-based aggregation operators". *Fuzzy Sets and Systems*, 137, pp. 3-10, 2003.
- [3] L. Canós and V. Liern, "Some fuzzy models for human resource management". *International Journal of Technology, Policy and Management*, 4, pp. 291-308, 2004.
- [4] C.H. Carlsson and R. Fullér, *Fuzzy reasoning in decision making and optimization*. Heidelberg: Springer-Verlag, 2002.
- [5] C.H. Carlsson and P. Korhonen, "A parametric approach to fuzzy linear programming". *Fuzzy Sets and Systems*, 20, pp. 17-30, 1986.
- [6] F. Chiclana, F. Herrera, E. Herrera-Viedma and L. Martínez, "A note on the reciprocity in the aggregation of fuzzy preference relations using OWA operators". *Fuzzy Sets and Systems*, 137, pp. 71-83, 2003.
- [7] D. Dubois and H. Prade, *Fuzzy sets and systems: theory and applications*. Academic Press. San Diego, 1980.
- [8] D.P. Filev and R.R. Yager, "On the issue of obtaining OWA operator weights". *Fuzzy Sets and Systems*, 94, pp. 157-169, 1998.
- [9] J.C. Fodor, J.L. Marichal and M. Roubens, "Characterization of some aggregation functions arising from MCDM problems" en B. Bouchon-Meunier, R.R. Yager y L.A. Zadeh (eds.), *Fuzzy Logic and Soft Computing*. Series: Advances in Fuzzy Systems-Applications and Theory, vol. 4 (World Scientific Publishing Singapore), pp. 194-201, 1995.
- [10] E. Jacquet-Lagrèze and Y. Siskos, "Preference disaggregation: 20 years of MCDA experience". *European Journal of Operational Research*, 130, pp. 233-245, 2001.
- [11] C. Kao and S.T. Liu, "Fraccional programming approach to fuzzy weighted average". *Fuzzy Sets and Systems*, 120, pp. 435-444, 2001.
- [12] U. Kaymak and J.M. Sousa, "Weighted constraint aggregation in fuzzy optimization". *Constraints*, 8, (1), pp. 29-46, 2001.
- [13] H. Legind Larsen, "Efficient importance weighted aggregation between min and max". *9th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems* (IPMU' 2002). Annecy (Francia), 2002.
- [14] R.A. Marques Pereira and R. Almeida Ribeiro, "Aggregation with generalized mixtura operators using weighting functions". *Fuzzy Sets and Systems*, 137, pp. 43-58, 2003.
- [15] R. Smolková and M.P. Wachowiak, "Aggregation operators for selection problems". *Fuzzy Sets and Systems*, 131, pp. 23-34, 2002.
- [16] A. Spyridakos, Y. Siskos, D. Yannacopoulos and A. Skouris, "Multicriteria job evaluation for large organizations". *European Journal of Operational Research*, 130, pp. 375-387, 2001.
- [17] R.R. Yager, "On ordered weighted averaging aggregation operators in multi-criteria decision making". *IEEE Transactions on Systems, Man and Cybernetics*, 18, pp. 183-190, 1988.
- [18] L.A. Zadeh, "Fuzzy sets". *Information and Control*, 8, pp. 338-353, 1965.
- [19] H.J. Zimmermann, *Fuzzy Set Theory and its Applications*. Kluwer Academic Publishers. Boston, 1996.

# Cutting Plane and Column Generation for the Capacitated Arc Routing Problem

David Gómez-Cabrero<sup>\*</sup>, José Manuel Belenguer<sup>†</sup> and Enrique Benavent<sup>‡</sup>

<sup>\*</sup> Departamento de Estadística e Investigación Operativa / Universitat de València  
Dr. Moliner 50, 46100 Burjassot, Valencia, Spain  
Email: lunacab@yahoo.es

<sup>†</sup> Departamento de Estadística e Investigación Operativa / Universitat de València  
Dr. Moliner 50, 46100 Burjassot, Valencia, Spain  
Email: jose.belenguer@uv.es

<sup>‡</sup> Departamento de Estadística e Investigación Operativa / Universitat de València  
Dr. Moliner 50, 46100 Burjassot, Valencia, Spain  
Email: enrique.benavent@uv.es

**Abstract**— This paper presents a new procedure for computing lower bounds for the Capacitated Arc Routing Problem (CARP). We propose a Set Covering model with additional constraints whose linear relaxation is solved by combining column generation with a cutting plane algorithm. The set of columns is relaxed as to include the so called q-routes that can be efficiently generated by means of a dynamic programming algorithm. The additional constraints and the cutting plane algorithm have been adapted from the one by Belenguer and Benavent. This new method outperforms the pure cutting plane algorithm of Belenguer and Benavent and is competitive with the best lower bounding procedures for the CARP.

**Keywords**— Capacitated Arc Routing Problem, Column Generation, Cutting Plane.

## I. INTRODUCTION.

THE Capacitated Arc Routing Problem (CARP), introduced by Golden and Wong [9], can be defined as follows. Let  $G'=(V,E)$  be an undirected graph with vertex set  $V$  and edge set  $E$ . Every edge  $e \in E$  has associated a cost  $c_e \geq 0$  and a demand  $q_e \geq 0$ . We denote by  $E_R$  the set of edges with positive demand, which are called required edges.

Given a vehicle capacity  $Q$ , the CARP consists of finding a set of vehicle routes of minimum cost, such that every required edge is served by exactly one vehicle, each route starts and ends at a depot (assumed to be vertex 1) and the total demand served by a route does not exceed  $Q$ .

The CARP has been used to model many real situations, including garbage collection, postal

delivery, snow cleaning, routing of electric meter readers, etc. By this reason, several heuristics have been proposed for it; see Hertz and Mittaz [11] for an exhaustive review. The best solutions have been obtained with metaheuristic algorithms like the tabu search proposed by Hertz et al. [10] and the memetic algorithm proposed by Lacomme et al. [14].

The CARP is NP-hard (see [9]) and it is considered to be very difficult to solve exactly. Several lower bounding procedures for the CARP, based in matching problems, have been proposed: Golden and Wong [9], Assad et al. [2], Pearn [17], Benavent et al. [5], Hirabayashi et al. [12], Amberg and Voß [1], and Wohlk [18]. All these procedures are outperformed by the cutting plane algorithm proposed by Belenguer and Benavent [4]. The first exact method proposed was a Branch & Bound algorithm described in Kiuchi et al. [13] where, at each node, a lower bound is computed by the procedure described in [12]. Only small instances have been solved with this method. Recently, Longo et al. [15] and Baldacci and Maniezzo [3] transform the CARP into a Capacitated Vehicle Routing Problem (CVRP). Then, [15] solves the transformed instance with a Branch-and-Cut-and-Price algorithm, while [3] uses Branch-and-Cut. They are able to solve instances with up to 98 required edges. The lower bound obtained in the root node by these methods outperforms all the above mentioned lower bounding procedures.

See also Dror [6] for an exhaustive review of the CARP and other arc routing problems.

The approach proposed in this work considers a Set Covering formulation of the CARP, and then a column generation approach is used where routes are generated and added to the model as needed. The model is strengthened by adding several classes of valid constraints for the CARP that are dynamically included in the model using a cutting plane procedure.

In section II we present the Set Covering model for the CARP that includes additional constraints. In the next section we present the whole algorithm that combines Column and Cutting Plane Generation. We explain there the dynamic programming algorithm used to generate new columns, and the valid constraints and their corresponding identification procedures. Afterwards, in section IV, we present the computational results obtained by our method. Finally, section V draws some conclusions and proposes future research.

## II. THE SET COVERING MODEL

We use a Set Covering model to formulate the CARP: if we consider the set of feasible routes for the vehicles the CARP can be viewed as the problem of selecting a set of routes with minimum total cost that serve all the required edges. One drawback of this model is the huge number of feasible routes that usually exists, so we use a column generation approach where routes are generated and added to the model as needed. Unfortunately, in order to generate appropriate routes, an NP-hard problem has to be solved; we avoid this problem by relaxing the set of routes and including the so called q-routes, which allows generating columns more efficiently. Similar approaches are quite known in other routing problems like the Vehicle Routing Problem with Time Windows (VRPTW), see for instance Desrosiers et al. [7], but have not been applied successfully to the CARP, up to our knowledge. What is less usual is the use of additional constraints that allow to strengthen the model. We use several classes of valid constraints for the CARP that are dynamically included in the model using a cutting plane procedure.

Let  $V$  be the subset of vertices that includes the depot and those vertices which are incident with at least one required edge. We denote by  $s_{ij}$  to the cost of the shortest path cost in  $G'$  between

vertices  $i$  and  $j$ ,  $i,j \in V$ . Our model works on the transformed graph  $G = (V, E_R \cup H)$  where  $H$  contains a non required edge  $e=(i,j)$  with cost  $s_{ij}$  (also denoted  $s_e$ ), for each pair of vertices  $i,j \in V$ .

We call q-route to a closed walk in  $G$  containing required edges, that are served, and edges from  $H$ , that are just traversed, and satisfying the following constraints:

- it contains the depot,
- it may serve more than once the same required edge,
- the total demand it serves is not greater than  $Q$ ,
- it cannot traverse two edges from  $H$  consecutively.

It is obvious that the set of q-routes contains all routes that can be part of an optimal solution to the CARP. The prohibition of consecutively traversing two edges of  $H$  only eliminates dominated routes as the costs of these edges satisfy  $s_{ij} \leq s_{ik} + s_{kj}$ , for any  $i,j,k \in V$ . This prohibition is justified to avoid q-routes with unbounded lower cost, which could appear when transformed costs (to be defined later on) are negative.

Let  $\Omega$  denote the set of q-routes. We use the following decision variables, for any  $i \in \Omega$ :

$$x_i = \begin{cases} 1 & \text{if q-route } i \text{ is used} \\ 0 & \text{if q-route } i \text{ is not used} \end{cases}$$

We will denote by:

- $a_{ie} =$  number of times q-route  $i \in \Omega$  serves edge  $e \in E_R$ ,
- $b_{ie} =$  number of times q-route  $i \in \Omega$  traverses edge  $e \in H$ ,
- $C(i) =$  total cost of q-route  $i \in \Omega$ .

As we will see, the CARP can be formulated as a Set Covering Problem with these variables. On the other hand, Belenguer and Benavent [4] introduced a set of valid constraints for the CARP using the integer variables  $z_e$  that represent the number of times that the non required edge  $e \in H$  is traversed by the set of routes. Let us denote by  $W$  the set of valid constraints we want to use in our model. Given  $r \in W$ , we denote by  $\alpha_{re}$  the coefficient of  $z_e$ ,  $e \in H$ , in constraint  $r$ , and by  $\beta_r$  the corresponding Right-Hand-Side; then constraint  $r \in W$  can be written:

$$\sum_{e \in H} \alpha_{re} z_e \geq \beta_r$$

These constraints can be used in the Set Covering model because:

$$z_e = \sum_{i \in \Omega} b_{ie} x_i. \quad (1)$$

Then, the CARP can be formulated as the following Set Covering Problem with additional constraints:

$$\text{Min} \sum_{i \in \Omega} C(i)x_i \quad (2)$$

$$\sum_{i \in \Omega} a_{ie} x_i \geq 1 \quad \forall e \in E_R \quad (3)$$

$$\sum_{i \in \Omega} \sum_{e \in H} b_{ie} \alpha_{re} x_i \geq \beta_r \quad r \in W \quad (4)$$

$$0 \leq x_i \leq 1 \quad \forall i \in \Omega \quad (5)$$

$$x_i \in \{0,1\} \quad \forall i \in \Omega \quad (6)$$

Constraints (3) assure that all the required edges are served by the selected q-routes. It is easy to see that there always exists an optimal solution using only routes that serve all the required edges exactly once. The additional constraints (4) are redundant in this program but are very useful in order to strengthen the linear relaxation of it. The linear program defined by (2), ..., (5) will be called Master Program (MP) and its optimal cost is a valid lower bound for the CARP. This problem contains a huge number of variables (columns) and constraints (cutting planes), then we use a combined method that generates columns and cutting planes only when they are needed thus keeping the size of the linear program (LP) inside reasonable limits. We call the Restricted Master Program (RMP) to the problem that results from the MP when only a subset of columns and a subset of constraints (4) are considered.

### III. COMBINING COLUMN AND CUTTING PLANE GENERATION

An initial RMP is built with a subset of columns and constraints (4). Then, at each iteration, the RMP is solved using the simplex method and its optimal dual values are used to define the Subproblem that consists of generating new columns (q-routes) with negative reduced cost that are added to the RMP. If no such q-route is found, we look for constraints (4) that are violated by the LP solution (cutting planes) and, if any is found, they are added to the RMP. If any column or cutting plane has been added to the RMP, it is solved again, and so on. The whole procedure stops when no negative reduced cost q-route and no cutting plane are found for the current LP solution (see Figure I). The following subsections

describe with more detail the main components of the algorithm.

#### A. Subproblem

Given a set of dual values, the role of the Subproblem is either to identify a column (or columns) that have a negative reduced cost or to prove that no such column exists.

*1) Reduced cost of a q-route:* Let us denote by  $\pi_e$  the value of the dual variable associated with constraint (3) corresponding to  $e \in E_R$ , and let us denote by  $\gamma_r$  the value of the dual variable associated with constraint  $r$  in (4) in the optima

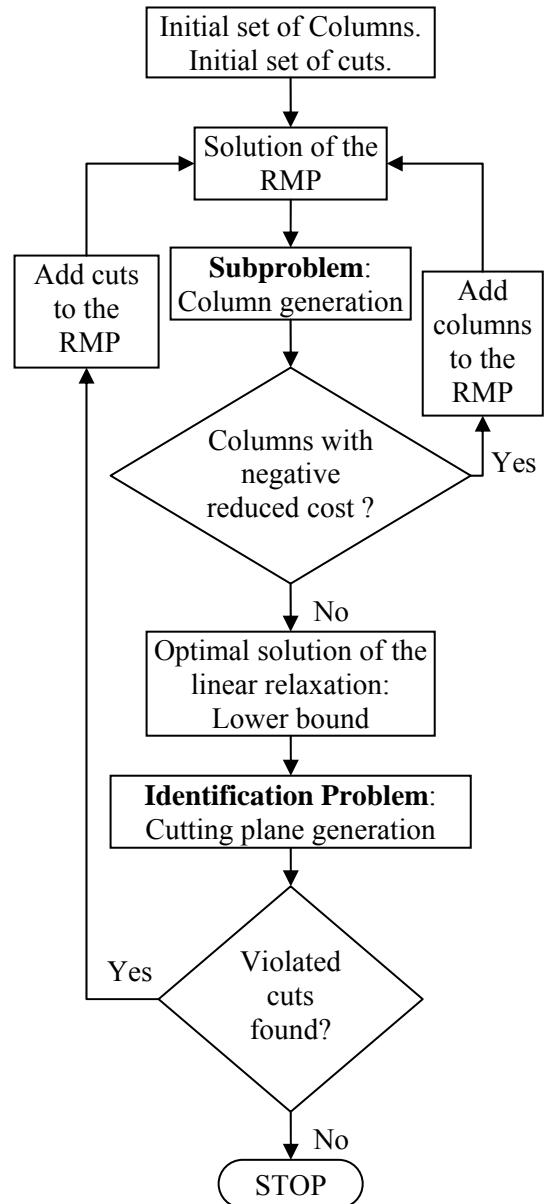


Figure I. Algorithm.

solution of the current RMP. The reduced cost of a q-route  $i \in \Omega$  will be denoted by  $C(i)$ , i.e.:

$$C'(i) = C(i) - \sum_{e \in E_R} \pi_e a_{ie} - \sum_{r \in W} \sum_{e \in H} \gamma_r b_{ie} \alpha_{re}$$

Given that:

$$C(i) = \sum_{e \in E_R} c_e a_{ie} + \sum_{e \in H} s_e b_{ie},$$

it easily follows:

$$C'(i) = \sum_{e \in E_R} (c_e - \pi_e) a_{ie} + \sum_{e \in H} (s_e - \sum_{r \in W} \gamma_r \alpha_{re}) b_{ie}$$

Let us denote by  $\hat{c}_e$   $e \in E_R$  and  $\hat{s}_e$   $e \in H$ , the modified costs:

- $\hat{c}_e = c_e - \pi_e \quad \forall e \in E_R$
- $\hat{s}_e = s_e - \sum_{r \in W} \gamma_r \alpha_{re} \quad \forall e \in H$

Note that the cost of a q-route under these modified costs is in fact its reduced cost. Note also that these modified costs can be negative and the triangular inequality is no longer satisfied by the modified costs of the non required edges.

2) *Minimum cost q-routes*: Let us define a q-chain as an ordered set of vertices and edges  $v_0 e_1 v_1 \dots e_t v_t$  where  $e_i \in E_R \cup H$  is incident to  $v_{i-1}$  and  $v_i$  for  $i = 1, \dots, t$ , and at least one of any two consecutive edges, say  $e_i$  and  $e_{i+1}$ , must be a required edge. The load of a q-chain is defined as the sum of the loads of the required edges it contains. Let us denote by  $V_R$  the set of vertices which are incident with at least one required edge.

Given  $j \in V$  and an integer  $q$ ,  $0 \leq q \leq Q$ , let  $u(j, q)$  be the cost of the least-cost q-chain from the depot to vertex  $j$  with a total load of  $q$ . These costs can be computed for every vertex  $j \in V$  and every load  $q=0, \dots, Q$  with a dynamic programming algorithm that is an adaptation of the one described in Benavent et al. [5]. The interested reader is referred to this paper for a detailed description of the algorithm.

The algorithm initializes  $u(j, 0) = \hat{s}_{ij}$  for all  $j \in V$ . Then, at iteration  $q$ ,  $q=1, \dots, Q$ , the algorithm computes  $u(j, q)$  in two steps. In a first step, only q-chains where the last edge is served are considered. Their corresponding cost is denoted by  $u'(j, q)$  and is computed as follows for every  $j \in V_R$ :

$$u'(j, q) = \min\{u(i, q - q_e) + \hat{c}_e : \forall e = (i, j) \in E_R\}$$

In a second step the algorithm computes  $u(j, q)$  by considering also a non required edge as the possible last edge:

$$u(j, q) =$$

$$\min\{u'(j, q), \min\{u'(i, q) + \hat{s}_{ij} : \forall (i, j) \in H\}\}$$

We avoid in this way q-chains traversing two non required edges consecutively.

Let us denote by  $u_e^q$  the cost of the least cost q-route whose last served edge is  $e = (i, j) \in E_R$ , and with load  $q \in \{1, \dots, Q\}$ . These costs can be computed as follows:

$$u_e^q = \min\{u(i, q - q_e) + \hat{c}_e + \hat{s}_{j1}, u(j, q - q_e) + \hat{c}_e + \hat{s}_{i1}\}$$

Given that the minimum cost q-route will serve at least one edge, its cost will be equal to  $u_e^q$  for some  $e \in E_R$  and  $q \in \{1, \dots, Q\}$ .

The q-routes obtained by the procedure described above may serve edge  $e = (i, j) \in E_R$ , while traversing it from  $i$  to  $j$  and immediately after, serving once more this edge from  $j$  to  $i$ . We denote this behavior as *2-loop*. It is possible to prevent *2-loops*, thus improving the method, without an excessive increment of the computational cost. The method is also described in Benavent et al. [5] and we have also made minimum modifications to allow the use of negative costs. The resulting q-routes will be referred to as q-routes-S2B.

### B. Cutting plane

We have used the capacity constraints and odd cutset constraints, that were introduced by Belenguer and Benavent [4], in order to strengthen our model. Let us introduce some new notation. Given a vertex set  $S \subseteq V \setminus \{1\}$ ,  $H(S)$  will represent the set of edges of  $H$  with both endpoints in  $S$ , while  $\delta(S)$  will denote the edge cutset defined by  $S$ , i.e. the set of edges of  $H$  with one endpoint in  $S$  and the other not in  $S$ . The corresponding sets for required edges are denoted by  $E_R(S)$  and  $\delta_R(S)$  respectively. Recall that the formulation considered in [4] uses a set of variables  $z_e$ ,  $e \in H$ , where  $z_e$  represents the number of times that edge  $e$  has been traversed by the set of vehicles. The capacity constraints are:

$$\sum_{e \in \delta(S)} z_e \geq 2k(S) - |\delta_R(S)|, \text{ with } S \subseteq V \setminus \{1\},$$

where  $k(S)$  is the minimum number of routes necessary for serving all the required edges of  $\delta_R(S) \cup E_R(S)$ . The idea is that all the routes that

will serve all the demand of the edges in  $\delta_R(S) \cup E_R(S)$  will have to pass through the cutset  $\delta(S)$  at least  $2k(S)$  times; then, the cutset will be traversed using edges of  $H$  at least  $2k(S) - |\delta_R(S)|$  times.

The odd cutset constraints, also introduced in [4], are a consequence of the fact that any cutset will be traversed an even number of times in any solution to the CARP:

$$\sum_{e \in \delta(S)} z_e \geq 1, \quad S \subseteq V \setminus \{1\} \text{ with } |\delta_R(S)| \text{ odd}$$

Belenguer and Benavent also introduced the disjoint path constraints but we have not used them in our method. Note that the number of capacity and odd cutset inequalities is exponential in the number of vertices. Instead of including all of them in the RMP, we use a cutting plane procedure in which we identify the constraints that are violated by the LP solution and add them to the RMP. We have used similar procedures to those described in [4] to identify violated odd cutset and capacity constraints. These procedures are mostly heuristic algorithms that have proved to be very effective in finding violated inequalities.

In order to use these identification algorithms, we have to express the current optimal solution of the RMP in terms of the variables  $z_e$ . Thus, for each edge  $e \in H$ , we compute  $z_e$  as the total number of times that the q-routes uses the non required edge  $e$  weighted by the corresponding value of each q-route variable (see (1)).

In our separation procedures we work with two kinds of solution graphs. Let  $G(z, \varepsilon)$  be the graph induced by the edges  $e \in H$  with  $z_e > \varepsilon$ , and let  $G(E_R, z, \varepsilon)$  be the graph induced by the edges  $e \in E_R$  and by the edges  $e \in H$  with  $z_e > \varepsilon$  plus the depot.

The strategy used here for finding violated constraints is the following. Firstly, we check for possible violation of the corresponding capacity and/or odd cutset constraint using the vertex sets of the connected components of  $G(R, z, \varepsilon)$ , for several values of  $\varepsilon = 0, 0.1, 0.2, \dots, 1$ ; and, similarly for  $G(z, \varepsilon)$ , for  $\varepsilon = 0, 1$ . We use also the fractional capacity constraint procedure (see [4] for the details). If one or more violated constraints are found, they are added to the RMP and we stop searching new cuts.

When the previous procedures failed, we call the exact separation procedure of odd cutset constraints which is a polynomial time algorithm

based on the Odd Minimum Cut algorithm of Padberg and Rao [16] (see [4] for a detailed explanation of this procedure).

We have designed a new heuristic that is called when all the preceding procedures fail. The algorithm considers sequences of vertex sets for which the capacity constraint is checked. It is initialized with one vertex, and, at each step, the current vertex set, say  $W$ , is enlarged by adding a new vertex which is adjacent to at least one vertex in  $W$  in the graph induced by the required edges without the depot, until  $W$  cannot be enlarged. The whole procedure is applied starting each time with a different vertex.

### C. Initialization of the algorithm

1) *Initial q-routes*: We have to define an initial set of q-routes assuring that the initial RMP will have a feasible solution. We compute the initial routes using two simple heuristic algorithms.

The first one builds one route for each required edge  $(i, j) \in E_R$  by joining its endpoints to the depot with the corresponding non required edge (recall that the costs of these edges are equal to the shortest path costs).

The second method starts with the routes built in the first method. Afterwards, it modifies the routes by serving as many required edges as possible from those contained in the shortest paths which have been previously broken down into the original edges.

2) *Initial cuts*: The initial RMP contains a reduced set of constraints (4) that are generated as follows:

- an odd cut constraint for every vertex  $v$  that is incident with an odd number of required edges,
- capacity constraints defined by the vertex sets of the connected components of the graph induced by  $E_R$ , that is denoted by  $G(E_R)$  and
- capacity and odd cut constraints for a sequence of vertex sets generated by starting with the depot, and sequentially adding to the set all its neighbour vertices in  $G(E_R)$ .

### D. Selection of q-routes

We have developed several methods for selecting the q-routes to be introduced in the RMP among those generated by the Subproblem with negative reduced cost. We have used several

criteria: to minimize the reduced cost, to maximize the diversity and to give priority to valid routes. The following methods combine these criteria. In all of them, the number of columns that are finally added to the RMP in each iteration is not larger than  $p_c |E_R|$ , where  $0 < p_c \leq 1$  is a parameter.

The first method selects a set of q-routes with minimum reduced cost. One drawback of this method is that the selected q-routes are usually very similar. The second method selects for each required edge the q-route with minimum reduced cost among those for which that edge is the last served edge. We obtain in this way a more heterogeneous set of q-routes. The third method is similar, but we give priority to the q-routes that are valid, i.e. no required edge is served more than once in the q-route. This third method produce the worst results in terms of CPU time and number of iterations in the column generation phase. The second method provides the best results and it is the one used in what follows.

After some testing the value of parameter  $p_c$  was set to 0.15; larger values of  $p_c$  have the effect of reducing the number of iterations of the column generation phase and the CPU time but increases considerably the sizes of the LPs to be solved. Our current implementation does not include the

possibility of removing columns from the RMP, so we run out of memory when we try to solve the biggest instances using  $p_c > 0.15$ .

#### IV. COMPUTATIONAL RESULTS

The algorithm has been implemented using Microsoft Visual C++ 6.0 and run on a Pentium IV at 2 Ghz. Linear problems have been solved with CPLEX 8.0.

We have tested the algorithm with three different sets of instances. The first set, denoted gdb, contains 23 instances generated by Golden et al. [8]. The second one, denoted val, were generated by Benavent et al. [5] and contains 34 instances

defined on 10 different graphs with several vehicle capacities for each one. The third set of instances, denoted egl, was constructed by Belenguer and Benavent [4] from data obtained by Li and Eglese for a study on winter gritting. This last set contains the biggest instances and, contrarily to the first two sets of instances, many of them contain non-required edges and the graph induced by the required edges is usually non-connected. The characteristics of each set of instances are shown in Table I.

The results obtained by our algorithm, using q-routes, on the three sets of instances are

TABLE I.  
CHARACTERISTICS OF THE INSTANCES.

	Number of instances.	Number of vertices	Number of required edges.	Number of non required edges	Minimum number of vehicles.
gdb	23	7-27	11-55	0	3-10
val	34	24-50	34-97	0	2-10
egl	24	77-140	51-190	0-115	5-35

TABLE II.  
RESULTS USING Q-ROUTES.

	dev %	N. of Op.	It. CG	It. CPG	CPU Total
gdb	0.10	21	42.3	2.2	0.4
val	0.42	20	168.2	6.9	29.9
egl	2.45	0	267.7	23.6	2412.9

TABLE III.  
RESULTS USING Q-ROUTES S2B.

	dev %	N. of Op.	It. CG	It. CPG	CPU Total
gdb	0.07	21	62.2	2.4	0.6
val	0.39	21	342.1	6.5	52.0
egl	2.36	0	399.2	18.3	2562.0

summarized in Table II; all the entries show the average results in each set of instances. 'dev %' is a measure of the quality of the lower bound (LB) obtained by this method and it is computed as:  $\text{dev\%} = 100 * (\text{UB-LB})/\text{UB}$ , where UB represents the best upper bound known of the corresponding instance. 'N. of Op.' is the number of instances where the lower bound matches the upper bound, i.e. the optimality is proved. 'It. CG.' denotes the total number iterations of the Column Generation Algorithm (number of times that the Subproblem was solved), 'It. CPG' denotes the number of iterations of the Cutting Plane Algorithm (number of times that the identification procedures were called), and 'CPU Total' denotes the CPU time in seconds.

The results obtained by the algorithm that uses q-routes-S2B are shown in Table III while Table IV shows the percentage of decrease in the deviation 'dev %', and the percentage of increase in the total CPU time when using q-routes-S2B instead of q-routes.

TABLE IV.  
COMPARISON BETWEEN Q-ROUTES  
AND Q-ROUTES-S2B.

	dev %	CPU Total
gdb	-30%	+33.3%
val	-7.14%	+44.2%
egl	-3.67%	+5.82%

Note that using q-route without 2-loops we get a slight improvement in the lower bound that in some cases (like in gdb instances) may imply a substantial decrease in the percentage deviation from the upper bound, with a moderate increase in the CPU time.

TABLE V.  
COMPARISON WITH OTHER LOWER  
BOUNDING PROCEDURES.

	q-routes S2B	q-routes S2B(CG)	CP1	CP2	TNR
gdb	0.07	4.92	0.13	0.13	*
val	0.39	7.21	0.66	0.41	0.32
egl	2.36	*	2.69	2.39	1.45

Table V compares our results with some outstanding methods in the literature. All the entries in that table show the average percentage

deviation of the lower bound obtained with different methods, to the best upper bound known. 'q-routes S2B' denotes our algorithm using q-routes-S2B, 'q-routes S2B(CG)' denotes the same algorithm but without inserting any constraints (4). Next two columns show the results of the cutting plane algorithm of Belenguer and Benavent [4], using only the capacity and odd set constraints (CP1) and using also the disjoint path constraints (CP2); finally, 'TNR' denotes the method by Longo et al. [15] (the method of Baldacci and Maniezzo [3] is very similar).

The first conclusion is that a dramatic improvement is achieved by the addition of the capacity and odd cutset inequalities to the pure Column Generation procedure. We can see also that our algorithm outperforms the cutting plane algorithm of Belenguer and Benavent [4]; it is even better than CP2, despite of the fact that our present algorithm does not include the disjoint path inequalities.

Nevertheless TNR produces better results than our algorithm on val instances and, specially, in the egl instances (the authors did not run their algorithm on the gdb instances). TNR is based on transforming the CRP into a CVRP that is then solved with a method that combines also column generation and the addition of cutting planes. This method gets benefit from the large number of classes of valid inequalities that are known for the CVRP, a problem that has been much more studied than the CARP. We hope that the use of the disjoint path constraints and the development of new classes of valid constraints will benefit our method.

## V. CONCLUSIONS.

We have presented a new algorithm that uses column generation and cutting plane procedures for computing lower bounds for the Capacitated Arc Routing Problem. The algorithm finds competitive lower bounds improving the ones computed by the cutting plane algorithm of Belenguer and Benavent [4], although they are worse than the ones computed by Longo et al. [15] and Baldacci and Maniezzo [3] that use a transformation of the CARP into a Capacitated Vehicle Routing Problem. We plan to include in the near future the disjoint path inequalities from [4] and other new valid inequalities for the CARP in order to strengthen the method.

We have also tested several methods for selecting the new columns to insert in the RMP. The conclusion is that increasing the number of q-routes inserted at each iteration results in a decreasing of the time necessary to compute the lower bound. On the other hand, this causes that the number of columns becomes excessive in the biggest instances. We propose, for a future research, inserting at each iteration a larger number of new columns, and eliminating periodically from the RMP the columns that have not been used in several consecutive iterations, thus keeping the current number of columns to a reasonable limit.

#### ACKNOWLEDGEMENTS

The contribution by J.M. Belenguer, E. Benavent and D. Gómez-Cabrero has been partially supported by the AVCiT of the Generalitat Valenciana (Ref: GRUPOS03/174). The contribution by E. Benavent has been partially supported by the Ministerio de Ciencia y Tecnología of Spain through project TIC2003-05982-C05-01.

#### REFERENCES.

- [1] A. Amberg and S. Voß, "A hierarchical relaxations lower bound for the capacitated arc routing problem", in *Proceedings of the 35<sup>th</sup> Annual Hawaii International Conference on System Sciences*, 2002.
- [2] A.A. Assad, W.L. Pearn and B.L. Golden, "The Capacitated Chinese Postman Problem: Lower Bounds and Solvable Cases", *American Journal of Mathematical and Management Sciences* 7, pp. 63—88, 1987.
- [3] R. Baldacci and V. Maniezzo, "Exact methods based on node routing formulations for Arc Routing Problems", Technical report UBLCS-2004-10, University of Bolonia, 2004.
- [4] J.M. Belenguer and E. Benavent, "Cutting Plane Algorithm for the Capacitated Arc Routing Problem", *Computers and Operations Research* 30(6), pp. 705-728, 2003.
- [5] E. Benavent, V. Campos, A. Corberán and E. Mota, "The Capacitated Chinese Postman Problem: Lower Bounds". *Networks* 22, pp. 669-690, 1992.
- [6] M. Dror, "Arc Routing. Theory, solutions and applications". Boston: Kluwer Academic Publishers, 2000
- [7] J. Desrosiers, F. Soumis and M. Desrochers, "Routing with Time Windows by Column Generation". *Networks* 14, pp. 545-565, 1984.
- [8] B.L. Golden, J.S. DeArmon and E.K. Baker, "Computational Experiments with Algorithms for a Class of Routing Problems". *Computers and Operations Research* 10 (2), pp. 47-59, 1983.
- [9] B.L. Golden and R. Wong, "Capacitated Arc Routing Problems". *Networks* 11, pp. 305-315, 1981.
- [10] A. Hertz, G. Laporte and M. Mittaz, "A Tabu Search Heuristic for the Capacitated Arc Routing Problem". *Operations Research* 48 (2), pp. 129-135, 2000.
- [11] A. Hertz and M. Mittaz, "Heuristic algorithms", in *Arc Routing. Theory, solutions and applications* M. Dror, Ed., Boston: Kluwer Academic Publishers, pp. 327-386, 2000.
- [12] R. Hirabayashi, Y. Saruwatari and N. Nishida, "Tour Construction Algorithm for the Capacitated Arc Routing Problem". *Asia-Pacific Journal of Operational Research* 9 (3), pp. 155-175, 1992.
- [13] M. Kiuchi, Y. Shinano, R. Hirabayashi and Y. Saruwatari, "An exact algorithm for the Capacitated Arc Routing Problem using parallel Branch and Bound method", *Abstracts of the 1995 Spring National Conference of the Operational Research Society of Japan*, pp. 28-29, 1995.
- [14] P. Lacomme, C. Prins and W. Ramdane-Chérib, "Competitive memetic algorithms for arc routing problems", *Annals of Operations Research*, 131, pp. 159-185, 2004.
- [15] H. Longo, M. Poggi de Aragão and E. Uchoa, "Solving Capacitated Arc Routing Problems using a transformation to the CVRP", Technical Report PUC-RioInf.MCC10/04, Pontifícia Universidade Católica do Rio de Janeiro, 2004.
- [16] MW Padberg and MR Rao, "Odd minimum cut-sets and b-matching", *Mathematics of Operation Research* 7; pp. 67-80, 1982.
- [17] W.L. Pearn, "New lower Bounds for the Capacitated Arc Routing Problem". *Networks*, 18, pp. 181-191, 1988.
- [18] S. Wohlk "New Lower Bound for the Capacitated Arc Routing Problem". Preprint PP-2003-15, University of Southern Denmark, IMADA – Department of Mathematics and Computer Science, 2003.

# GRASP and Path Relinking for Project Scheduling under Partially Renewable Resources

F. Villa\*, R. Alvarez-Valdes<sup>†</sup>, E. Crespo<sup>‡</sup> and J.M. Tamarit<sup>†</sup>

\*Florida Universitaria, Valencia, Spain

Email: fvilla@florida-uni.es

<sup>†</sup>Department of Statistics and Operations Research

University of Valencia, Spain

Email: ramon.alvarez@uv.es, jose.tamarit@uv.es

<sup>‡</sup>Department of Mathematics for Economics and Business

University of Valencia, Spain

Email: enric.crespo@uv.es

**Abstract**— Recently, in the field of project scheduling problems the concept of partially renewable resources has been introduced. Theoretically, it is a generalization of both renewable and non-renewable resources. From an applied point of view, partially renewable resources allow us to model a large variety of situations that do not fit in classical models, but can be found in real problems in timetabling and labor scheduling. In this paper we develop some preprocessing techniques and several heuristic algorithms for the problem. Preprocessing significantly reduces the dimension of the problems, therefore improving the efficiency of solution procedures. Heuristic algorithms based on GRASP and Path re-linking are then developed and tested on existing test instances, obtaining excellent results.

**Keywords**— Project management and scheduling; Partially renewable resources; Heuristics; GRASP; Path re-linking

## I. INTRODUCTION

PROJECT scheduling consists of allocating scarce resources to the set of activities in a project over time. Project scheduling has been the object of a great deal of research since the first methods, CPM [7] and PERT [15], were developed in the 1950's. These initial methods were able to manage large projects and were considered a useful tool in the planning process. However, they assumed unlimited resources, an assumption severely reducing their application to most real problems. Therefore, many researchers started to study the resource-constrained case (RCPSP). Up to now, many exact and heuristic algorithms have been developed (see the book by Demeulemeester and Herroelen [4] for an excellent state-of-the -art description). The now classic RCPSP basically includes renewable resources, in which the availability of each resource is renewed at

each period of the planning interval. The case of non-renewable resources, whose availability is given once for the whole project and are consumed throughout the process of the activities requiring them, has also been extensively studied.

However, in recent years new types of resources have been proposed to allow the model to include new types of constraints. An example is allocatable resources [9], [14], in which the units of the resource required by a given operation  $i$  remain occupied from the start of a given allocating activity up to the completion of activity  $i$ . This type of resource is useful to model situations in which the units of a resource are not indistinguishable. For instance, if the resource corresponds to workers of a given type, the person performing an activity consisting of a direct service to clients must carry out some other activity involving the same clients and cannot be changed for another worker of the same type.

Another example of new resources is cumulative resources [10], [11]. In this case, the amount of a resource decreases when it is used by some activities, but it can increase as the result of the process of some other activities, as is the case in some industries involving chemical products.

Another new type of resource is partially renewable resources. The availability of the resource is associated to a subset of periods of the planning horizon and the activities requiring the resource only consume it if they are processed in these periods. Although these resources may seem strange at first glance, they can be a powerful tool for solving project scheduling problems. On the one hand, from a theoretical point of view, they include renewable and non-renewable resources as particular cases. In fact, a renewable resource can be considered a partially renewable resource with an associated subset of

periods consisting of exactly one period. Non-renewable resources are partially renewable resources where the associated subset is the whole planning horizon. On the other hand, partially renewable resources make it possible to model complicated labor regulations and timetabling constraints, therefore allowing us to approach many labor scheduling and timetabling problems as special cases of project scheduling problems.

As an example, let us consider a project involving human resources. We can find some contractual conditions like that of *working at most 2 weekend days out of every 3 consecutive weeks*. This condition cannot be modelled as a renewable resource, because this type of resource considers each period separately. It cannot be modelled as a non-renewable resource because this type of resource considers the whole planning horizon. We model this condition as a partially renewable resource with a set of periods  $\{6, 7, 13, 14, 20, 21\}$  for the first three weekends and a total availability of 2 units. Each task consumes 1 unit of this resource for each weekend day in which it is processed. In Figure 1 we see three activities A, B, and C scheduled within the timescale depicted above. Activity A is in process at periods 6 and 7 and then it consumes 2 units of the resource. Activity B does not consume the resource and activity C consumes 1 unit in period 20. If these 3 activities had to be done by the same worker, the solution in the figure would not be possible because it would exceed resource availability.

Partially renewable resources were first introduced by Böttcher et al. [1] in 1999. They proposed an integer formulation and developed exact and heuristic algorithms. Schirmer [13] studied these new type of resources thoroughly in his book on project scheduling problems. He presented many examples of special conditions which can be suitably modelled using partially renewable resources. He also proposed several families of approximate algorithms for solving the problem, which is denoted by RCPSP/ $\pi$ .

In this paper we develop some preprocessing techniques and several heuristic algorithms for project scheduling under partially renewable resources. Preprocessing reduces the dimension of the problems in terms of resources and possible finishing times for the activities in the project, therefore improving the efficiency of the algorithms. Some heuristic algorithms, based on GRASP and Path Relinking, are then developed and tested on existing test instances. In Section 2 the elements of the problem are defined and an integer formulation provided. Section 3 contains the preprocessing routines. In Section 4 we develop the heuristic algorithms. Section 5 is devoted to the computational experience and Section 6 to conclusions and future lines of research.

## II. FORMULATION OF THE PROBLEM

The RCPSP/ $\pi$  can be defined as follows: Let  $J$  be the set of  $n = |J|$  activities, numbered from 1 to  $n$ , where activity 1 and activity  $n$  are dummy activities representing the beginning and end of the project. Let  $P_j$  be the set of activities which are immediate predecessors of activity  $j$  and  $P'_j$  the set of all predecessors of  $j$ . Each activity  $j$  has a duration of  $d_j$  and cannot be interrupted. Let  $R$  be the set of partially renewable resources. Each resource  $r \in R$  has a total availability  $K_r$  and an associated set of periods  $\Pi_r$ . An activity  $j$  requiring resource  $r$  will consume  $k_{jr}$  units of it at each period  $t \in \Pi_r$  in which it is processed. Finally, let  $T$  be the planning horizon in which all the activities must be processed. For each activity  $j$  we obtain the earliest and latest finishing times,  $EFT_j, LFT_j$ , by critical path analysis. We denote  $E_j = \{EFT_j, \dots, LFT_j\}$ , the set of possible finishing times, and  $Q_{jt} = \{t, \dots, t + d_j - 1\}$ .

The RCPSP/ $\pi$  consists of sequencing the activities so that the precedence and resource constraints are satisfied and the makespan is minimized.

If we define the variables:

$$x_{jt} = \begin{cases} 1 & \text{if activity } j \text{ finishes at time } t \\ 0 & \text{otherwise.} \end{cases}$$

the problem can be formulated as follows:

$$\text{Min} \quad \sum_{t \in E_n} tx_{nt} \quad (1)$$

$$\text{s.t.} \quad \sum_{t \in E_j} x_{jt} = 1 \quad j \in J \quad (2)$$

$$\sum_{t \in E_i} tx_{it} \leq \sum_{t \in E_j} (t - d_j)x_{jt} \quad j \in J, i \in P_j \quad (3)$$

$$\sum_{j \in J} k_{jr} \sum_{t \in \Pi_r} \sum_{q \in Q_{jt} \cap E_j} x_{jq} \leq K_r \quad r \in R \quad (4)$$

$$x_{jt} \in \{0, 1\} \quad j \in J, t \in E_j \quad (5)$$

The objective function (1) minimizes the finishing time of the last activity and hence the makespan of the project. According to constraints (2) each activity must finish once. Constraints (3) are the precedence constraints and constraints (4) the resource constraints. Note that in this problem there is only one global constraint for each resource  $r \in R$ . Another special characteristic of this problem is that all the activities must finish inside a closed interval  $E_j$ , because sets  $\Pi_r$  are defined with respect to the planning horizon

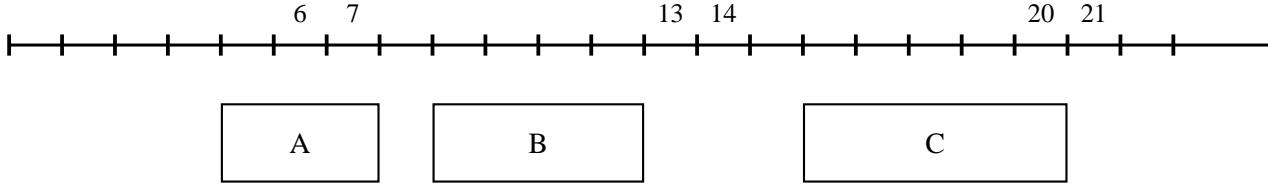


Fig. 1. Example of partially renewable resource

$T$ . Therefore, the existence of feasible solutions is not guaranteed. In fact, Schirmer [13] has shown that the feasibility variant of the RCPSP/ $\pi$  is NP-complete in the strong sense.

The above formulation is called the normalized formulation by Böttcher et al. [1] and Schirmer [13]. Alternative formulations are considered in their papers, but they finally adopt the normalized formulation due to its simplicity.

### III. PREPROCESSING

Preprocessing has two objectives. First, to decide whether a given instance is unfeasible or if it has feasible solutions. If the latter is the case, a second objective is to reduce the number of possible finishing times of the activities and the number of resources. If these two objectives are satisfactorily achieved, the solution procedures will not waste time trying to solve unfeasible problems and will concentrate their efforts on the relevant elements of the problem.

Preprocessing consists of several phases:

#### 1) Reducing the planning horizon $T$

For each instance, we are given a planning horizon  $T$ . This value plays an important role in the problem formulation. In fact, late finishing times of the activities,  $LFT_j$  are calculated starting from  $T$  in a backward recursion. Therefore, the lower the value  $T$ , the fewer variables the problem will have. In order to reduce  $T$ , we try to build a feasible solution for the given instance, using the GRASP algorithm which will be described later. The GRASP iterative process stops as soon as a feasible solution is obtained or after 200 iterations. The new value  $T$  is updated to the makespan of the feasible solution obtained. Otherwise,  $T$  is unchanged.

If the makespan of the solution equals the length of the critical path in the precedence graph, the solution is optimal and the process stops and returns the solution.

#### 2) Eliminating idle resources

Each resource  $r \in R$  is consumed only if the activities requiring it are processed in periods  $t \in \Pi_r$ . Each activity can only be processed in a finite interval. It is therefore possible that no activity requiring the resource can be processed in any period of  $\Pi_r$ . In this case, the resource is idle and can be eliminated. More precisely, if we denote the possible processing times of activity  $j$  by  $PPT_j = \{EFT_j - d_j + 1, \dots, EFT_j, \dots, LFT_j\}$ , and  $\forall j \in J \mid k_{rj} > 0 : \Pi_r \cap PPT_j = \emptyset$ , the resource  $r \in R$  is idle and can be eliminated.

#### 3) Eliminating non-scarce resources

Schirmer [13] distinguishes between scarce and non-scarce resources. He considers a resource  $r \in R$  as scarce if  $\sum_{j \in J} k_{jr} d_j > K_r$ , that is, if an upper bound on the maximum resource consumption exceeds resource availability. In this case, the upper bound is computed by supposing that all the activities requiring the resource are processed completely inside  $\Pi_r$ .

We have refined this idea by taking into account the precedence constraints. Specifically, we calculate an upper bound on the maximal consumption of resource  $r$  by solving the following linear problem:

$$\text{Max } \sum_{j \in J} k_{jr} \sum_{t \in \Pi_r} \sum_{q \in Q_{jt} \cap E_j} x_{jq} \quad (6)$$

$$\text{s.t. } \sum_{t \in E_j} x_{jt} = 1 \quad j \in J \quad (7)$$

$$\sum_{m=t}^T x_{im} + \sum_{s=1}^{t+d_j-1} x_{js} \leq 1, \quad j \in J, \quad i \in P_j, \quad t \leq T \quad (8)$$

$$x_{jt} \geq 0 \quad j \in J, \quad t \in E_j \quad (9)$$

The objective function (6) maximizes the resource consumption over the whole project. Constraints (7) ensure that each activity finishes once. Constraints (8) are the precedence constraints. We use this expression, introduced by Christofides et al. [3], because it is more efficient than the usual

precedence constraint. In fact, this linear problem has the integrality condition and its optimal solution is always integer [2]. If the solution value is not greater than the resource availability, this resource will not cause any conflict and can be skipped in the solution process.

#### 4) A filter for variables based on resources

For each activity  $j$  and each possible finishing time  $t \in E_j$  we perform the following test to decide if this time  $t$  is not feasible for activity  $j$  to finish in.  $Q'_{jrt} = \{t-d_j+1, t-d_j+2, \dots, t\} \cap \Pi_r$  is the set of processing times of  $j$  which lie inside  $\Pi_r$  if it finishes at time  $t$ , and  $m_{jrt} = |Q'_{jrt}|$  denotes the corresponding number of periods.

- For each predecessor  $i \in P'_j$ , let  $ER_{it}$  be the reduced set of possible finishing times of  $i$  if  $j$  finishes at time  $t$ .
- For each successor  $l$  ( $j \in P'_l$ ), let  $ER_{lt}$  be the reduced set of finishing times of  $l$  if  $j$  finishes at time  $t$ .
- Let  $U_j$  be the set of activities not related to  $j$  by precedence constraints.
- Consider the resources  $r \in R$ , one at a time, and compute the minimal consumption of the resource if activity  $j$  finishes at time  $t$ :

$$MC_r = m_{jrt}k_{jr} + \sum_{i \in P'_j} \min_{s \in ER_{it}} \{m_{irs}k_{ir}\} + \sum_{l|j \in P'_l} \min_{s \in ER_{lt}} \{m_{lrs}k_{lr}\} + \sum_{h \in U_j} \min_{s \in E_h} \{m_{hrs}|k_{hr}\}$$

(predecessors of  $j$ )  
(successors of  $j$ )  
(remaining activities)

- If  $MC_r > K_r$ , time  $t$  is not feasible for activity  $j$  to finish in and the corresponding variable  $x_{jt} = 0$ .

When this filter is applied to an activity  $j$  some of its possible finishing times can be eliminated. From then on, the set of possible finishing times is no longer  $E_j$ . We denote by  $PFT_j$  the set of finishing times passing the filter.

This filter is applied iteratively. After a first run on every activity and every finishing time, if some of the variables are eliminated the process starts again but this time computing  $MC_r$  on the sets  $PFT_j, PFT_i, PFT_l, PFT_s$  instead of the original  $E_j, E_i, E_l, E_s$ . As the minima are calculated over restricted subsets, it is possible that new finishing times fail the test and are eliminated. The process is repeated until no finishing time is eliminated in a complete run.

#### 5) Consistency test for finishing times

When the above filter eliminates a finishing time of an activity  $j$ , it is possible that some of the finishing times of its predecessors and successors are no longer feasible. For instance, suppose an activity  $j$  with  $PFT_j = \{9, 10, 11, 12, 13, 14, 15\}$  and duration  $d_j = 3$  and an activity  $i \in P_j$  with  $PFT_i = \{6, 7, 8, 9, 10, 11, 12\}$ . If the resource filter eliminates  $t = 15$  from  $PFT_j$ , then  $t = 12$  is not possible for activity  $i$  because if  $i$  finishes at time 12,  $j$  must necessarily finish at time 15, which is unfeasible. Therefore, time 12 for activity  $i$  is eliminated.

In general, for an activity  $j$  let us denote by  $\tau_j = \max\{t \mid t \in PFT_j\}$ . Then, for each  $i \in P_j$  the finishing times  $t \in PFT_i$  such that  $t > \tau_j - d_j$  can be eliminated. Analogously, if for activity  $j$ ,  $\gamma_j = \min\{t \mid t \in PFT_j\}$ , for each  $i \mid j \in P_i$  the finishing times  $t \in PFT_i$  such that  $t < \gamma_j + d_i$  can be eliminated.

This test is also applied iteratively until no more finishing times are eliminated. If, after applying these two procedures for reducing variables, an activity  $j$  has  $PFT_j = \emptyset$ , the problem is unfeasible and the procedure stops, returning the unfeasibility status of the given instance.

#### 6) Constructing a trial solution

In the first step of the preprocessing procedure we tried to build a feasible solution. If the feasible solution was obtained, we checked if its makespan was equal to the length of the critical path. If this was the case, the solution was optimal. After the elimination of variables, we then check if the makespan of that solution equals the minimum time in  $PFT_n$ . If this is the case, the solution is optimal.

Otherwise, we build a trial solution by assigning a finishing time  $t_j = \min\{t \mid t \in PFT_j\}$  to each activity  $j$ . Obviously this solution satisfies the precedence constraints. If it satisfies the resource constraints as well, it is the optimal solution.

## IV. GRASP ALGORITHM

GRASP, greedy randomized adaptive search procedure, is an iterative process combining a constructive phase and an improvement phase. The construction phase builds a solution step by step, adding elements to a partial solution. The element to add is selected according to a greedy function which is dynamically adapted as the solution is built. However, the selection is not deterministic, but subjected to a randomization process. Hence, when we repeat the process we can

obtain different solutions. When a feasible solution has been built, its neighborhood is explored in a local search phase until a local optimum is found. Resende and Ribeiro [12] present a comprehensive review of GRASP and an extensive survey of GRASP literature can be found in Festa and Resende [6].

### A. The constructive phase

#### *A deterministic constructive algorithm*

We have adapted the Serial Scheduling Scheme (SSS) proposed by Schirmer [13], which in turn is an adaptation of the Serial Scheduling Scheme commonly used for the classical RCPSP. We denote by  $FT_j$  the finishing time assigned to activity  $j$ . At each stage of the iterative procedure an activity is scheduled by choosing from among the current set of decisions, pairs  $(j, t)$  of an activity  $j$  and a possible finishing time  $t \in PFT_j$ . The selection is based on a priority rule. The algorithm is described in Figure 2.

At Step 1, the construction of  $D_s$  could have included the feasibility test of Step 3, as in Schirmer's [13] original scheme. However, we have preferred not to check the resource availability of every decision and only check the decision already chosen. In problems with a large number of possible finishing times for the activities, this strategy is more efficient.

We keep the set of possible scarce resources  $SR_s$  updated because some priority rules based on resource consumption only take this type of resources into account.

#### *Priority rules*

We have tested the 32 priority rules used by Schirmer [13]. The first 8 are based on the network structure, including classical rules such as EFT, LFT, SPT or MINSLK. The other 24 rules are based on resource utilization. 12 of them use all the resources and the other 12 only the scarce resources. A preliminary computational experience, which will be fully described in Section 6, allowed us to choose the most promising rules and use them in the next phases of the algorithm's development. These preliminary results also showed that even with the best performing rules the deterministic constructive algorithm failed to obtain a feasible solution for many instances of 10 activities generated by Böttcher et al. [1]. Therefore, the objective of the randomization procedures which were included in the algorithm was not only to produce diverse solutions but to ensure that for

most of the problems the algorithm would obtain a feasible solution.

#### *Randomization strategies*

We introduce randomization procedures for selecting the decision at Step 2 of the constructive algorithm. Let  $s_{jt}$  be the score of decision  $(j, t)$  on the priority rule we are using and  $s_{max} = \max\{s_{jt} | (j, t) \in D_s\}$ , and let  $\delta$  be a parameter to be determined ( $0 < \delta < 1$ ). We have considered three alternatives:

- a) *Random selection on the Restricted Candidate List, S*

Select decision  $(j^*, t^*)$  at random in set  
 $S = \{(j, t) | s_{jt} \geq \delta s_{max}\}$

- b) *Biased selection on the Restricted Candidate List, S*

We build the Restricted Candidate List as in alternative (a), but instead of choosing at random from among its elements, the decisions involving the same activity  $j$  are given a weight which is inversely proportional to the order of their finishing times. For instance, if in  $S$  we have decisions  $(2, 4), (2, 5), (2, 7), (2, 8)$  involving activity 2 and ordered by increasing finishing times, then decision  $(2, 4)$  will have a weight of 1, decision  $(2, 5)$  weight  $1/2$ , decision  $(2, 7)$  weight  $1/3$  and decision  $(2, 8)$  weight  $1/4$ . The same procedure is applied to the decisions corresponding to the other activities. Therefore, the decisions in  $S$  corresponding to the lowest finishing times of the involved activities will be equally likely and the randomized selection process will favor them.

- c) *Biased selection on the set of decisions  $D_n$*

We have also implemented the Modified Regret-Based Biased Random Sampling (MRBRS/ $\delta$ ) proposed by Schirmer [13], in which the decision  $(j, t)$  is chosen from among the whole set  $D_n$  but with its probability proportional to its regret value. The regret value is a measure of the worst possible consequence that might result from selecting another decision.

#### *A repairing mechanism*

The randomization strategies described above significantly improve the ability of the constructive algorithm to find feasible solutions for tightly constrained instances. However, a limited computational experience showed that not even with the best priority rule and the best randomization

**Step 0. Initialization**

$s = 1$  (counter of stage)

$FT_1 = 0$  (sequencing dummy activity 1)

$S_1 = \{1\}$  (partial schedule at stage 1)

$\forall r \in R : RK_{r1} = K_r$  (remaining capacity of resource  $r$  at stage 1)

$$TD_{r1} = \sum_{j \in J} k_{jr} d_j \text{ (maximum possible demand for } r \text{ at stage 1)}$$

$SR_1 = \{r \in R \mid TD_{r1} > RK_{r1}\}$  (set of possible scarce resources)

$EL_1$  = set of eligible activities, those for which activity 1 is the only predecessor

**Step 1. Constructing the set of decisions**

$$D_s = \{(j, t) \mid j \in EL_s, t \in PFT_j\}$$

**Step 2. Choosing the decision**

Select the best decision  $(j^*, t^*)$  in  $D_s$ , according to a priority rule

**Step 3. Feasibility test**

If  $(j^*, t^*)$  is resource-feasible, go to Step 4.

Else

$$D_s = D_s \setminus \{(j^*, t^*)\}$$

If  $D_s = \emptyset$ , STOP. The algorithm does not find feasible solution.

Else, go to Step 2.

**Step 4. Update**

$$s = s + 1$$

$$FT_{j^*} = t^*$$

$$S_s = S_{s-1} \cup \{j^*\}$$

$$EL_s = (EL_{s-1} \setminus \{j^*\}) \cup \{j \in J \mid P_j \subseteq S_s\}$$

$$\forall l \in J \mid j \in P_l : PFT_l = PFT_l \setminus \{\tau \mid t^* + d_l > \tau\}$$

$$\forall r \in R : RK_{rs} = RK_{r,s-1} - k_{j^*r} m_{j^*rt^*}$$

$$TD_{rs} = TD_{r,s-1} - k_{j^*r} d_{j^*}$$

$$\text{If } TD_{rs} \leq RK_{rs}, \text{ then } SR_s = SR_{s-1} \setminus \{r\}$$

If  $s = n$ , STOP. The sequence is completed.

Else, go to Step 2.

Fig. 2. Constructive algorithm

procedure could the constructive algorithm obtain feasible solutions for all the instances of 10 activities generated by Böttcher et al. [1]. Therefore, we felt that the algorithm was not well-prepared for solving larger problems and we decided to include a repairing mechanism for unfeasible partial schedules.

In the construction process, if at Step 3 all decisions in  $D_n$  fail the feasibility test and finally  $D_n$  becomes empty, instead of stopping the process and starting a new iteration, we try to re-assign some of the already sequenced activities to other finishing times in order to free some resources that could be used for the first of the unscheduled activities to be processed. If this procedure succeeds, the constructive process continues. Otherwise, it stops. A detailed description of the repairing mechanism is not provided because it is very similar to the double move described in the next subsection.

**B. The improvement phase**

Given a feasible solution obtained in the constructive phase, the improvement phase basically consists of two steps. First, identifying the activities whose finishing times must be reduced in order to have a new solution with the shortest makespan. These activities are labelled as *critical*. Second, moving critical activities in such a way that the resulting sequence is feasible according to precedence and resource constraints. We have designed two types of moves: simple and double. In a simple move, only a critical activity is moved, leaving the remaining activities unchanged. In a double move, non-critical activities are moved to make the move of a critical activity possible.

The procedure for building  $M$ , the set of critical activities, appears in Figure 3. At Step 1, the condition for including an activity in  $M$  simply says that if  $j$  has to be moved to the left, reducing its finishing time, a predecessor  $i$  which is processed immediately before

$j$  must also be moved to the left in order to leave room for moving  $j$ . This condition can be refined if we take into account that the preprocessing filters may have eliminated some possible finishing times of the activities. If  $t'_j = \max\{t \in PFT_j \mid t'_j < FT_j\}$ , the condition of Step 1 can be written as: If  $FT_i + d_j > t'_j$ , then  $i$  is critical.

For instance, suppose we have activity  $4 \in M$  with  $FT_4 = 14$ ,  $d_4 = 5$ ,  $PFT_4 = \{10, 11, 12, 14\}$ , and activity 2 is a predecessor of 4 with  $FT_2 = 8$ . If activity 4 has to be moved to the left, its new finishing time will be 12 at most and therefore  $FT_2$  can no longer be 8. Activity 2 must be moved to the left and hence  $2 \in M$ . In this example,  $t'_4 = 12$  and  $FT_2 + d_4 = 13$ .

The simple move is described in Figure 4. We try to move every activity  $j \in M$  to the left, in topological order, to a new finishing time satisfying the precedence and resource constraints. If an activity cannot be moved, the procedure stops. If for an activity there are several possible new finishing times, that with minimum global resource consumption is chosen.

A description of the double move appears in Figure 5. In Step 2, the new finishing time which is being considered for activity  $j \in M$  may be resource-feasible and no other activity needs to be moved. If this is not the case, in Step 3 other activities are considered for moving. An activity  $i$  is moved to a new provisional finishing time if this move offsets the resource violation provoked by moving  $j$  or, at least, reduces the deficit. Therefore, throughout the search in  $J$ , a provisional list of changes  $LC$  is built until the solution is repaired or  $J$  is exhausted. If the solution is repaired with the list of changes in  $LC$ , those moves are made and a new  $j \in M$  is considered. Otherwise, the procedure stops without improving the solution.

The double move can be enhanced in the following way. If we arrive at Step 6 without completely covering the deficit created by moving  $j$ , but this deficit is partially reduced, we can go back to Step 3 and search  $J$  again from the beginning, trying to further reduce or eliminate the remaining deficit. The procedure is more complex but sometimes offers feasible moves for critical activities.

The three procedures of the improvement phase are run iteratively:

```

 $S =$  current solution
 $improve = false$ 
do{
    Build set  $M$  of critical activities
     $improve = SimpleMove(S, M)$ 
    if  $improve = false$ 
         $improve = DoubleMove(S, M)$ 
} while ( $improve = true$ )

```

### C. An aggressive procedure

The standard version of our heuristic algorithm starts by applying the preprocessing procedure of Section 3. The reduced problem then goes through the iterative GRASP algorithm described above, combining a constructive phase and an improvement phase at each iteration, until the stopping criterion, here a fixed number of iterations, is met.

An enhanced version of the heuristic algorithm combines preprocessing and GRASP procedures in a more aggressive way. After a given number of iterations (stopping criterion), we check if the best known solution has improved. If this is the case, we run the preprocessing procedures again, setting the planning horizon  $T$  to the makespan of the best known solution and running the filters for variable reduction. The GRASP algorithm is then applied on the reduced problem. Obtaining feasible solutions is now harder, but if the procedure succeeds we will get high quality solutions. A scheme of the modified algorithm appears in Figure 6.

### D. Path Relinking

If throughout the iterative procedures described above we keep a set of the best solutions, usually denoted as *elite solutions*, we can perform a Path Relinking procedure. Starting from one of these elite solutions, called the *initiating solution*, we build a path towards another elite solution, called the *guiding solution*. To the intermediate solutions in the path we progressively impose the attributes of the guiding solution, so these intermediate solutions evolve from the initiating solution until they reach the guiding solution. Hopefully, along these paths we will find solutions which are better than both extremes, the initiating and the guiding solutions.

We keep the 10 best solutions obtained in the GRASP procedure. We consider one of them in turn as the initiating solution and another as the guiding solution. We build a path from the initiating to the final solution with  $n - 1$  intermediate solutions. The  $j^{th}$  solution will have the finishing times of the first  $j$  activities taken from the guiding solution, while the remaining  $n - j$  finishing times will still correspond to those of the initiating solution. Therefore, along the path, the intermediate solutions will be progressively more similar to the guiding solution and more different from the initiating one. In some cases these intermediate solutions will not be feasible. If this is the case, a repairing mechanism similar to that described in Section 4 is applied. We proceed from activity 1 to activity  $n$ , checking for each activity  $j$  if the partial solution from 1 to  $j$  is feasible. If it is not, we first try to find a feasible finishing time for activity  $j$ , keeping

**Step 0. Initialization**

$M = \{n\}$  (the last activity of the project  $n$  is always critical)  
 $s_n = 1$  (activity  $n$  has not yet been studied for enlarging  $M$ ).

**Step 1. Adding activities to  $M$** 

```
While(  $\exists j \in M | s_j = 1$  ) {
    Take the largest  $j \in M$  with  $s_j = 1$ . Set  $s_j = 0$ 
     $\forall i \in P_j :$ 
        If  $FT_i + d_j = FT_j$  (there is no slack between  $i$  and  $j$ )
         $M = M \cup \{i\}$ 
         $s_i = 1$ 
}
```

Fig. 3. Building the critical set

**Step 0. Initialization**

$RK_r, \forall r \in R,$  are the resources not used in the current sequence  
 $u_j = 1, \forall j \in M$  (activity still to be moved)  
 $possible = true$  (the move is still possible)

**Step 1. Moving activities in  $M$** 

```
While(  $\exists j \in M | u_j = 1$  )
{
    Take the minimum  $j \in M$  with  $u_j = 1$ 
    Set  $u_j = 0$ 
     $MINFT_j = \max \{FT_i + d_j | i \in P_j\}$ 
     $t_{best} = FT_j$ 
     $max_{excess} = 0$ 
     $\forall t \in PFT_j | MINFT_j \leq t < FT_j$ 
        {  $\forall r \in R : RK_r = RK_r + k_{jr}m_{jrFT_j} - k_{jrt}m_{jrt}$ 
            If  $RK_r \geq 0, \forall r \in R$  (possible move)
                 $excess = \sum_{r \in R} RK_r$ 
                If  $max_{excess} < excess$ 
                     $max_{excess} = excess$ 
                     $t_{best} = t$ 
                Recover previous  $RK_r$ 
        }

```

```
If  $t_{best} = FT_j$  (no change)
    Recover the original  $FT_j, \forall j \in J$  and return false
Else,  $FT_j = t_{best}$ 
}
```

**Step 2.**

Return *true* and the modified solution

Fig. 4. Simple move

previous activities unchanged. If that is not possible, we try to re-assign some of the previous activities to other finishing times in order to obtain some resources which are necessary for processing activity  $j$  at one of its possible finishing times. If this procedure succeeds, we consider activity  $j + 1$ . Otherwise, the solution is discarded and we proceed to the next intermediate solution. If we obtain a complete intermediate solution which is feasible, we apply to it the improvement phase described in the GRASP algorithm.

**V. COMPUTATIONAL RESULTS****A. Test instances**

Böttcher et al. [1] generated a first set of test instances. Taking as their starting point PROGEN 2 [8], an instance generator for the classical RCPSP with renewable resources, they modified and enlarged the set of parameters and generated a set of 2160 instances with 10 non-dummy activities, 10 replications for each one of the 216 combinations of parameter values. As

**Step 0. Initialization**

$RK_r, \forall r \in R$ , are the resources not used in the current sequence

$u_j = 1, \forall j \in M$  (activity still to be moved)

$possible = true$  (the move is still possible)

While(  $\exists j \in M | s_j = 1$  )

{

**Step 1. Selecting an activity  $j \in M$  to be moved**

Take the minimum  $j \in M$ , with  $u_j = 1$ . Set  $u_j = 0$ .

$MINFT_j = \max \{FT_i + d_j | i \in P_j\}$

**Step 2. Considering a new finishing time for  $j$** 

$\forall t \in PFT_j | MINFT_j \leq t < FT_j$

{  $repaired = true$

$\forall r \in R : RK_r = RK_r + k_{jr}m_{jrFT_j} - k_{jr}m_{jrt}$

If  $RK_r \geq 0, \forall r \in R$

$FT_j = t$ . Go to Step 1, to move another critical activity.

Else

**Step 3. Moving other activities  $i \in J$** 

$LC = \emptyset$ , list of possible changes

$\forall i \in J | i \neq j$

{  $MINFT_i = \max \{FT_l + d_i | l \in P_i\}, MAXFT_i = \min \{FT_k - d_k | i \in P_k\}$

$t_{best} = FT_i$ .  $best_{viol} = 0$

**Step 4. New finishing times for activity  $i$** 

$\forall u \in PFT_i | MINFT_i \leq u \leq MAXFT_i$

{  $new_{viol} = 0$

{  $\forall r \in R : change_r = k_{ir}m_{iru} - k_{ir}m_{irFT_i}$

If  $RK_r > 0$  and  $RK_r - change_r < 0$

$new_{viol} = new_{viol} + (change_r - RK_r)$

$repaired = false$

If  $RK_r < 0$

If  $change_r < 0$

$new_{viol} = new_{viol} - \min\{-RK_r, -change_r\}$

Else,  $new_{viol} = new_{viol} + change_r$

If  $RK_r < change_r$

$repaired = false$  }

If  $repaired = true$

$t_{best} = u$ . Go to Step 5.

If  $best_{viol} > new_{viol}$

$best_{viol} = new_{viol}$   $t_{best} = u$

} (end of Step 4)

**Step 5. Add to the list of possible changes**

If  $t_{best} \neq FT_i$

$LC = LC \cup \{(i, t_{best})\}$ . Update  $RK_r$  and  $FT_i$ .

If  $repaired = true$  : Go to Step 6

} (end of Step 3)

**Step 6. Make changes associated to activity  $j$** 

If  $repaired = true$

$FT_j = t$ . Make changes in  $LC$  and update  $FT_i, RK_r$ .

Else, return  $false$

} (end of Step 2)

} (end of main While)

**Step 7.**

Return  $true$  and the modified solution

Fig. 5. Double move

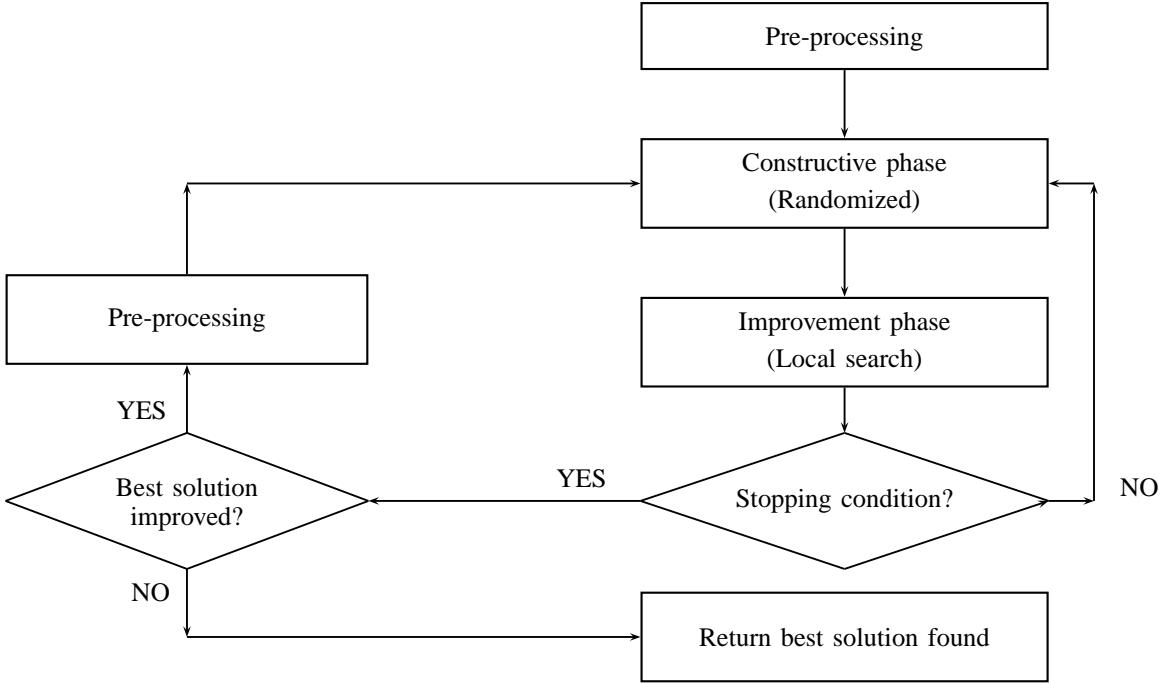


Fig. 6. Scheme of Aggressive GRASP

most of the problems were unfeasible, they restricted the parameter values to the 25 most promising combinations and generated 250 instances of sizes 15, 20, 25, 30 and 60 of non-dummy activities, always keeping the number of resources to 30.

More recently, Schirmer [13] has developed PROGEN 3, an extension of PROGEN 2, and has generated some new test instances. He has generated 960 instances of sizes 10, 20, 30 and 40, with 30 resources. Most of them have a feasible solution, while a few of them are unfeasible and some of them are labelled as undecided because a time-limited run of the branch and bound algorithm by Böttcher et al. [1] failed to obtain a feasible solution. Table 1 shows the status of Schirmer's problems as reported in [13].

#### B. Preprocessing results

The preprocessing procedures in Section 3 have been applied to the Böttcher et al. [1] problems of 10, 15, 20, 25 and 30 (non-dummy) activities which are available upon request from the authors. Different aspects of the results appear in Tables 2, 3 and 4. Table 2 shows the performance of preprocessing in determining problem status.

The last line of Table 2 shows the status we have been able to determine for the problems left undecided by the

preprocessing procedures. We have tried to solve these instances with CPLEX, using an integer programming formulation of the problem adapted from that appearing in Section 2, though for 2 instances of size 20 and 3 instances of size 30 long time runs of this powerful code failed to obtain even a feasible integer solution. In summary, we can say that our preprocessing procedures are very efficient in determining the actual status of a given instance.

Table 3 shows the optimal solutions that the preprocessing obtains either by proving that the initial feasible solution is optimal, or by building a trial solution in which each activity is assigned to its minimum finishing time after the reduction filters have been applied. For more than 70 % of the instances the optimal solutions are found.

Table 4 presents the reduction in the number of resources and variables for the problems not solved in preprocessing, for which some other algorithm has to be applied. The fast preprocessing techniques significantly reduce the number of resources to be taken into account and, more importantly, the number of possible values of the decision variables.

Similar results have been obtained for the test problems generated by Schirmer [13]. Table 5 shows the performance of preprocessing, first determining the status of all of the problems and then providing optimal

Instance	Non-optimally solved	Optimally solved	Feasibly solved	Undecided	Proven infeasible	Total
Set						
J10	39	901	940	11	9	960
J20	203	734	937	23	0	960
J30	181	757	938	22	0	960
J40	183	743	926	34	0	960
Total	606	3135	3741	90	9	3840

TABLE I  
*Test problems generated by Schirmer*

	n=10	n=15	n=20	n=25	n=30
Problems	2160	250	250	250	250
Detected as impossible	1205	16	17	12	8
Detected as possible	879	233	231	236	239
Undecided	76	1	2	2	3
Actual status	Impossible	Possible	Undecided	Impossible	Undecided

TABLE II  
*Böttcher et al. problems - Determining the status*

	n=10	n=15	n=20	n=25	n=30
Problems	2160	250	250	250	250
Feasible problems	879	234	233	236	242
Solved to optimality by pre-processing	646	165	177	190	193
Remaining problems	233	67	56	46	49

TABLE III  
*Böttcher et al. problems - Optimal solutions identified in the preprocessing*

	n=10	n=15	n=20	n=25	n=30
Problems	233	67	56	46	49
Initial resources	30	30	30	30	30
Remaining resources (on average)	18	18	23	25	25
Initial variables (on average)	90	268	565	874	1314
Remaining variables (on average)	51	130	348	611	906

TABLE IV  
*Böttcher et al. problems - Reductions of resources and variables*

solutions for many of them. Note that the status of all problems left undecided in Schirmer's book [13] have been determined. In fact, all of them have been proven to be feasible, except for five instances of size 10 which are impossible. For more than 75 % of the feasible problems, the preprocessing procedures are able to provide a proven optimal solution.

A characteristic of PROGEN 3 is that it tends to produce large values of the planning horizon  $T$ . On the one hand, that favors the existence of feasible solutions. On the other hand, as the number of possible finishing times of activities depends directly on  $T$ , a very large number of variables are initially defined. Therefore, for this set of problems the reduction of  $T$  described in Section 3 is especially useful. Table 6 shows the reduction of  $T$  obtained by that procedure on the non-optimally solved problems.

The reductions of the planning horizon  $T$ , together with the procedures for reducing possible finishing times for the activities, produce dramatic decreases in the final number of variables to be used by solution procedures. Table 7 presents the reductions in the number of resources and variables obtained by the preprocessing strategies.

### C. Computational results of constructive algorithms

The 32 priority rules described by Schirmer [13] were coded and embedded in the constructive algorithm of Section 4.1. These rules were tested on the 879 feasible instances of size 10 generated by Böttcher et al. [1]. Table 8 shows the results obtained by the 6 best performing rules. The first 3 rules are based on the network structure of the problems. The last 3 rules are based on resource consumption. In them,  $ES$  indicates that the rules require the use of only scarce resources, indexed by  $r$ .  $Rk_{rs}$  is the remaining capacity of resource  $r$  at stage  $s$ , as defined in Section 4.1.  $RD_{jrt}$  is the relevant demand, defined as  $RD_{jrt} = k_{jr}|Q_{jt} \cap \Pi_r|$ .  $MDE_{jrt}$  is the minimum relevant demand entailed for resource  $r$  by all successors of activity  $j$  when started at period  $t$ . The most important feature of Table 8 is that even the best rules fail to produce a feasible solution for 20% of these small instances of size 10. Therefore, we need randomizing strategies and repairing mechanisms to significantly increase the probability of finding feasible solutions in the constructive phase of the GRASP algorithm.

Table 9 presents the improvement in the number of feasible and optimal solutions obtained by the constructive algorithm when one of the randomizing strategies are included in Step 2. As in Table 8, the test problems

are the size 10 instances of Böttcher et al. [1]. Only two rules have been kept for this second test,  $LFT$ , which is the best rule among those based on network structure and  $DRC/ES$ , the best rule based on resource usage. Table 9 shows that the randomization procedures allow us to get an important increase in the number of feasible solutions. However, not all these small problems can be solved. That is the reason for the development of a repairing mechanism to help the constructive algorithm to find feasible solutions for the more tightly constrained problems.

Table 10 shows the final results of the complete constructive algorithm, including the repairing mechanism. From Table 9 we have kept *Random 3* because it obtains the highest number of feasible solutions and *Random 2* because it obtains the highest number of optimal solutions. The results show that the constructive algorithm now seems to be well-prepared for solving larger problems. The priority rule  $LFT$  produces many more optimal solutions than  $DRC/ES$ . This rule, based on the use of resources, is more orientated to attaining feasibility by choosing times with low resource requirements than to get optimality by processing activities as early as possible. However, as the feasibility of the solutions is guaranteed by the joint effort of a randomizing strategy and the repairing mechanism, rule  $LFT$  will be chosen for the GRASP algorithm.

### D. Computational results of GRASP algorithms

Tables 11 and 12 show the results of the GRASP algorithms on the problems of Böttcher et al. [1] and Schirmer [13] respectively. Four versions of the algorithm have been tested: *GRASP*, the basic GRASP algorithm, *GR+PR*, in which the best solutions obtained in the GRASP iterations go through the Path Relinking phase described in Section 4.4, *AG-GR*, the modified GRASP procedure described in Section 4.3, and *AG-GR+PR*, combining modified GRASP and Path Relinking. The GRASP algorithms use priority rule  $LFT$  and the second randomization procedure with  $\delta = 0.85$ . For each problem size the Tables show the number of non-optimal solutions, the average distance to optimum and the maximal distance to optimum. However, not all the optimal solutions are known. In fact, in Table 11 for 1 instance of size 20, 7 instances of size 25 and 7 instances of size 30 the optimal solution is unknown. Analogously, in Table 12 the optimal solution is not known for 1 instance of size 30 and 5 instances of size 40. In these cases, which are marked (\*), the comparison is made with the best-known solution, obtained by a time-limited run of the CPLEX integer code or by heuristic methods.

	n=10	n=20	n=30	n=40
Problems	951	960	960	960
Feasible problems	946	960	960	960
Solved to optimality by pre-processing	609	727	796	793
Remaining problems	337	233	164	137

TABLE V  
*Schirmer problems - Optimal solutions identified in the preprocessing*

	n=10 337 problems	n=20 233 problems	n=30 164 problems	n=40 137 problems
Average initial $T$	43	82	120	158
Average reduction	9 (21%)	31 (38%)	52 (43%)	80 (50%)
Maximal reduction	30 (70%)	61 (76%)	100 (84%)	134 (85%)

TABLE VI  
*Schirmer problems - Reductions of planning horizon  $T$*

	n=10 337 problems	n=20 233 problems	n=30 164 problems	n=40 137 problems
Initial resources	30	30	30	30
Remaining resources (average)	15 (50%)	15 (50%)	18 (60%)	16 (53%)
Initial variables (average)	210	965	2287	4255
Remaining variables (average)	101 (48%)	332 (34%)	720 (31%)	1062 (25%)

TABLE VII  
*Schirmer problems - Reductions of resources and variables*

Rule	Definition	Feasible solutions (%)	Optimal solutions (%)
LFT	$\text{Min}\{LFT_j\}$	80.09	64.28
MTS	$\text{Max}\{\{i j \in P_i'\}\}$	79.64	69.98
SLK	$\text{Min}\{LST_j - EFT_j\}$	76.22	61.66
DRC/ES	$\text{Max}\{\sum_r (RK_{rs} - RD_{jrt})\}$	81.57	27.08
DRS/ES	$\text{Min}\{\sum_r (RK_{rs}/RD_{jrt})\}$	79.29	27.53
TRS/ES	$\text{Min}\{\sum_r (RD_{jrt} + MDE_{jrt})\}$	79.41	28.56

TABLE VIII  
*Results of priority rules*

Rule	Randomizing strategy	Feasible solutions (%)	Optimal solutions (%)
LFT	Deterministic	80.09	64.28
	Random 1	97.95	61.89
	Random 2	97.61	93.83
	Random 3	98.41	61.66
DRC/ES	Deterministic	81.57	27.08
	Random 1	96.25	72.81
	Random 2	95.56	76.11
	Random 3	98.41	54.38

TABLE IX  
*Results of randomizing strategies*

Rule	Strategy	Iterations	Feasible solutions (%)	Optimal solutions (%)
LFT	Random 2	1000	99.89	99.09
	Random 2	2000	100	99.43
	Random 3	1000	100	93.63
	Random 3	2000	100	96.36
DRC/ES	Random 2	1000	99.66	89.31
	Random 2	2000	99.66	89.31
	Random 3	1000	100	81.91
	Random 3	2000	100	84.41

TABLE X  
*Results of the complete constructive algorithm*

Problem size	Feasible instances		GRASP	GR + PR	AG - GR	AG - GR + PR
10	879	Non-optimal	1	1	2	2
		Mean dist. (%)	0.006	0.006	0.15	0.15
		Max dist. (%)	5.6	5.6	7.7	7.7
15	234	Non-optimal	4	4	3	3
		Mean dist. (%)	0.13	0.13	0.09	0.09
		Max dist. (%)	17.9	17.9	17.9	17.9
20	231	<i>Non-optimal*</i>	8	8	8	8
		Mean dist. (%)	0.41	0.41	0.33	0.33
		Max dist. (%)	24.2	24.2	27.3	27.3
25	236	<i>Non-optimal*</i>	7	6	8	7
		Mean dist. (%)	0.20	0.19	0.24	0.23
		Max dist. (%)	21.7	21.7	21.7	21.7
30	239	<i>Non-optimal*</i>	5	5	5	4
		Mean dist. (%)	0.10	0.10	0.06	0.05
		Max dist. (%)	11.5	5.8	3.9	3.9

TABLE XI  
*Results of GRASP algorithms on Böttcher et al. problems*

Problem size	Feasible instances		GRASP	GR + PR	AG - GR	AG - GR + PR
10	946	Non-optimal	1	1	2	2
		Mean dist. (%)	0.003	0.003	0.007	0.007
		Max dist. (%)	2.9	2.9	3.4	3.4
20	960	Non-optimal	33	22	20	19
		Mean dist. (%)	0.12	0.08	0.07	0.06
		Max dist. (%)	13.0	13.0	13.0	13.0
30	960	<i>Non-optimal*</i>	58	55	34	34
		Mean dist. (%)	0.22	0.20	0.12	0.11
		Max dist. (%)	12.1	12.1	13.6	13.6
40	960	<i>Non-optimal*</i>	79	76	56	50
		Mean dist. (%)	0.48	0.42	0.25	0.22
		Max dist. (%)	32.0	32.0	20.5	20.5

TABLE XII  
*Results of GRASP algorithms on Schirmer problems*

The results in Table 11 show that only a few very difficult problems of every size are not optimally solved. However, these problems are so hard that almost no difference between algorithms can be observed. The maximum distance to optimum can be relatively very high. For instance, problem *P2408* of size 15 has an optimal solution of 28 while the heuristic solution is 33. However, due to the special type of resources involved, it is possible that no feasible solutions of length 29, 30, 31 and 32 exist. If that were the case, only one possibility of improving is left, though the high value of the maximum distance would seem to suggest the opposite.

The results in Table 12 allow us to observe the different performance of the four algorithms more clearly. The aggressive GRASP procedure does not guarantee a better solution than the basic GRASP algorithm, as can be seen in the first row of the Table, but for larger problems it tends to produce better results. The Path Relinking algorithm adds little improvement to the good results obtained by GRASP procedures.

Tables 13 and 14 complement the information in previous Tables by providing the running times of the algorithms on both sets of problems. In all cases preprocessing is included as a part of the solution procedure. The algorithms have been coded in C++ and run on a Pentium IV at 2.8 Ghz. The basic GRASP algorithm stops after 2000 iterations, while the stopping criterion of the aggressive GRASP is set to 500 iterations. The average running times are very short, though some problems would require quite long times. Adding the Path Relinking procedure increases the running times very slightly and therefore it seems convenient to keep it in the final implementation. If we compare the running times of the basic and the aggressive GRASP procedures, we do not see large differences, except in the last line of Table 14. However, that is the case in which the results of the aggressive GRASP are more clearly superior to the basic algorithm and the larger computing time is efficiently used to obtain better results. Therefore, the aggressive GRASP algorithm with Path Relinking seems to be the best option for an efficient heuristic algorithm.

## VI. CONCLUSIONS

We have studied a generalization of the classical resource constrained project scheduling problem. A new type of resource is considered, the partially renewable resource in which the availability of the resource is associated to a given set of periods and the activities only consume it when they are processed in these periods. These resources can be seen as a generalization of renewable and non-renewable resources, but their main interest comes from their usefulness to model complex

situations appearing in timetabling and labor scheduling problems, which can be approached as project scheduling problems.

We have developed several preprocessing techniques which help to determine the existence of feasible solutions and to reduce the number of variables and constraints. We have also designed and implemented heuristic algorithms based on GRASP and Path Relinking. Preprocessing procedures and heuristic algorithms have been tested on two sets of instances previously proposed in the literature. They have been able to determine the feasibility status of many instances which up to now were undecided and to solve most of the feasible instances optimally.

We are convinced that the preprocessing techniques developed here should be used by any solution procedure, exact or heuristic, applied to this problem. Our heuristic algorithms are also very efficient and can be considered a useful tool for obtaining high quality solutions for the problem.

Future lines of research will be the development of an exact algorithm and the design of new heuristic algorithms for problems in which partially renewable resources are combined with classical renewable resources, as happens in real situations.

## Acknowledgements

This work has been partially supported by the Spanish Ministry of Science and Technology DPI2002-02553, and the Valencian Science and Technology Agency, GRUPOS03/174.

## REFERENCES

- [1] J. Böttcher, A. Drexl, R. Kolish, F. Salewski, Project Scheduling Under Partially Renewable Resource Constraints, *Management Science* 45 (1999) 544-559.
- [2] S. Chaudhuri, R.A. Walker, J.E. Mitchell, Analyzing and exploiting the structure of the constraints in the ILP approach to the scheduling problem, *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 2 (1994) 456-471.
- [3] N. Christofides, R. Alvarez-Valdes, J.M. Tamarit, Project scheduling with resource constraints: a branch and bound approach, *European Journal of Operational Research* 29 (1987) 262-273.
- [4] E.L. Demeulemeester, W.S. Herroelen, *Project Scheduling: A Research Handbook*, Kluwer Academic Publishers, Boston, 2002.
- [5] A. Drexl, R. Nissen, J.H. Patterson, F. Salewski, ProGen/ $\pi_x$  - An instance generator for resource-constrained project scheduling problems with partially renewable resources and further extensions, *European Journal of Operational Research* 125 (2000) 59-72.

Problem size		GRASP	GR + PR	AG - GR	AG - GR + PR
10	Average time	0.41	0.41	0.21	0.21
	Maximum time	30.4	30.5	45.8	45.9
15	Average time	1.34	1.35	1.25	1.25
	Maximum time	51.0	51.1	93.0	93.1
20	Average time	4.91	4.97	2.98	3.02
	Maximum time	180.8	186.6	154.9	155.2
25	Average time	8.85	8.91	6.73	6.76
	Maximum time	316.5	316.6	299.2	299.5
30	Average time	8.11	8.12	8.99	9.00
	Maximum time	455.6	455.6	457.7	457.8

TABLE XIII  
Running times of GRASP algorithms on Böttcher et al. problems

Problem size		GRASP	GR + PR	AG - GR	AG - GR + PR
10	Average time	0.89	0.90	1.05	1.05
	Maximum time	41.9	42.2	33.1	33.3
20	Average time	0.95	0.97	0.70	0.71
	Maximum time	162.4	163.0	53.0	53.1
30	Average time	2.03	2.10	2.04	2.11
	Maximum time	135.0	143.7	144.5	144.7
40	Average time	3.96	4.05	4.32	4.40
	Maximum time	155.6	155.8	507.5	519.8

TABLE XIV  
Running times of GRASP algorithms on Schirmer problems

- [6] P. Festa, M.G.C. Resende, GRASP: An annotated bibliography, in: M.G.C. Resende, P. Hansen (Eds.), Essays and Surveys in Metaheuristics, Kluwer Academic Press, Boston, 2001, pp. 325-367.
- [7] J.E. Kelley, Critical path planning and scheduling: Mathematical basis, *Operations Research* 9 (1961) 296-320.
- [8] R.Kolisch, A. Sprecher, A. Drexl, Characterization and generation of a general class of resource-constrained project scheduling problems, *Management Science* 41 (1995) 1693-1703.
- [9] C. Mellentien, C. Schwindt, N. Trautmann, Scheduling the factory pick-up of new cars, *OR Spectrum* (2004), in press.
- [10] K. Neumann, C. Schwindt, N. Trautmann, Advanced production scheduling for batch plants in process industries, *OR Spectrum* 24 (2002) 251-279.
- [11] K. Neumann, C. Schwindt, N. Trautmann, Scheduling of continuous and discontinuous material flows with intermediate storage restrictions, *European Journal of Operational Research* (2004), in press.
- [12] M.G.C. Resende, C.C. Ribeiro, Greedy Randomized Adaptive Search Procedures, in: F. Glover, G. Kochenberger (Eds.), State-of-the-art Handbook in Metaheuristics, Kluwer Academic Press, Boston, 2001, pp. 219-250.
- [13] A. Schirmer, Project Scheduling with Scarce Resources, Verlag Dr. Kovac, Hamburg, 2000.
- [14] C. Schwindt, N. Trautmann, Scheduling the production of rolling ingots: industrial context, model and solution method, *International Transactions in Operations Research* 10 (2000) 547-563.
- [15] J.D. Wiest, F.K. Levy, A management guide to PERT/CPM, Prentice-Hall, Englewood Cliffs, New Jersey, 1976.

# Evaluation of a hierarchical production planning and scheduling model for a tile company under different coordination mechanisms

M<sup>a</sup> del Mar Alemany\*, Eduardo Vicens<sup>†</sup>, Carlos Andrés<sup>‡</sup> and Andrés Boza<sup>§</sup>

\*Universidad Politécnica de Valencia/Organización de Empresas  
Camino de Vera s/n

Email: [mareva@omp.upv.es](mailto:mareva@omp.upv.es) (corresponding author)

<sup>†</sup>Universidad Politécnica de Valencia/Organización de Empresas  
Camino de Vera s/n

Email: [evicens@omp.upv.es](mailto:evicens@omp.upv.es)

<sup>‡</sup>Universidad Politécnica de Valencia/Organización de Empresas  
Camino de Vera s/n

Email: [candres@omp.upv.es](mailto:candres@omp.upv.es)

<sup>§</sup>Universidad Politécnica de Valencia/Organización de Empresas  
Camino de Vera s/n

Email: [aboza@omp.upv.es](mailto:aboza@omp.upv.es)

**Abstract**—Hierarchical Production Planning (HPP) is a philosophy that has been applied to several productive systems, however little literature exists on tile sector. Tile companies are multi-stage systems characterised by high sequence dependent setup times and high inventory levels. The existing research simplifies the problem considering the productive environment as constitute by a unique stage (usually the bottleneck one). However this approach introduces excessive simplifications. In this paper tile companies are identified as composed by several hybrid flow-shops. Then, a four level hierarchical production planning model is proposed for multiple hybrid flow-shops with high setup times. Hierarchical decomposition is evaluated for different coordination mechanisms for a tile company and results are reported.

**Keywords**—Hierarchical production planning, coordination mechanisms, hybrid flow shops, high setup times and tile companies.

## I. INTRODUCTION

HIERARCHICAL Production Planning (HPP) constitutes a classical approach to handle the decision-making process associated with the production planning of companies. HPP partitions the production planning problem into subproblems and assigned them to the different levels of the hierarchical organisation of the company. Each level of the hierarchy presents its own characteristics about planning time horizons and periods. HPP also aggregates and disaggregates the information through the various hierarchical levels. Coordination between levels must be established in order to ensure consistent decisions [40].

[3] introduces the concept of HPP. Examples of following theoretical works on this field are [5], [7], [8], [9], [13], [15], [25], [30] and [35].

But not only theoretical work has been developed around this topic, various HPP applications are reported in the literature such as: steel manufacturing ([10],[21],[22],[23]), metal can manufacturing ([28]), metal parts fabrication ([16]), shoe production ([11]), motor industry ([40]), detergent manufacturing company ([6]), milk powder manufacturing ([33]), furniture company ([18]), batch size production ([38]), multi-machine environment ([1997]), food production system ([39]), flexible automation workshops ([42]), supply chain management ([37]), and hierarchical production planning with demand constraints ([43]).

Though HPP has been applied to several productive systems, little literature exists on tile sector. One of the pioneering applications has been made by [20]. [27] deal with the issue of solving the capacitated and loading problem when parallel machines exist. [29] extend this later work taking into account the activation/deactivation of firings in a tile company.

These entire investigations simplify the problem considering the productive environment as constitute by a unique stage (the bottleneck stage,

usually the firing section). However this approach introduces excessive simplifications for certain productive systems that require a multi-stage treatment. For instance, one of the most important problems in tile sector is a high inventory level not only of final product but work-in-process. To control this two kinds of inventory it is necessary the inclusion of the multi-stage case.

On the other hand, due to the high personalization of final products to produce a great variety of products in the same line is required. This feature leads to a high numerous of setups to occur. Setup times are very important in tile industry (for example, setup times of around 20 hours exist). Therefore, to model the system as a multi-stage one, allows a major real estimation of the capacity consumption in every stage of the productive system as a consequence of setups. This fact receives a greater importance if, in addition, it is tried to determine normal and extra capacity of each stage ([1]).

The paper is organized as follows. First a tile company description is made and then it is identified as composed by several hybrid flow shops. A decisional problem classification is made and main contributions of the proposed model are outlined. A description of the proposed hierarchy to solve the problem is made as well as a description of the proposed models at each level. Then, different coordination mechanisms between levels are exposed. The proposed hierarchy is experimentally tested and results are reported. Finally, a set of considerations about obtained results are made.

## II. TILE COMPANY DESCRIPTION

Production of tiles is made from clay. Once clay is atomized it can follow two different production processes: single firing or double firing process. As a summary, in the single firing process the atomized clay is pressed forming pieces that are passed to glazing lines where it is come to the piece decoration through different applications. Then pieces are dried and taking into kilns to be cooked. Between the glazing lines section and the kilns section there are intermediate warehouses due to the different rate of production in each section. Once product is cooked it will be classified in different qualities by means of

sophisticated machines (sorters and packaged machines). At the same time, a worker analyzes the surface defects. The own sorters makes the cardboard boxes in which the product is packaged. A robot gathers these boxes and it stores them in pallets that are transported to the finished product warehouse. The product is ready already for its expedition. The double firing process would be almost the same unlike pressed pieces are cooked twice: before and later being processed by glazing lines. So the use of a process or another one mainly corresponds to the size of the format. Thus formats superior to 20x31 are processed by means of double firing line with the objective of give them a greater resistance before being processed in glazing lines.

Productive system layout of this company for single firing process is composed by four presses, four glazing lines, two kilns and three sorters. For double firing process it is composed by one press, one kiln, one glazing line, another kiln, and two sorters and packaged machines. There are two intermediate warehouses (buffers) and a warehouse for final products (figure 1).

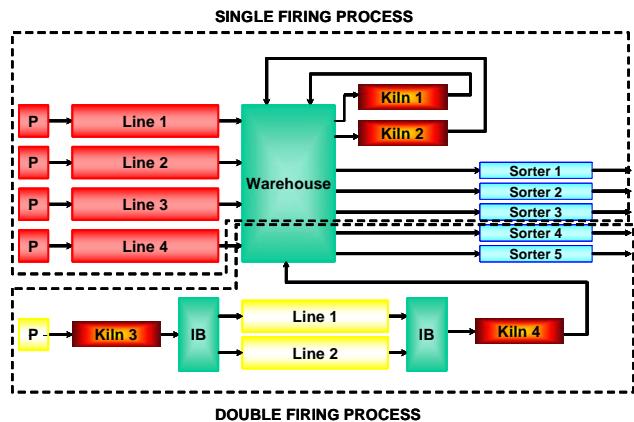


Figure 1. Productive system layout of tile company

In this company, two kinds of workforce can be distinguished: that dedicated to productive process and that dedicated to setups. With respect to the first class, due to the high automatization of tile companies a worker must manage several machines in the same section. Furthermore, workforce is polyvalent in such a manner that a worker can be transferred from one section to another. With respect to the second class, there is a maintenance team that attends setups of machines and maintenance work.

Some stages of the productive process work on shifts (presses and glazing lines, sorters and packages) but others work on a continuous basis (maximum number of shifts) or simply they are or not in operation (kilns).

When a change of model in one facility is performed, a maintenance team executes all setup operations. Six men compose it and each team will only work at one facility at a time. By watching setup operations, a setup time can be defined. It includes the time to:

1. Set presses. It consists of putting a new matrix, when changes in the tile size and shape are required.
2. Set the glazing lines. A glazing line is a conveyor where a set of machines is distributed in accordance with the product characteristics. The change consists of placing new machines, modifying their distribution if it is required, and adjusting them.
3. Prepare the kiln conditions. Modifying the distribution of heat and adjusting the carriers and conveyors.
4. Prepare the sorter conditions. Modifying the conveyor with.

Initially, when the catalogue of the tile companies was not very extensive, the productive system was very efficient since in the same line one or two models were made. Nevertheless, nowadays in a same glazing line (and due to its great automatization and cost), a great product diversity are made. To do this, it is necessary to change and fit the machines along the entire process. These setup times can be very important reaching an order of 20 hours. It is for that reason, that a suitable management becomes necessary to diminish the impact of setup times on capacity consumption of the productive system. A suitable management of setups would obtain advantages like: diminution of the number of setups and as a result of the costs associated to such including those of personnel, saving of run time of the master plan and, therefore, giving more operation flexibility.

Due to uncontrollable aspects of the productive process, the final product can present certain

deviations with respect to nominal dimension (calibre) as well as certain variability in colours (tone problem). This leads to a production of minimum lot sizes. This fact allows making necessary adjustments of controllable variables to correct possible defects and ensure product reaches the final section to be classified. On the other hand, due to the difficulty of obtaining a uniform final product of first quality, lot sizes are increased to avoid losses due to quality defects. These two aspects contribute to final product inventory increases.

The proposed hierarchical model of the following sections pretends to cover all this aspects, but at the same time, to be as general as possible. For this reason, the tile company is identified as composed by several hybrid flow-shops and the hierarchical model is formulated in terms of several hybrid flow-shops. With this approach we intend to extend the obtained results to productive systems with similar characteristics.

### III. IDENTIFICATION OF A TILE COMPANY AS COMPOSED BY TWO HYBRID FLOW-SHOPS

In order to identify the productive system layout of figure 1 as composed by several hybrid flow-shops the following assumption is made: those resources between there is no possibility of altering the sequence will be considered as a unique resource ([4]). This is the case for presses and glazing lines because they are connected by a conveyor. The same is valid for sorter and package machines. [32] define a hybrid flow shop as a configuration of  $m$  machines organized in  $r$  sections or stages where they process a series of  $n$  pieces. Each piece receives at most  $r$  operations (one in each stage or section). A section contains a set of  $m_r$  resources (machines) susceptible to execute the same operation. These are equivalent as far as its operation although they can not be it in its efficiency because the duration of an operation can depend on the resource chosen within a section. The resources can process an only piece simultaneously and each piece receives an only operation by stage. Between each stage there are buffers with finite capacity that can be common to several stages. The flow of the work is unidirectional from stage 1 to the stage  $r$ .

Stages share buffers (figure 1), but if one makes

as many copies of the shared buffers as stages make use of them, the logical flow of products can be represented as in figure 2. Productive system layout can be schematised as two hybrid flow-shops: the first one corresponds to the single firing process and it can be considered as a three stage hybrid flow-shop with four facilities at the first stage (press +glazing line), an intermediate buffer ( $b=1$ ), two facilities at the second stage (kilns) and three facilities at the third stage (sorter and packaged machines) and a warehouse of final products ( $b=3$ ). The second one corresponds to the double firing process and it is composed by four stages with one facility at the first stage (press and kiln), an intermediate buffer ( $b=2$ ), two facilities at the second stage (glazing lines), an intermediate buffer ( $b=2$ ), one facility at the third stage (kiln), an intermediate buffer ( $b=1$ ), two facilities at the fourth stage (sorter and packaged machines) a warehouse of final products ( $b=3$ ). Furthermore, these hybrid flow-shops share common elements as workforce to make setups (maintenance equipment) and buffers. For this particular case, decisions about capacity for stages 2, 4, and 6 (set EA) is reduced to activate or deactivate the kilns, meanwhile for the rest of stages (set NEA) the number of shifts (normal capacity) and extra capacity must to be established.

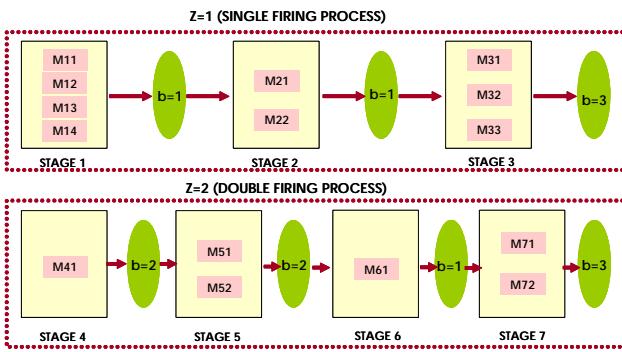


Figure 2. Identification of the tile company as composed by two hybrid flow-shops.

#### IV. DECISIONAL PROBLEM CLASSIFICATION AND CONTRIBUTIONS

A hierarchical model for the mid-term planning of manufacturing systems composed by several hybrid flow shops with high sequence dependent setup times is presented. For these systems to completely ignore setup times at aggregate level can lead to several problems related to feasibility

and performance of detailed plans, even more when capacity dimension of the productive system has to be made for tactical planning. In order to represent more accurately important setup times in a parallel processor environment it seems to be adequately to make mid term lot sizing decisions and short term loading decisions simultaneously. Then, the problem of concern to us consists of determining which products are produced by each machine (loading) of each stage and flow shop and their production quantity (lot sizing) to satisfy dynamic demand over a planning horizon. This problem belongs to the Capacitated Lot-Sizing and Loading Problem class (CLSLP). Several works dealing with the CLSLP for a unique hybrid flow shop exist ([2],[12],[27],[29]). However the modelization of the proposed problem introduces some new aspects:

1. There are several hybrid flow shops. Independently treatment of each flow shop is not possible, in principle, due to the existence of common elements (workers, buffers, etc.).
2. Setups have a very important capacity consumption of two resources classes: those dedicated to production and those dedicated to setups.
3. Lot sizes should be assigned within available capacity limits. Another new aspect is the treatment of the available capacity of production resources. The way of dimensioning production resources capacity depends on the productive stage:
  - o There are stages where capacity decisions are restricted to active or deactivate a machine in a period. That is, once activated it will be active continuously. This leads to the bin packing problem ([17]). These stages belong to the set named EA.
  - o Capacity decisions of the rest stages include determination of number of shifts (normal capacity dimension) and extra capacity. These stages belong to the set named NEA.
4. There is also a possibility of subcontracting final products (an unlimited source).
5. Stages are decoupled by capacity finite

- buffers which can be shared by several stages.
6. Minimum lot sizes are considered.
  7. Lot sizes augment is modelled because of uncertainty in faulty pieces.

Model supposes high sequence dependent setup times, but at the detail level considered sequence it is not known. In order to do not overestimate available capacity in making plans that can result infeasible at plant level, an estimation of independent sequence setup times is proposed. The methodology to do this presupposes a certain sequence through the definition of a four level hierarchical structure: items are aggregated into families, families into types which in turn are aggregated into lines of products. Furthermore, it is assumed that in a certain time period all products lot sizes belonging to the same family that are assigned to a given machine are jointly processed. The same is valid for all families belonging to the same type. Therefore, this especial treatment of setups is also considered a differentiating aspect.

Modelization of the above problem in one step results in a monolithic model belonging to the CLSPLP class and that can be seen in [1]. Furthermore model formulation includes types, families and products jointly, in order to allow properly modelling of sequence dependent setups in a model that does not take into account the product sequence. To reflect this fact the model has been named as Hierarchical Capacitated Multilevel Multi-flowshop Lot Sizing and Loading Problem (HCMLSLP). Mathematical formulation of the monolithic model is made in order to compare solution obtained by this model with its hierarchical decomposition (as it is proposed in the methodology of [41]).

## V. HIERARCHICAL DESIGN AND TYPES OF AGGREGATION

Hierarchical decomposition of detailed model establishes the existence of four levels: lines, types, families and products. The type of aggregation existing between two consecutive levels of the proposed hierarchy includes items, resources and time (figure 3).

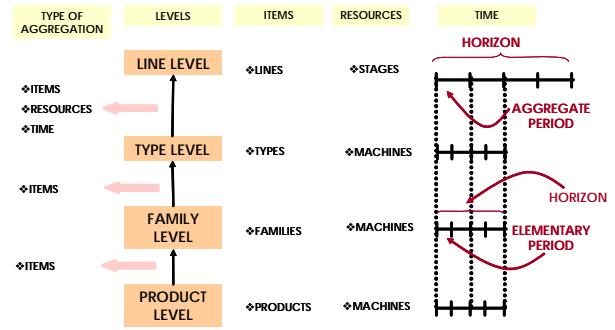


Figure 3. Type of aggregation between hierarchical levels.

In relation to the hierarchical product structure four levels are defined: lines of products, types, families and products. Criteria used to define the product hierarchical structure is based on two aspects: the productive process followed by a product and the necessary setups of machines for processing a product. Based on this, it is possible to define:

1. Lines of products: group of products that are processed by the same hybrid flow shop. For the tile company under study it is possible to establish the existence of two lines of products; those belonging to single firing process and those belonging to double firing process.
2. Types: group of articles belonging to the same line of products that share a common independent sequence setup of some machines. For the tile company, types correspond with formats and the common sequence independent setup takes place at presses, kilns, sorters and packages.
3. Families: group of articles that belong to the same type share a similar sequence dependent setup of some machines. For this particular case, families correspond with models that share a similar sequence dependent setup at the glazing lines.
4. Products: different articles provided to the customers. In this case, products correspond with the different existing models in the tile company.

For the tile company under study there are two lines of products (single firing process and double firing process). The first line is composed by three types (formats with different size) and the second one is integrated by six types. As it can be

observed as well as lines as types definition corresponds to a physical differentiation (process type and size, respectively). However, what products belong to which family is something that must be determined. This research supposes that aggregation process has been made and therefore families have been determined and it is based on research developed by [4].

Resource aggregation presents two levels:

1. Machine: resource able to process a set of items
2. Stage: set of machines able to make the same operation on the same set of items (perhaps with different processing and setup times).

For time aggregation, analysis of periodicity of decisions establishes two possible magnitudes for time periods:

1. Elementary period: it represents the time unity for the hierarchical system
2. Aggregate period: integer number of detailed periods that present a similar level of capacity consumption.

Usually definition of aggregate periods does not require none similarity measure. But this definition intends to avoid infeasibilities at lower levels during disaggregation process that lead to unfulfilled demand.

## VI. PROPOSED MODELS AT EACH LEVEL

As it has been mentioned before, planning hierarchy is composed by four levels (line, type, family and product). Decision making at each level is based on mathematical models. Only one mathematical model is associated at line and type level. However, at family level there are so many submodels as types exist. In the same way, at product level there are so many submodels as families exist. These independent submodels make more easily the solution process (one of the advantages of hierarchical philosophy).

Decisions at each level are not making simultaneously. Decisions based on line level are made at  $t_0$ , those of type level at  $t_1$ , and those of family and product level at  $t_2$  (being  $t_0 < t_1 < t_2$ ), therefore information about demand can not be the

same one. This is the reason why consistency conditions are not referred to production quantities but to planning inventory levels determined by upper levels. In contrast to traditional HPP, there is not so important planning aggregate production quantity as capacity decisions that must be making by upper levels.

Characteristics of different models are shown at Table I and Table II in the appendix.

## VII. COORDINATION MECHANISMS BETWEEN LEVELS

In the most general case, levels integrating a hierarchical system are connected by a top-down and a bottom-up influence ([37]). In case two level exist, the top-down influence is called instruction IN which is a function of the top-decision ( $a^T$ ):  $IN = IN(a^T)$ . The bottom-up influence is more complicated. It is, in principle, the anticipation of the base-level behavior which has to be taking into account by the top-level. If this anticipation does not depend on the instruction, it is called non-reactive. In the reactive case, however, the anticipation depends on IN and gives rise to the anticipation function  $AF = AF(IN)$ . The anticipation function describes the possible optimal reaction of the base-level. It can be viewed as an optimal anticipated (not necessarily realized) response to an impulse IN.

Based on the kind of coordination, four types on anticipation can be defined:

1. Pure top-down hierarchy: the top-level is assumed not to take into account any feature of the base-level.
2. Non-reactive anticipation: in contrast to the pure top-down hierarchy, the non-reactive anticipation accounts for important features of the base level's model. A reaction (to the instruction), however, is again not taking into account.
3. Reactive anticipation: The anticipation function is obtained by the top-level in anticipating and optimizing the entire base-level. The anticipated base-level model is simply obtained by the producer in replacing unknown parameters by their estimates.
4. Ideal model: for the ideal situation top-level and base-level are considered as one

decision maker.

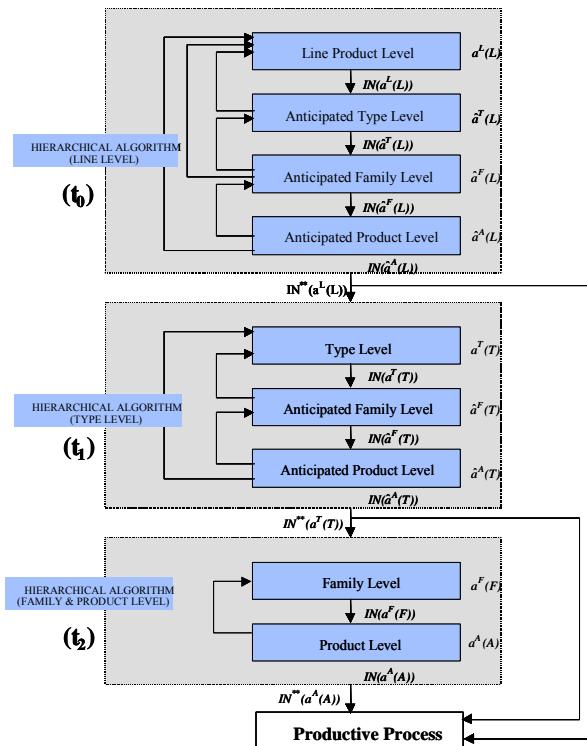


Figure 4. Hierarchical algorithms at each level for a reactive anticipation.

Traditionally, two of the main problems of the hierarchical systems have been to ensure feasibility of the detailed decisions and their optimality. To reduce these problems the non-reactive anticipation is replaced by the reactive one. In this kind of anticipation, the top-level before passing its decision take into account the lower level's behavior (base-level anticipation). To obtain an anticipated version of the lower levels, the top-level replaces their respective models by an estimation of them based on aggregate parameters and orders forecasting at the top-level decision instant. The anticipated version of lower levels is solved in order to anticipate possible infeasibilities. Solution of the lower levels is used to update aggregate parameters with the aim of give solutions nearer to the optimal one. This process is repeated until a stop criterion is achieved. This process is implemented for each hierarchical level giving as a result what is called hierarchical algorithm (Figure 4). It is necessary to define a hierarchical algorithm for a given set of levels in case the decisions are made in different points of time and/or by different decision-makers. This is the reason because an unique hierarchical algorithm is defined for the

family and product level.

## VIII. HIERARCHY EVALUATION

In order to evaluate the quality of the proposed hierarchy a computational study has been developed following the methodology of [41].

### A. Experimental Design

In order to have a performance measure of the planning hierarchy a computational study has been developed. The methodology followed is composed by the following stages:

- 1.-Objective definition
- 2.-Selection of performance measures and experimental factors
- 3.-Design and execution
- 4.-Results analysis and conclusions

#### 1) Objective definition

Though an optimal solution of monolithic model was possible, a hierachal approach is preferable because it reduces the computational complexity, the needed of required information and its associated uncertainty and furthermore it presents a parallelism with the organizational structure. However, the following analysis of computational results pretends to evaluate the quality of solutions obtained from the hierarchical decomposition in contrast to those obtained from the monolithic model. Therefore, it is supposed that mathematical formulation of monolithic model has been developed and solved [1].

#### 2) Selection of performance measures and experimental factors

##### Performance measures

##### **Feasibility**

A first stage to measure the quality of solutions obtained through the planning hierarchy is to verify the feasibility of the solutions. Then feasibility is the first performance measure. A solution of the hierarchy is considered feasible if it respects all the restrictions of the monolithic model. Infeasibilities that can arise during hierarchical process are of two kinds:

- ◆ First type infeasibilities: those that

appear during the disaggregation process due to the impossibility of solving some of the hierarchical models. In this case a detailed solution cannot be obtained.

- ◆ Second type infeasibilities: though a detailed solution is obtained from the execution of the planning hierarchy, it is not feasible at plant level. That is, the solution does not respect some of the monolithic model restrictions.

As much if infeasibility of first type like of second type exists the value of feasibility performance measure will be one. A value of zero for feasibility means that the solution is feasible.

### Percentage Deviation

The following performance measure pretends to taking into account not only quality of solutions but computational effort through the definition of execution times of models. Definition of original percentage deviation is the following:

$$DPO = \frac{FO(t) - FO_{mon}(t')}{FO_{mon}(t')} * 100$$

(1)

where:

$t$  → execution time of the planning hierarchy  
 $t'$  → execution time of the monolithic model.  
 $FO(t)$  → objective function value of monolithic model obtained through the substitution of the values of detailed decision variables from the hierarchical model (executed during a time  $t$ ).  
 $FO_{mon}(t')$  → objective function value of monolithic model obtained through the monolithic model (executed during a time  $t'$ ).

Execution times are the result of a calibration process that considers the difficulty of solving optimally monolithic model (even the difficulty of obtaining a feasible solution) and some hierarchical levels (in particular type level). To ensure at least one solution to the monolithic model the following condition is established:  $t' \gg t$ .

### Calculation process of performance measures: Feasibility and Percentage Deviation (DP)

Mathematical models of the planning hierarchy and monolithic model have been translated to MPL language and have been solved by CPLEX package. Once randomly generated detailed data has been aggregated, hierarchy is executed (figure 5). If there is not a first type infeasibility disaggregate decision variable values obtained from the hierarchy solution are replaced in monolithic model. Its resulting objective function value will be  $FO(t)$ . Though there was a second type infeasibility (that is, some monolithic restrictions are violated) an objective function value is obtained. However, as it will be seeing later a penalization must be introduced). On the other hand, the same detailed input data is transferred to monolithic model which is solved during  $t'$  obtaining  $FO_{mon}(t')$ . At this point it is possible to calculate DPO performance measure.

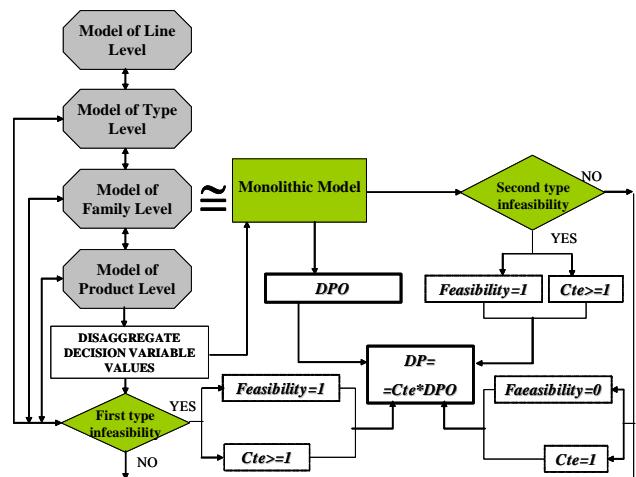


Figure 5. Calculation process of performance measures

Two cases can occur:

1.- Second type infeasibility does not exist: then “feasibility” is equal to 0 and “percentage deviation” (DP) is equal to “original percentage deviation” (DPO), because a penalization is not required.

2.- Second type infeasibility exists: then “feasibility” is equal to 1 and “percentage deviation” (DP) is equal to “original percentage deviation” (DPO) multiplied by a factor (Cte) major than one to penalize this fact. In this study  $Cte=10$ .

But during the resolution process it is also possible that first type feasibility appears when solving some hierarchical level. In this case more

capacity is reserved at upper levels to restore feasibility and planning hierarchy is solved again until a disaggregate decision variable values are obtained. The remaining of the process will be identical but in this case “feasibility” is equal to 1 and “percentage deviation” (DP) is equal to “original percentage deviation” (DPO) multiplied by a factor (Cte) major than one to penalize this fact. In this study Cte=10.

### 3) Experimental factors

Hierarchical evaluation is based on problems randomly generated. Computational design factors can be seen at figure 6. Each factor has two levels.

Factors	Levels	Level Values
Hierarchical complexity of references	Simple Complex	(see figure) (see figure)
Type of coordination mechanism	No reactive anticipation Reactive anticipation	
Process time dispersion	Low Level High Level	[L.I.(pr), 1.1 * L.I.(pr)] [L.I.(pr), 2 * L.I.(pr)]
Setup time dispersion	Low Level High Level	[L.I.(s), 1.1 * L.I.(s)] [L.I.(s), 2 * L.I.(s)]
Setup cost dispersion	Low Level High Level	[L.I.(cs), 1.1 * L.I.(cs)] [L.I.(cs), 2 * L.I.(cs)]
Other reference cost dispersion	Low Level High Level	[L.I.(c), 1.1 * L.I.(c)] [L.I.(c), 2 * L.I.(c)]
Capacity demand	Low Level High Level	70% of maximum lines capacity 120% of maximum lines capacity

Figure 6. Factors and levels of the computational study

Factor “hierarchical complexity of references” has two levels. The “simple” hierarchy supposes that there is not any reference type aggregation because each reference group of a level is composed by a unique reference of the lower level. However, “complex” hierarchy includes reference aggregation at all levels (figure 7).

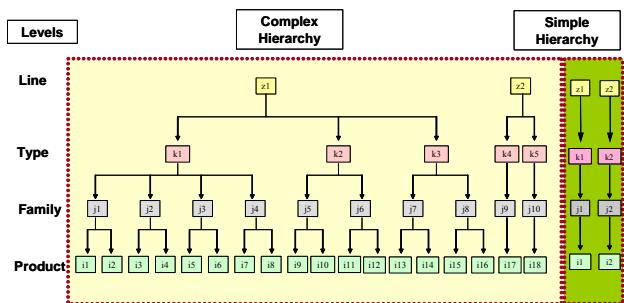


Figure 7. Values of each level of factor “Hierarchical complexity of references”

Factor “type of coordination mechanism” has two levels that evaluate interdependence between levels based on a non reactive anticipation and based on a reactive one.

Following factors (except “capacity demand”) has two levels that measure the dispersion degree of detailed data to be aggregated. “Low” level generates detailed data between a lower limit for the considered factor and an upper limit equal to 1.1 per lower one. That is, the dispersion of data is reduced. However, “high” level generates detailed data between the same lower limit for the considered factor as the “low” level and an upper limit equal to 2 per lower one. That is, the dispersion of data is ample.

Finally, low level for “capacity demand” supposes the necessary capacity to be 70% from the maximum available one. High level supposes the necessary capacity to be 120% from the maximum available one (in this last case a feasible solution is possible because external capacity is unlimited).

### 4) Design and execution

To evaluate planning hierarchy several problems with different detailed data based on real case of a tile company have been generated. For these problems there are fixed data, for example productive layout (figure 1), but other input data (experimental factors) vary from one problem to another. This way there are generated  $2^7$  problem types. Each combination of factors has been three times repeated.

The computational experiments were conducted on a PC with an AMD 1300 processor with a 512 MB memory. Models have been translated to a MPL language and solved with CPLEX package. Application developed has been programmed in Delphi 5.0

### B. Result analysis and conclusions

To design and analyse computational results the statistic package Statgraphics ® 5.0 has been used. From the variance analysis for “feasibility” (figure 8) can be deducted that only “type of coordination mechanisms” and “setup time dispersion” are statistically meaningful (a P-value lower than 0.05).

From the figure 9 it can be deducted that to implement a reactive anticipation between levels

highly reduce the infeasibilities (not only during the disaggregation process but at plant level). On the other hand (figure 10), more feasible solutions are obtained when the “setup time dispersion” is at low level. This fact shows the relevance of properly estimating capacity due to setups in a production environment with high setup times and of grouping references with similar setups of machines (low setup time dispersion).

Analysis of Variance For Factibilidad - Experimento jerarquía					
Source	Sum of Squares	DF	Mean Square	F-Ratio	P-Value
A:jerarquía	0,0416667	1	0,0416667	0,77	0,3794
B:anticipación	1,76892	1	1,76892	32,73	0,0000
C:proc	0,166667	1	0,166667	3,18	0,0792
D:tsetup	1,04167	1	1,04167	19,37	0,0000
E:capacidad	0,09375	1	0,09375	1,74	0,1876
F:tsetup	0,09375	1	0,09375	1,74	0,1876
G:cotros	0,0104167	1	0,0104167	0,19	0,6601
AB	0,0416667	1	0,0416667	0,77	0,3794
AC	0,0104167	1	0,0104167	0,19	0,6601
AD	0,09375	1	0,09375	1,74	0,1876
AE	0,0416667	1	0,0416667	0,77	0,3794
AF	0,0416667	1	0,0416667	0,77	0,3794
AG	0,166667	1	0,166667	3,18	0,0792
BC	0,166667	1	0,166667	3,18	0,0792
BD	1,04167	1	1,04167	19,37	0,0000
BE	0,09375	1	0,09375	1,74	0,1876
BF	0,09375	1	0,09375	1,74	0,1876
BG	0,0104167	1	0,0104167	0,19	0,6601
CD	0,0104167	1	0,0104167	0,19	0,6601
CE	0,0104167	1	0,0104167	0,19	0,6601
CF	0,0416667	1	0,0416667	0,77	0,3794
CG	0,0416667	1	0,0416667	0,77	0,3794
DE	0,0416667	1	0,0416667	0,77	0,3794
DF	0,0	1	0,0	0,00	1,0000
DG	0,0416667	1	0,0416667	0,77	0,3794
EF	0,0104167	1	0,0104167	0,19	0,6601
EG	0,09375	1	0,09375	1,74	0,1876
FG	0,0104167	1	0,0104167	0,19	0,6601
blocks	0,114583	5	0,0229167	0,43	0,8304
Total error	18,8229	358	0,0537798		
Total (corr.)	24,2396	383			

R-squared = 22,3464 percent  
R-squared (adjusted for d.f.) = 16,2216 percent  
Standard Error of Est. = 0,231905  
Mean absolute error = 0,123318  
Durbin-Watson Statistic = 2,01805 (P=0,4209)  
Lag 1 residual autocorrelation = -0,00982739

Figure 8. Analysis of variance for feasibility

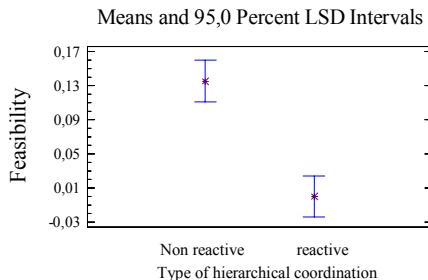


Figure 9. Fisher's test for type of hierarchical coordination (Feasibility)

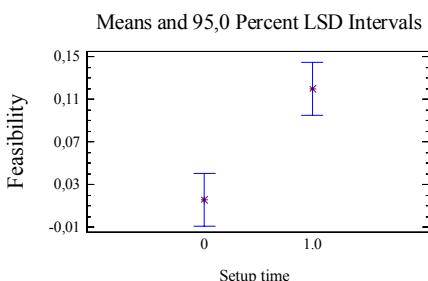


Figure 10. Fisher's test for setup time (Feasibility).

Only the double interaction “type of coordination mechanisms”/“setup time

dispersion” is statically meaningful. Implementing a reactive anticipation not only minimize infeasibilities to appear but leads the “setup time dispersion” to be irrelevant.

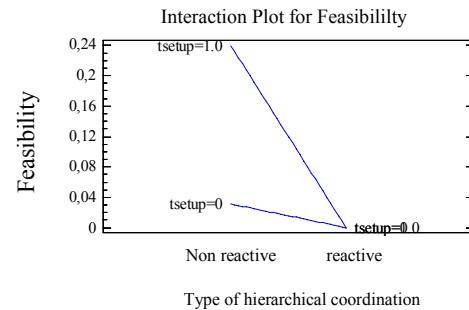


Figure 11. Graphic of double interaction between type of hierarchical coordination and setup time.

In case of “percentage deviation” variance analysis shows (figure 12) that only four of the seven factors and five double interactions are statistical meaningful.

Analysis of Variance For desv porc - Experimento jerarquía					
Source	Sum of Squares	DF	Mean Square	F-Ratio	P-Value
A:jerarquía	1,33996	1	1,33996	0,00	0,9446
B:anticipación	42383,4	1	42383,4	154,85	0,0000
C:proc	6091,13	1	6091,13	22,14	0,0000
D:tsetup	14428,5	1	14428,5	52,44	0,0000
E:capacidad	2268,09	1	2268,09	8,24	0,0043
F:tsetup	7,81098	1	7,81098	0,93	0,9663
G:cotros	9,59703	1	9,59703	0,00	0,9939
AB	12,1536	1	12,1536	0,00	0,9337
AC	977,14	1	977,14	3,55	0,0603
AD	467,554	1	467,554	1,70	0,1932
AE	1552,57	1	1552,57	5,64	0,0181
AF	30,0249	1	30,0249	0,11	0,7413
AG	589,798	1	589,798	2,14	0,1441
BC	1560,01	1	1560,01	46,63	0,0001
BD	2296,53	1	2296,53	8,35	0,0001
BE	1778,62	1	1778,62	0,46	0,9114
BF	230,786	1	230,786	0,84	0,3604
BG	65,1785	1	65,1785	0,24	0,6268
CD	261,535	1	261,535	0,95	0,3302
CE	1,29551	1	1,29551	0,00	0,9453
CF	306,77	1	306,77	1,11	0,2925
CG	270,702	1	270,702	0,99	0,3119
DE	68,502	1	68,502	0,25	0,6181
DF	1335,93	1	1335,93	8,86	0,0282
DG	344,429	1	344,429	1,25	0,2640
EF	2,25955	1	2,25955	0,01	0,9278
EG	100,426	1	100,426	0,37	0,5461
FG	69,7851	1	69,7851	0,25	0,6148
blocks	711,981	5	142,397	0,52	0,8304
Total error	96295,1	359	275,129		
Total (corr.)	177521,8	383			

R-squared = 45,7557 percent  
R-squared (adjusted for d.f.) = 41,4773 percent  
Standard Error of Est. = 16,5878  
Mean absolute error = 9,51887  
Durbin-Watson Statistic = 2,03731 (P=0,83576)  
Lag 1 residual autocorrelation = -0,0215267

Figure 12. Analysis of variance for percentage deviation.

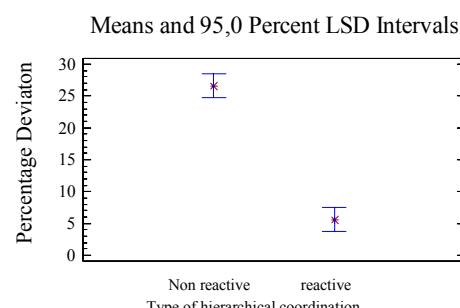


Figure 13. Fisher's test for type of hierarchical

coordination (DPO).

Results for “percentage deviation” are better when a reactive anticipation (Figure 13) is implemented and level of factors “process time dispersion” (Figure 14) and “setup time dispersion” (Figure 15) are low. These results are in concordance with traditional hierarchical approach of aggregate elements with similar characteristics. On the other hand, better results are obtained when capacity is at low level, that is, there is a surplus of available capacity in comparison with the needed one.

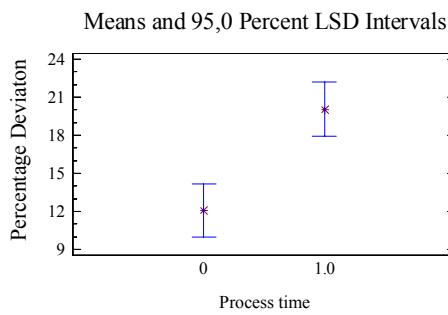


Figure 14. Fisher's test for process time (DPO).

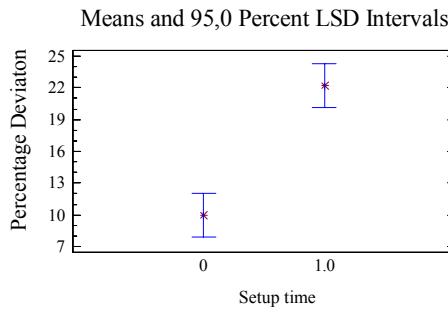


Figure 15. Fisher's test for setup time (DPO).

As it can be seen (Figure 16), exact estimation of capacity is crucial for obtaining better results from the hierarchical decomposition more when capacity is lean. In fact, there is not an influence of other factors like costs.

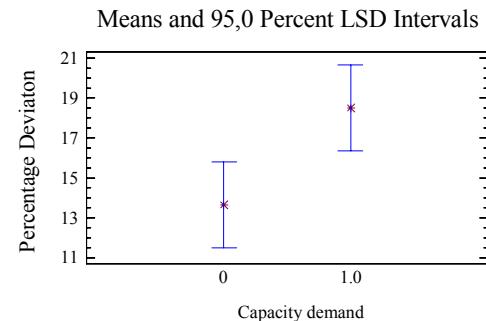


Figure 16. Fisher's test for capacity demand (DPO).

There are five double interactions. From the analysis of the three following interactions:

- type of coordination mechanisms/process time dispersion
- type of coordination mechanisms/setup time dispersion
- type of coordination mechanisms/capacity demand

From the figure 17, figure 18 and figure 19, it can be deducted that the reactive anticipation not only improve the results with respect to non-reactive one but reduce the difference between a high and low dispersion degree of the process time dispersion, setup time dispersions and high and low capacity demand. Therefore, when a reactive anticipation exists between levels the relevance of grouping entities with similar attributes is lower. This feature extends the real applicability of hierarchical systems because it is not necessary that the strict requirements for aggregation are accomplished.

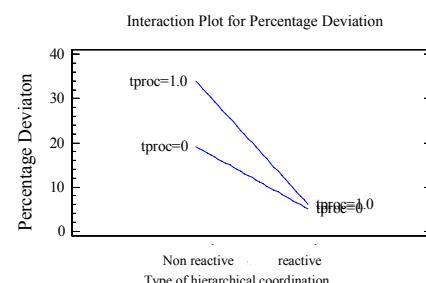


Figure 17. Graphic of double interaction between type of hierarchical coordination and process time.

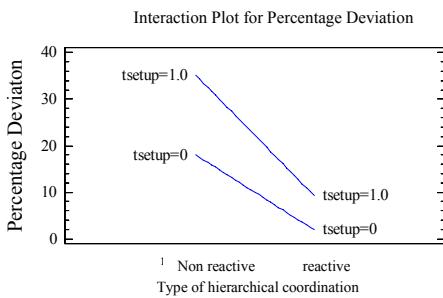


Figure 18. Graphic of double interaction between type of hierarchical coordination and setup time.

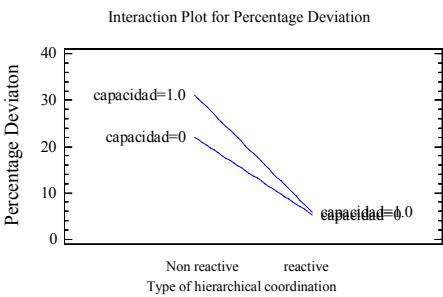


Figure 19. Graphic of double interaction between type of hierarchical coordination and capacity demand.

For the double interaction “setup time dispersion” and “setup cost dispersion” (Figure 20) results are better when “setup time dispersion” is low. For the two levels of “setup time dispersion” better results are obtained when levels of “setup time dispersion” and “setup cost dispersion” are opposed. This seems to be logic because loading and lot sizing decisions are easily if a group of references with similar setup time have very different setup costs and vice versa.

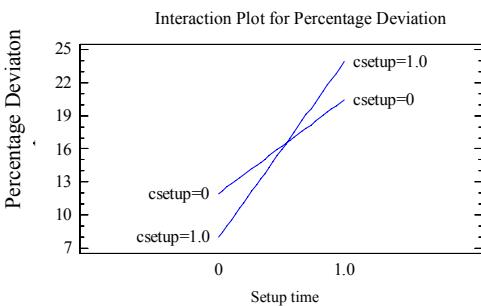


Figure 20. Graphic of double interaction between setup time and setup cost.

Finally, for double interaction between “hierarchical complexity of references” and “capacity demand” (Figure 21) best results are obtained when capacity demand level is low and

hierarchical complexity of references is simple.

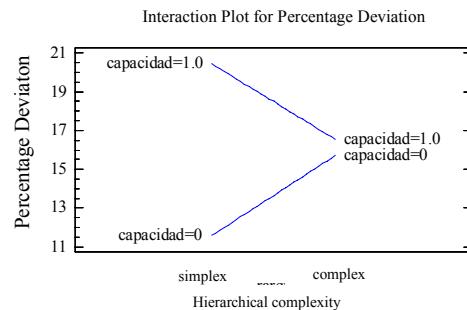


Figure 21. Graphic of double interaction between hierarchical complexity and capacity demand.

## IX. FINAL CONSIDERATIONS

A hierarchical production planning model has been proposed for supporting decision making on loading and lot sizing for hybrid flow shops with high sequence dependent setup times. A suitable management of setup capacity consumption in these types of production environments is crucial for the performance and feasibility of detailed plans. Computational experiments have been conducted in order to analyse the influence of different factors in the feasibility and quality of solutions obtained by the hierarchical decomposition. To do this, it has been necessary to establish performance measures and a process to evaluate the hierarchy.

Results show that reactive anticipation reduce the possibility of obtaining infeasible solutions and improves the quality of solutions. Furthermore, though better results are obtained when low dispersion of factors exists, the reactive anticipation considerably reduce the difference with respect feasibility and optimality of solutions obtained in case the data dispersion was high. This fact increases the applicability of hierarchical systems to real cases where the accomplishment of certain requirements for a traditional suitable aggregation (conditions for perfect aggregation) results very difficult.

Results also show that the properly anticipation of capacity consumption due to high setup times is crucial for properly working of hierarchy. In this case, to implement a reactive anticipation and to group products with similar setup times considerably improve the results obtained.

## APPENDIX

TABLE I  
OBJECTIVES AND DECISIONS AT EACH LEVEL

t	Level	Objectives	Decisions
T <sub>0</sub>	Line	Minimization of: • Production costs • Subcontracting costs • Holding costs • Extra and Regular capacity costs • Shift costs	• aggregated inventory level of line products at each stage • planned production quantities of line products at each stage • planned subcontracted quantities of line products • Shift number of stages do not belonging to EA. • Extra capacity of stages do not belonging to EA.
T <sub>1</sub>	Type	Minimization of: • Holding costs • Setup costs of types • Activation/deactivation n costs of machines belonging to EA • costs to maintain active machines belonging to EA	• aggregated inventory level of types at each stage • planned production quantities of types at each stage • planned subcontracted quantities of types • lot sizing and loading of types to machines • number of type setups • planned number of product setups of each type. • disaggregated extra capacity of machines do not belonging to EA. • activation/deactivation of machines belonging to EA
T <sub>2</sub>	Family	Minimization of: • Holding costs • Setup costs of families (including products)	• lot sizing and loading of families to machines • planned subcontracted quantities of each family • number of family setups • planned number of product setups of each family
T <sub>2</sub>	Product	STEP1 • Equalizing run out times  STEP 2 • Minimization of unfavourable deviations of upper level decisions	STEP1 • Quantity produce internally of final products for time horizon of product level (EROT quantities)  STEP 2 • lot sizes of products • loading of product lot sizes on individual machines • inventory of work in process and final products • number of product setups

TABLE II  
NUMBER OF PROBLEMS AND CONSISTENCY CONDITIONS AT EACH LEVEL

t	Level	Number of problems	Consistency Conditions
T <sub>0</sub>	Line	One	
T <sub>1</sub>	Type	One	• mid term capacity decisions of line level: ○ Number of shifts ○ Aggregated extra capacity • Subcontracted capacity • aggregate inventory level of line products • Erschler et al. (1986) rule.
T <sub>2</sub>	Family	So many as types	• Capacity constraints: total capacity consumption of all families belonging to each type must be less or equal than the capacity assigned to each type. • Subcontracted capacity • Aggregate inventory level of types
T <sub>2</sub>	Product	So many as families	• Total production quantity of family j for time horizon of product level
		So many as families	• Aggregate inventory level of families at each stage • Demand fulfilment of products • Planning number of setups belonging to each family • EROT quantity (from step 1) for each product.

## REFERENCES

- [1] M.M.E. Alemany, "Metodología y Modelos para el Diseño y Operación de los Sistemas de Planificación Jerárquica de la Producción", PhD, dissertation, Polytechnical University of Valencia, 2003.
- [2] E.H. Aghezzaf, and A. Artiba, "Aggregate Planning in Hybrid Flowshops", *International Journal of Production Research*, vol. 36, nº 9, pp. 2463-2477, 1998.
- [3] Anthony, "Planning and Control Systems: A Framework for Analysis", Harvard University, Graduate School of Business Administration, Division of Research, Boston, Massachusetts, 1965.
- [4] C. Andrés, "Programación de la Producción en Talleres de Flujo Híbridos con Tiempos de Cambio de Partida dependientes de la secuencia. Modelo, Métodos y Algoritmos de Resolución. Aplicación a Empresas del Sector Cerámico", PhD, dissertation, Polytechnical University of Valencia, 2001.
- [5] S. Axsäter, and H. Jönsson, "Aggregation and disaggregation in hierarchical production planning", *European Journal of Operational Research*, vol. 17, pp. 338-350, 1984.
- [6] G. Barbarosoglu, "Hierarchical Production Planning", in R.M. Burton y B. Obel, eds., *Design Models for Hierarchical Organizations: Computation, Information and Decentralization*, Kluwer Academic Publishers, pp. 181-206, 1995.
- [7] G.R. Bitran, E.A. Haas and A. C. Hax, "Hierarchical production planning: A single stage system", *Operations Research*, vol. 29, nº 4, pp. 717-743, 1981.
- [8] G.R. Bitran, E.A. Haas and A. C. Hax, "Hierarchical production planning: A two stage system", *Operations Research*, vol. 30, nº 2, pp. 232-251, 1982.
- [9] G.R. Bitran, and A.C. Hax, "Disaggregation and resource allocation using convex knapsack problems with bounded variables", *Management Science*, vol. 27, pp. 431-441, 1981.
- [10] M.R. Bowers, and J.P. Jarvis, "A Hierarchical Production Planning and Scheduling Model", *Decision Sciences*, vol. 23, pp. 144-159, 1992.
- [11] M.A. Caravilla, and J.P. De Sousa, "Hierarchical Production Planning in a Make-to-Order Company: A Case Study", *European Journal of Operational Research*, vol. 86, pp. 43-56, 1995.
- [12] A. Dumoulin, and C. Vercellis, 2000, "Tactical Models for Hierarchical Capacitated Lot-Sizing Problems with Set-ups and Changovers", *International Journal of Production Research*, vol. 38, nº 1, pp. 51-67, 2000.
- [13] J. Erschler, C. Fontan, and C. Merce, "Consistency of the disaggregation process in hierarchical planning", *Operations Research*, vol. 34, nº 3, pp. 464-469, 1986.
- [14] A.C. Hax, and D. Candea, "Hierarchical Integration of Production Planning and Scheduling", in: *Studies in the Management Sciences*, M. A. Geisler, eds., Logistics, North Holland, American Elsevier, 1975.
- [15] A.C. Hax, and D. Candea, *Production and Inventory Management*, Prentice Hall, Englewood Cliffs N.J., 1984
- [16] E. Iakovou, K. Malik, and A. Muckstadt, "A hierarchical approach for metal parts fabrication", *International Journal of Production Research*, vol. 33, nº 5, pp. 1257-1274, 1995.

- [17] D. S. Johnson, "Fast Algorithms for Bin Packing", *Journal of Computer and System Sciences*, vol. 8, pp. 272-314, 1974.
- [18] F.C. Lario, E. Vicens and L.R. McDonnell, "Application of an MRP matrix-based hierarchical planning model to a furniture company", *Production Planning & Control*, vol. 5, nº 6, pp. 562-574, 1994.
- [19] R. Leisten, "An LP-Aggregation view on aggregation in multi-level production planning", *Annals of Operations Research*, vol. 82, pp. 413-434, 1998.
- [20] M.J. Liberatore, and T. Miller, "A Hierarchical Production Planning System", *Interfaces*, vol. 15, pp. 1-11, 1985.
- [21] W. Lin. and C.L. Moodie, "Hierarchical Production Planning for a Modern Steel Manufacturing System", *International Journal of Production Research*, vol. 27, nº4, pp. 613-628, 1989.
- [22] G.T. Makulak, C.L. Moodie, and T.J. Williams, 1980, "Computerized Hierarchical Production Control in Steel Manufacturing", *International Journal of Production Research*, vol. 18, pp. 455-465, 1980.
- [23] B.D. Neureuther, G.G. Polak, and N.R. Sanders, "A hierarchical production plan for a make-to-order steel fabrication plant", *Production Planning & Control*, vol. 15, nº3, pp. 324-335, 2004.
- [24] E. Nowiki, and C. Smutnicki, "The flow shop with parallel machines: A tabu search approach", *European Journal of Operational Research*, vol. 106, pp. 226-253, 1998.
- [25] L. Özdamar, A.Ö. Atli, and M.A. Bozyel, "Heuristic family disaggregation techniques for hierarchical production planning systems", *International Journal of Production Research*, vol. 34, nº 9, pp. 2613-2628, 1996.
- [26] L. Özdamar, and S.I. Birbil, "Hybrid Heuristics for the capacitated lot sizing and loading problem with setup times and overtime decisions", *European Journal of Operational Research*, vol. 110, pp. 525-547, 1998.
- [27] L. Özdamar, and A. Bozyel, "Simultaneous lot sizing and loading of product families on parallel facilities of different classes", *International Journal of Production Research*, vol. 36, nº 5, pp. 1305-1324, 1998.
- [28] L. Özdamar, N. Yetis, and A. Ö. Atli, "A modified hierarchical production planning system integrated with MRP : a case study", *Production Planning & Control*, vol. 8, nº 1, pp. 72-87, 1997.
- [29] L. Özdamar, and S.I. Birbil, "A hierarchical planning system for energy intensive production environments", *International Journal of Production Economics*, vol. 58, pp. 115-129, 1999.
- [30] K. Pienkosz, and E. Tockzylowski, "On aggregation of items in single-stage production systems with limited inventory levels", *Operations Research*, vol. 41, nº2, pp. 419-427, 1993.
- [31] M.M. Qiu, and E.E. Burch, "Hierarchical Production Planning and Scheduling in a Multi-product, Multi-machine Environment", *International Journal of Production Research*, vol. 35, nº11, pp. 3023-3042, 1997.
- [32] F. Riane, A. Artiba, and S.E. Elmaghraby, "A hybrid three-stage flowshop problem: efficient heuristics to minimize makespan", *European Journal of Operational Research*, vol. 109, pp. 321-329, 1998.
- [33] W.G.M.M Rutten, "Hierarchical Mathematical Programming for Operational Planning in a Process Industry", *European Journal of Operational Research*, vol. 64, pp. 363-369, 1993.
- [34] M. Salomon, "Deterministic Lot-Sizing Models for Production Planning", in *Lecture Notes in Economics and Mathematical Systems*, Heidelberg: Springer-Verlag, 1991
- [35] C. Schneeweiss, "Hierarchical Planning in Organizations: Elements of a general theory", *International Journal of Production Economics*, vol. 56-57, pp. 547-556, 1998.
- [36] C. Schneeweiss, in *Hierarchies in Distributed Decision Making*, Ed. Springer-Verlag, Berlin-Heidelberg, 1999.
- [37] C. Schneeweiss, and K. Zimmer, "Hierarchical coordination mechanisms within the supply chain", *European Journal of Operational Research*, vol. 153, pp. 687-703, 2004.
- [38] V. Söhner, and C. Schneeweiss, "Hierarchically Integrated Lot Size Optimization", *European Journal of Operational Research*, vol. 86, pp. 73-90, 1995.
- [39] C.a. Soman, D.P. Van Donk, G. Gaalman, "Combined make-to-order and make-to-stock in a food production system", *International Journal of Production Economics*, in press.
- [40] H. Tsubone, and M. Sugawara, "A hierarchical production planning system in the motor industry", *OMEGA*, vol. 15, pp. 113-120, 1987.
- [41] E. Vicens, M.M.E. Alemany, C. Andrés, and J.J. Guarch, "A design and application methodology for hierarchical production planning decision support systems in an enterprise integration context", *International Journal of Production Economics*, vol. 74, pp. 5-20, 2001.
- [42] H. Yan, "Hierarchical stochastic production planning for flexible automation workshops", *Computers and Industrial Engineering*, vol. 38, pp. 435-455, 2000.
- [43] H. Yan, X.D. Zhang and J.A. Min, "Hierarchical production planning with demand constraints", *Computers & Industrial Engineering*, vol. 46, nº 3, pp. 533-551, 2004.

# A Restricted Median Location Model for Stop Location Design in Public Transportation Networks

Dwi Retnani Poetranto\*

\*Department of Mathematics, University of Kaiserslautern

67653 Kaiserslautern, Germany

Email: dwire@mathematik.uni-kl.de

**Abstract**—We consider the location of stops along the edges of an already existing public transportation network. This can be the location of bus stops along some given routes or railway stations along the tracks in a railway network. The goal is to minimize the total distance between demand points to their closest stops, with respect to the rectangular metric. The problem will be treated as a restricted location problem, where the restriction is given by the fact, that optimal locations have to lie on the transportation network. Two cases are of interest, completely redesigning the stops and opening additional stops. Using a reduction from the rectangular  $p$ -median problem, we show that this problem is  $\mathcal{NP}$ -hard. However, it is shown that if we want to establish only one stop then it is solvable in polynomial time. This result is used to develop a heuristic algorithm to solve the general problem where we want to establish more than one stop. Two heuristic strategies are proposed and computational results with randomly generated test problems are presented.

**Keywords**—facility location, rectangular metric,  $p$ -median problem, public transportation

## I. INTRODUCTION

THE planning process in a public transportation company starts by designing and building a transportation network. This includes to establish stops or stations and bus routes or train tracks. As a result we obtain a *public transportation network* (PTN). Throughout this paper, we will use the following definition for the PTN.

**Definition 1.1: (Public Transportation Network)** [23] The Public Transportation Network (PTN) is a graph  $PTN = (V, E)$ , given by a set of stops or stations  $V$  and a set  $E$  of direct connections between them. We assume that the PTN is an undirected graph.

This paper deals with the problem of placing stops/stations along the edges of an already existing public transportation network, based on the previous result presented in [22].

Installing new stops in a network of public transport has both positive and negative effects. On one hand, it could improve the coverage of residential areas, give

more convenient access to public transportation, and shorten access times. But on the other hand, it causes costs for establishing new stops and longer travelling time (“stop & go”).

The trade-off between the positive and negative effects of stops was discussed in [11] as a part of a project with the largest German rail company (Deutsche Bahn). The negative effect for the travelling time is subtracted from the positive effect for the access time, and the goal is to find a location of stops that maximizes this difference.

Recently, another model representing this trade-off was developed in [24]. In this paper, the problem is treated as a bicriteria problem, where the goal is to find a location of stops that achieve a maximal covering of given demand points with a minimal number of stops.

As we see from the references mentioned above, the question of *where* to establish the new stops is very important in planning a transportation network. In the current paper we also put emphasize on this question. This is where location planning enters the picture. The search for a set of vertices in the PTN that are optimal for establishing new stops under appropriately quantified objective functions is a typical task in mathematical location planning. Many references can be found e.g. in [1]–[4], [9], [15], [20], [21], [27] and the references therein.

Our goal is to establish stops/stations that serve a given set of demand areas (which might represent e.g. settlement areas, shopping centers, schools, etc.) such that the total distance from the demand areas to their closest stops is minimal, with respect to the rectangular metric. For simplicity, we assume that demand areas are represented by points in the plane. We do not assume a given finite candidate set, but allow a continuous set of possible locations for the stops given by the current bus routes or railway tracks.

This problem, we call it *N-stops location problem* ( $N$ -SLP), is a variant of the continuous stop location problem defined in [25]. In this paper, Schöbel et al discussed the problem to locate stops in an already existing PTN to

cover all customers with a minimal amount of additional travelling time, where covering is defined with respect to an arbitrary norm (or even a gauge). For this they assume that the traffic load for each bus route or railway track is given. Assuming equal weights for all tracks, the goal reduces to minimizing the number of stops.

In this paper, they proved that the problem of opening additional stations is equivalent to the new planning of all stations. This is not the case for our model and we will consider both of them. However, there is a special case where these two problems are identical.

It will be shown that the continuous set of possible stops is reducible to one, where only a finite candidate set needs to be considered. We develop a heuristic algorithm based on a polynomial algorithm for solving a special case in which we want to establish only one stop (1-SLP).

The remainder of this paper is structured as follows. In the next section we present a mathematical description for our stop location model. A solution approach for 1-SLP using the construction line algorithm as proposed in [4], [9], [21] is discussed in Section III. Section IV discusses the model for general problems ( $N$ -SLP). Furthermore we prove the  $\mathcal{NP}$ -hardness of the problem. In Section V we propose two heuristic algorithms to solve  $N$ -SLP. Some results of our computational experience are reported in Section VI. Finally we conclude the paper by briefly introducing some possible extensions for further research in Section VII.

## II. PROBLEM DESCRIPTION

Let  $PTN = (V, E)$  be a given public transportation network and let  $\mathcal{P} = \{p_1, \dots, p_M\} \subset \mathbb{R}^2$  be a given finite set of demand points. Denote the index set  $\{1, \dots, M\}$  by  $\mathcal{M}$ . Each demand point  $p_m \in \mathcal{P}$  and each existing stop/station in PTN  $v_i \in V$  is represented by their coordinates in the plane,  $p_m = (p_{m1}, p_{m2})$ ,  $v_i = (v_{i1}, v_{i2})$ . The tracks are assumed to be a piecewise linear set in the plane.

Let  $\mathcal{S}$  be the set of potential new stops. Since we allow a continuous set of possible locations,  $\mathcal{S}$  can be derived as

$$\mathcal{S} := \bigcup_{e \in E} e.$$

We want to determine the location of at most  $N$  stops which is represented by a set of points  $S^* \subset \mathcal{S}$ .

Two problems are considered, completely redesigning all stops and opening additional stops.

In the first problem, we assume that the customers will be served only by the new stops. The location of the existing stops does not effect our decision, we only

consider the existing routes where we are allowed to build the new stops.

We look for  $S^* \subset \mathcal{S}$ ,  $|S^*| \leq N$  that minimizes the objective function:

$$\min f_1(S) = \sum_{m \in \mathcal{M}} l_1(p_m, S) \quad (1)$$

In the second model, the customers can go to both the existing and the new stops, whichever closer to them. For this model, the objective function is:

$$\min f_2(S) = \sum_{m \in \mathcal{M}} l_1(p_m, V \cup S) \quad (2)$$

The distance between one demand point  $p_m$  to a set  $S$  is defined as the distance between  $p_m$  to its closest point  $s \in S$ .

*Definition 2.1: (Cover)* Let  $\mathcal{P}$  be a set of demand points and  $S$  be a set of feasible stops. The *cover* of  $\tilde{s} \in S$  is

$$cover_{\tilde{s}, S} = \left\{ p_m \in \mathcal{P} : \tilde{s} = \operatorname{argmin}_{s \in S} l_1(p_m, s) \right\}$$

and demand point  $p_m \in \mathcal{P}$  is said to be covered by  $\tilde{s}$ .

If it is clear that the covering is defined with respect to the set  $S$ , we drop the index  $S$  and denote  $cover_s$  only by  $cover_{\tilde{s}}$ .

It can happen that the closest stop to a demand point is not unique. In this case, the question which station covers the corresponding demand point has to be answered by a tie breaking rule. We define our tie breaking as follows: customers always go to the first closest station found. This means  $cover_s$ ,  $s \in S$ , is a partition of the set of demand points  $\mathcal{P}$ .

## III. 1-STOP LOCATION PROBLEM

In this section we will discuss a solution approach for the special case where we want to establish only one new stop. We denote the new planning problem by 1-SLP1 and the additional stop problem by 1-SLP2. If we refer to the general problem, we use 1-SLP.

### A. New Planning of All Stops

Within this setting, the problem turns out to be a classical restricted 1-facility location problem.

$$\begin{aligned} \min f_1(s) &= \sum_{m \in \mathcal{M}} l_1(p_m, s) \\ &= \sum_{m \in \mathcal{M}} |p_{m1} - s_1| + |p_{m2} - s_2| \end{aligned} \quad (3)$$

We sort the first and second coordinates from  $p_m$  in non-decreasing order,

$$\begin{aligned} p_{m_1} &\leq p_{m_2} \leq \dots \leq p_{m_M} \\ p_{n_1} &\leq p_{n_2} \leq \dots \leq p_{n_N} \end{aligned}$$

In the case where  $p_{m_1} = p_{m_{i+1}} = \dots = p_{m_j}$  we omit  $p_{m_{i+1}}, \dots, p_{m_j}$ . The list of first coordinates then become  $\tilde{p}_{11}, \dots, \tilde{p}_{P1}$  ( $P \leq M$ ) where  $\tilde{p}_{11} < \dots < \tilde{p}_{P1}$ . Define a weight  $v_{i1} := |\{p_{m1} : p_{m1} = \tilde{p}_{i1}\}|$  for  $\tilde{p}_{i1}$ ,  $i = 1, \dots, P$ .

The list  $\tilde{p}_{12} < \dots < \tilde{p}_{Q2}$  and weight  $v_{12}, \dots, v_{Q2}$ , ( $Q \leq N$ ) with respect to the second coordinates of  $p_m$  are defined analogously.

Additionally we define  $\tilde{p}_{01} = \tilde{p}_{02} = -\infty$  and  $\tilde{p}_{P+1,1} = \tilde{p}_{Q+1,2} = \infty$ .

Now we get the decomposition of  $\mathbb{R}^2$  into rectangles

$$\begin{aligned} \langle i, j \rangle &:= \{(x_1, x_2) : \tilde{p}_{i1} \leq x_1 \leq \tilde{p}_{i+1,1}, \\ &\quad \tilde{p}_{j2} \leq x_2 \leq \tilde{p}_{j+1,2}\} \end{aligned}$$

for  $i \in \mathcal{P}_0 := \{0, 1, \dots, P\}$  and  $j \in \mathcal{Q}_0 := \{0, 1, \dots, Q\}$ .

**Definition 3.1: (Construction lines)** The lines determining the rectangles  $\langle i, j \rangle$ ,

$$\begin{aligned} \mathcal{K}_1 &:= \{x_1 = \tilde{p}_{m1} : m \in \{1, \dots, P\}\} \cup \\ &\quad \{x_2 = \tilde{p}_{n2} : n \in \{1, \dots, Q\}\} \end{aligned}$$

are called *construction lines*.

It can be shown now that it is sufficient to consider a finite set of candidates, in which we know that it contains an optimal solution  $s^*$ .

**Definition 3.2: (Candidate Set)**  $\mathcal{S}_{cand1} := V \cup \{\mathcal{S} \cap \mathcal{K}_1\}$  (see Figure 1)

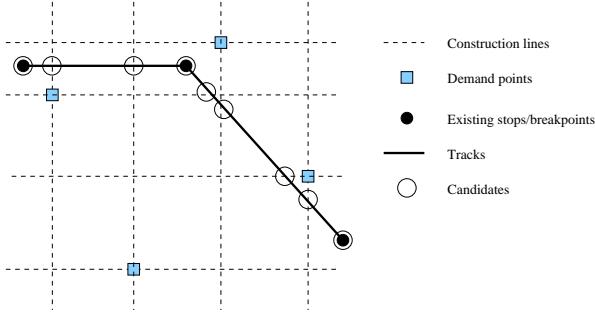


Fig. 1. Set of candidates along the tracks for 1-SLP1

**Theorem 3.3:**  $\exists s^* \in \mathcal{S}_{cand1}$  such that  $\forall s \in \mathcal{S} \sum_{m \in \mathcal{M}} l_1(p_m, s) \geq \sum_{m \in \mathcal{M}} l_1(p_m, s^*)$ .

**Proof:** As mentioned in Section II, we assume that our tracks are piecewise linear in a plane. Thus,  $s = (s_1, s_2) \in e \in \mathcal{S}$  satisfies the linear equation

$$s_2 = as_1 + b$$

for some  $a, b \in \mathbb{R}$ .

Along the track  $e \in \mathcal{S}$ , for  $s \in \langle i, j \rangle$ , we can rewrite (3) as:

$$\begin{aligned} f_1(s) &= \sum_{m=1}^i v_{m1}(s_1 - \tilde{p}_{m1}) + \\ &\quad \sum_{m=i+1}^P v_{m1}(\tilde{p}_{m1} - s_1) + \\ &\quad \sum_{n=1}^j v_{n2}(as_1 + b - \tilde{p}_{n2}) + \\ &\quad \sum_{n=j+1}^Q v_{n2}(\tilde{p}_{n2} - as_1 - b) \end{aligned} \quad (4)$$

This is a piecewise linear function with breakpoints at  $s_1 = \tilde{p}_{m1}$ ,  $m \in \{1, \dots, P\}$  and at  $as_1 + b = s_2 = \tilde{p}_{n2}$ ,  $n \in \{1, \dots, Q\}$ , i.e. on construction lines  $\mathcal{K}_1$ . Since  $f_1(s)$  is a convex function (see e.g. [15]), from the piecewise linearity it follows that its minimal value will be achieved either at breakpoints, i.e. at  $s^* \in \mathcal{S} \cap \mathcal{K}_1$ , or at  $s^* \in \{v_1, v_2\}$  if  $f_1(s)$  is defined in the interval  $[v_1, v_2]$ . Notice that  $v_1, v_2$  are elements of  $V$ . ■

Unfortunately, it may happen that  $|\mathcal{S} \cap \mathcal{K}_1| = \infty$ , i.e. there is a construction line  $\tilde{k}$  which coincides with  $\tilde{e} \in \mathcal{S}$  in infinitely many points. Using the structure of level curves, we will show in the following that in this case we do not need to consider  $\tilde{k}$  for the corresponding track  $\tilde{e}$ .

The level curves of  $f_1(s)$  are polygons, that are linear in rectangle  $\langle i, j \rangle$ . Let  $Opt^*$  be the optimal single facility in the *unrestricted* problem. Furthermore, we denote the level curve that goes through  $s \in \langle i, j \rangle$ ,  $s \notin Opt^*$  with  $L_=(f_1(s))$ . The gradient of level curves can be shown as in Figure 2 (see e.g. [4] or [9] for details).

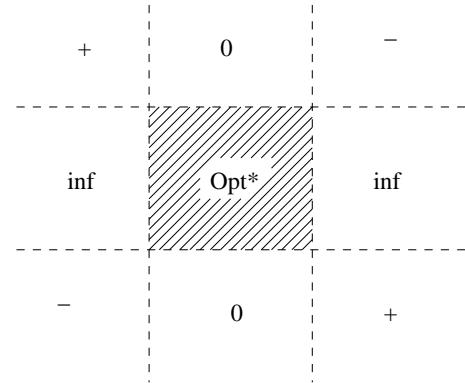


Fig. 2. Gradient of level curves

If  $s^*$  is an optimal solution for 1-SLP1, it is clear that  $L_=(f_1(s^*))$  will never cross the set of tracks  $\mathcal{S}$ ,

but it will only touch  $\mathcal{S}$  at certain points. In particular,  $L_=(f_1(s^*))$  will touch  $\mathcal{S}$  only at  $s^*$  itself, or in case  $s^*$  is not a unique optimal solution,  $L_=(f_1(s^*))$  touches  $\mathcal{S}$  at the set of optimal solutions  $S^*$ . This statement holds because the level set  $L_{\leq}(f_1(s^*))$  is convex. Therefore if there exists some  $\tilde{s}$ ,  $\tilde{s} \notin S^*$  so that  $\tilde{s} \in L_=(f_1(s^*)) \cap \mathcal{S}$ , then there would exist  $\hat{s} \in \mathcal{S}$ , i.e.  $\hat{s}$  feasible, such that the  $L_=(f_1(\hat{s}))$  is inside  $L_=(f_1(s^*))$ , or with other word  $f_1(\hat{s}) < f_1(s^*)$ . This contradicts the fact that  $s^*$  is an optimal solution. Illustration for this is shown in Figure 3.

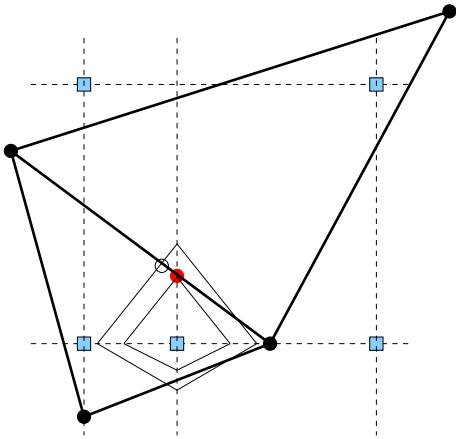


Fig. 3. Level curves through  $s \in e \in E$

This fact, combined with the fact that the gradient of a level curve changes only at construction lines, can be used to prove our claim before. This will be stated in the following lemma.

*Lemma 3.4:* If track  $\tilde{e} = [\tilde{v}_i, \tilde{v}_j] \in \mathcal{S}$  coincides with some construction line  $\tilde{k}$ , then it is not necessary to consider  $\tilde{k}$  for the corresponding track  $\tilde{e}$ .

Before we prove this, we will first give the interpretation of Lemma 3.4. The lemma says that it is sufficient to consider  $\tilde{v}_i, \tilde{v}_j$  and intersection points of  $\tilde{e}$  with other construction lines rather than  $\tilde{k}$  (as “normal” definition of candidate set). See Figure 4 for an illustration.

*Proof:* Let us take  $\tilde{s} \in \tilde{e}, \tilde{s} \neq \tilde{v}_i, \tilde{s} \neq \tilde{v}_j$  and  $\tilde{s} \notin \tilde{e} \cap \{\mathcal{K}_1 \setminus \tilde{k}\}$ . Since  $\tilde{e}$  coincide with construction line  $\tilde{k}$ , the gradient of level curve  $L_=(f_1(\tilde{s}))$  will change at  $\tilde{s}$ . But then  $L_=(f_1(\tilde{s}))$  would cross  $\tilde{e}$ , thus  $\tilde{s}$  can not be optimal. ■

After establishing that  $\mathcal{S}_{cand1}$  is, indeed a finite set, and moreover there exists some  $s^* \in \mathcal{S}_{cand1}$  which is an optimal solution for 1-SLP1, we get the following algorithm :

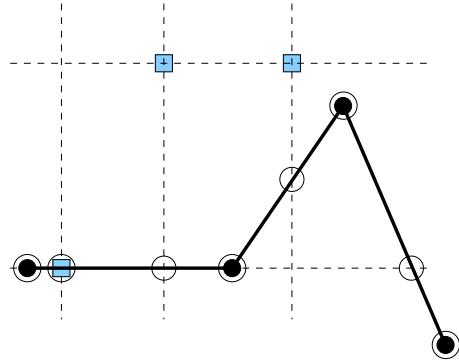


Fig. 4. Candidate set for a 1-SLP1 where a track lies upon a construction line

**Alg.1 :** Construction Line Algorithm for 1-SLP1

**Input :**  $PTN = (V, E)$ , demand points  $\mathcal{P} = \{p_1, \dots, p_M\} \subset \mathbb{R}^2$ .

**Output :**  $s^*$  optimal single stop for 1-SLP1.

**Step 1 :** Compute construction lines  $\mathcal{K}_1$ .

**Step 2 :** Determine the set of candidates  $\mathcal{S}_{cand1} = V \cup \{S \cap \mathcal{K}_1\} = \{s_1, \dots, s_L\}$ .

**Step 3 :** Let  $s^* = \operatorname{argmin} \{f_1(s_1), \dots, f_1(s_L)\}$

**Step 4 :** Output:  $s^*$

Notice that the construction line algorithm to solve 1-SLP1 is an algorithm with polynomial time complexity. If  $S$  is the number of existing stops and  $R$  is the number of tracks/rails in our PTN<sup>1</sup>, then the candidates  $s_1, \dots, s_L$  can be determined in  $\mathcal{O}(S + MR)$ . The effort to compute objective value for each candidates is  $\mathcal{O}(M)$  and since  $L \leq S + 2MR$  we need to make at most  $S + 2MR - 1$  comparison to find the minimum. Thus the overall complexity of the algorithm is  $\mathcal{O}(M^2R)$ .

### B. Opening Additional Stations

In this setting, we consider both the existing stops and the new one. We will see later that several results of the previous model, with some modifications, can be carried over to this model.

We start by explaining our solution idea graphically, which is shown in Figure 5.

For each demand point  $p_m, m \in \mathcal{M}$ , we can determine the existing stop that cover  $p_m$ . Let us denote this cover by  $v_m$  and the distance between them by  $d_m$ .

$$\begin{aligned} v_m &:= \operatorname{argmin}_{v \in V} l_1(p_m, v) \\ d_m &:= l_1(p_m, v_m) \end{aligned}$$

<sup>1</sup> $S = |V|, R = |E|$ , where  $V$  is the set of vertices and  $E$  is the set of edges in  $PTN = (V, E)$

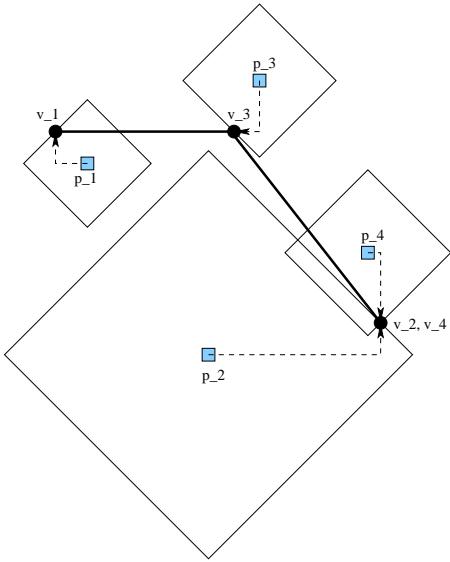


Fig. 5. Graphical solution idea

For  $s \in \mathbb{R}^2$ , we have to decide whether  $p_m$  is closer to  $s$  or to the original cover  $v_m$ . This is easily done with the help of a rectilinear circle about  $p_m$  with radius  $d_m$ . If  $s$  lies inside of the circle then  $p_m$  will be covered by  $s$ , otherwise it is covered by  $v_m$  and the objective value is not improved. Therefore, we only need to consider the points that lie inside of at least one circle.

*Remark 3.5:* Notice that

$$f_2(v) = \sum_{m \in \mathcal{M}} d_m, \forall v \in V.$$

Moreover,

$$f_2(s) \leq f_2(v), \forall s \in \mathbb{R}^2.$$

Equality holds if  $s \in \emptyset$  or  $s \notin D$ , where

$$\begin{aligned} D &:= \bigcup_{m \in \mathcal{M}} D_m, \\ D_m &:= \{s \in \mathbb{R}^2 : l_1(p_m, s) < d_m\}, m \in \mathcal{M}. \end{aligned}$$

From Remark 3.5, we see that the level curve  $L_=(z) = \emptyset$  for  $z > f_2(v)$  and  $L_=(f_2(v)) = \mathbb{R}^2 \setminus D$ . Furthermore, the level set  $L_<(f_2(v)) = \mathbb{R}^2$ . The level set  $L_<(f_2(s))$  is in general not convex. As a consequent, the objective function  $f_2(s)$  is not convex either. However, it will be shown later that  $f_2(s)$ , like  $f_1(s)$ , is a piecewise linear function along the tracks  $e \in \mathcal{S}$ . This fact can be used to establish a finite candidate set that contains a global optimal solution for 1-SLP2.

Let us introduce some other notations that we will use throughout this paper.

$$\mathcal{M}_s := \{m \in \mathcal{M} : s \in D_m\} \subseteq \mathcal{M}.$$

For each  $p_m \in \mathcal{P}$ , define the open line segments:

$$\begin{aligned} k_{m1} &:= \{(x_1, x_2) \in \mathbb{R}^2 : x_1 = p_{m1}, |p_{m2} - x_2| < d_m\} \\ k_{m2} &:= \{(x_1, x_2) \in \mathbb{R}^2 : |p_{m1} - x_1| < d_m, x_2 = p_{m2}\}. \end{aligned}$$

*Definition 3.6: (Construction lines)* Construction lines for 1-SLP2 are defined as follows:

$$\mathcal{K}_2 = \{k_{m1} : m \in \mathcal{M}\} \cup \{k_{m2} : m \in \mathcal{M}\}$$

These lines are basically the same lines as in Definition 3.2, but they are only “active” in  $D_m$ .

The construction lines  $k_{m1}$  and  $k_{m2}$  decompose  $D_m$  into four quadrants:

$$D_{m1} := \{(x_1, x_2) \in D_m : x_1 \geq p_{m1}, x_2 \geq p_{m2}\}$$

$$D_{m2} := \{(x_1, x_2) \in D_m : x_1 < p_{m1}, x_2 \geq p_{m2}\}$$

$$D_{m3} := \{(x_1, x_2) \in D_m : x_1 < p_{m1}, x_2 < p_{m2}\}$$

$$D_{m4} := \{(x_1, x_2) \in D_m : x_1 \geq p_{m1}, x_2 < p_{m2}\}$$

Using these notations, we can rewrite the objective function (2) as:

$$f_2(s) = \sum_{m \in \mathcal{M}} l_1(p_m, V \cup \{s\}) \quad (5)$$

$$= \sum_{m \in \mathcal{M}} \min \{d_m, l_1(p_m, s)\} \quad (6)$$

$$= \sum_{m \in \mathcal{M} \setminus \mathcal{M}_s} d_m + \sum_{m \in \mathcal{M}_s} l_1(p_m, s) \quad (7)$$

$$\begin{aligned} &= \sum_{m \in \mathcal{M} \setminus \mathcal{M}_s} d_m + \\ &\quad \sum_{m:s \in D_{m1}} ((s_1 - p_{m1}) + (s_2 - p_{m2})) + \\ &\quad \sum_{m:s \in D_{m2}} ((p_{m1} - s_1) + (s_2 - p_{m2})) + \\ &\quad \sum_{m:s \in D_{m3}} ((p_{m1} - s_1) + (p_{m2} - s_2)) + \\ &\quad \sum_{m:s \in D_{m4}} ((s_1 - p_{m1}) + (s_2 - p_{m2})) \quad (8) \end{aligned}$$

From (8), we see that  $f_2(s)$  are linear in each  $D_{mi}$ ,  $m \in \mathcal{M}$ ,  $i = 1, \dots, 4$ .

*Definition 3.7: (Candidate Set)*  $\mathcal{S}_{cand2} := \mathcal{S} \cap \mathcal{K}_2$  (see Figure 6).

*Theorem 3.8:* The optimal solution for 1-SLP2 is contained in  $\mathcal{S}_{cand2}$ .

*Proof:* Since the tracks are assumed to be linear in the plane,  $f_2(s)$  are piecewise linear along the tracks with breakpoints at construction lines,  $\mathcal{K}_2$ , and at the boundaries of  $D_m$ ,  $\partial D_m$ . A piecewise linear function attains its minimal value either at a breakpoint or at the boundary of the interval on which the function is defined.

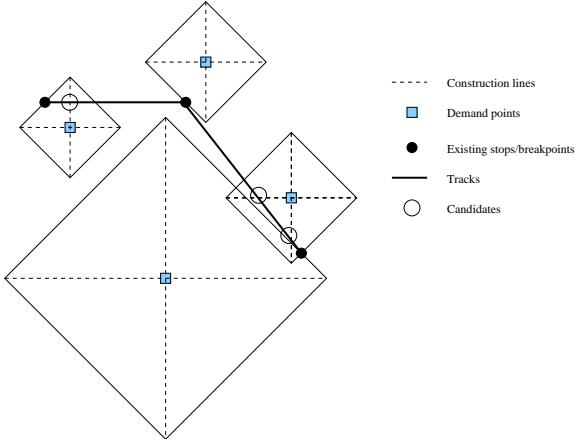
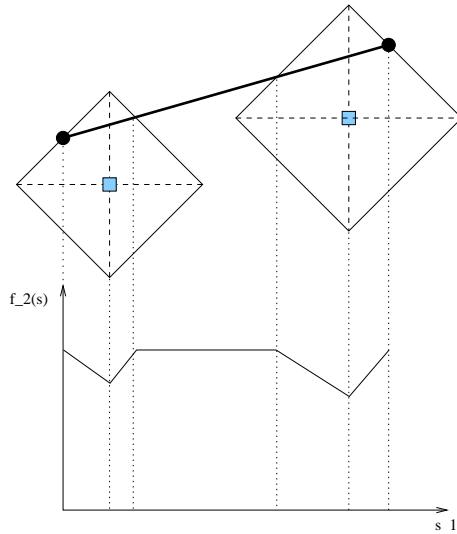


Fig. 6. Set of candidates along the tracks for 1-SLP2

Fig. 7.  $f_2(s)$  along one track

Along a track  $e \in E$ ,  $f_2(s)$  is defined on the interval  $[v_1, v_2]$ ,  $v_1, v_2 \in V$ . But we know that  $f_2(s) \leq f_2(v)$ ,  $\forall v \in V, s \in \mathbb{R}^2$ , therefore we do not need to consider  $v \in V$  as the candidates.

It will be shown in the following lemma that  $f_2(s)$  will not achieve the minimum at a point  $\tilde{s} \in \partial(D_m)$ ,  $m \in \mathcal{M}$ , even though it may be a breakpoint of  $f_2(s)$ .

Thus, we only need to consider the breakpoints  $s \in S_{cand2} = \mathcal{S} \cap \mathcal{K}_2$ . ■

**Lemma 3.9:**  $f_2(s)$  can not achieve the minimum at any point  $s \in \partial(D_m)$ ,  $m \in \mathcal{M}$ .

*Proof:* Denote the gradient left and gradient right of  $f_2(s)$  by  $f_2^-(s)$  and  $f_2^+(s)$  respectively. Take  $\tilde{s} \in \mathcal{S} \cap \partial(D_{\tilde{m}})$ , for some  $\tilde{m} \in \mathcal{M}$ . If  $\mathcal{S}$  is “entering”  $D_{\tilde{m}}$ ,  $f_2^-(\tilde{s})$  does not depend on  $p_{\tilde{m}}$ . Since  $l_1(p_{\tilde{m}}, s') < d_{\tilde{m}}$ , for  $s' \in \mathcal{S} \cap D_{\tilde{m}}$ , whereas the other demand points  $p_m, m \in \mathcal{M} \setminus \{\tilde{m}\}$  does not change the gradient as  $\mathcal{S}$  entering  $D_{\tilde{m}}$ , the gradient is decreasing, i.e.,  $f_2^-(\tilde{s}) > f_2^+(\tilde{s})$ . In a similar way, we can show that gradient of  $f_2(s)$  is also decreasing when  $\mathcal{S}$  is “leaving”  $D_{\tilde{m}}$ . Thus,  $f_2(s)$  does not attain its minimum at the boundary of  $D_m$ ,  $\forall m \in \mathcal{M}$ . ■

Figure 7 illustrates how the gradient changes along a track  $e \in \mathcal{S}$ .

If a construction line  $k_{\tilde{m}i}$  lies upon a track  $\tilde{e} \in \mathcal{S}$ ,  $p_{\tilde{m}}$  must lie on  $\tilde{e}$ . It is easy to see that in this case we only need to consider  $p_{\tilde{m}}$  in our candidate set. It follows that the candidate set is finite.

It can happen that  $S_{cand_2} = \emptyset$ . It means, there does not exist any point on the tracks that improves the objective value  $f_2(v)$ ,  $v \in V$ . Thus, in such a situation it is not reasonable to add any station to our PTN.

The following algorithm summarizes the above discussion to find an optimal stop location for 1-SLP2.

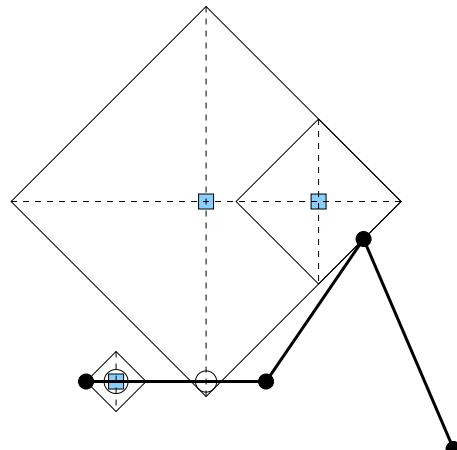


Fig. 8. Candidate set for a 1-SLP2 where a track lies on a construction line

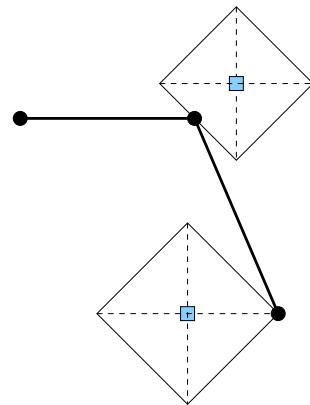


Fig. 9. An example where the existing PTN can not be improved

<b>Alg.2 :</b>	Construction Line Algorithm for 1-SLP2
<b>Input :</b>	$PTN = (V, E)$ , demand points $\mathcal{P} = \{p_1, \dots, p_M\} \subset \mathbb{R}^2$ .
<b>Output :</b>	$s^*$ optimal single stop for 1-SLP2.
<b>Step 1 :</b>	For all $p_m, m \in \mathcal{M}$ , evaluate $v_m, d_m$ , and $D_m$ .
<b>Step 2 :</b>	Compute construction lines $\mathcal{K}_2$ .
<b>Step 3 :</b>	Determine the set of candidates $\mathcal{S}_{cand2} = \{S \cap \mathcal{K}_2\} = \{s_1, \dots, s_L\}$ .
<b>Step 4 :</b>	If $\mathcal{S}_{cand2} = \emptyset$ , STOP. Output: $s^* \in \emptyset$
<b>Step 5 :</b>	Let $s^* = \operatorname{argmin} \{f_2(s_1), \dots, f_2(s_L)\}$
<b>Step 6 :</b>	Output: $s^*$

Computation of  $v_m, d_m$ , and  $D_m$  needs  $\mathcal{O}(MS)$  time. The candidates  $s_1, \dots, s_L$  can be determined in  $\mathcal{O}(MR)$  and the effort to compute objective value for each candidates is  $\mathcal{O}(M)$ . Since  $L \leq 2MR$  we need to make at most  $2MR - 1$  comparison to find the minimum. Thus the overall complexity of Alg.2 is  $\mathcal{O}(M^2R)$ .

Notice that  $\mathcal{S}_{cand2} \subseteq \mathcal{S}_{cand1}$ . Moreover, if  $\mathcal{S} \subseteq D_m, \forall m \in \mathcal{M}$ , then 1-SLP1 and 1-SLP2 are equivalent. The consequence of this condition is that all of the existing stops have to be in the same distance to all of the demand points. This happens, e.g., if all demand points have the same coordinate, or if all the existing stops are at infinity. The second situation has a nice practical interpretation: if all the existing stops are far away from the set of demand points, we do not need to consider them in our planning to open some additional stop. All the demand points will be served only by the new stops, and the problem is reduced to the problem of completely redesigning all stops.

#### IV. N-STOPS LOCATION PROBLEM

In this section we study the general stop location problem where we want to establish at most  $N$  stops. Following the notation from the previous section, we call the problem of completely redesigning the stops and opening additional stops as  $N$ -SLP1 and  $N$ -SLP2 respectively. The notation  $N$ -SLP is used if we refer to both of the problem. In this case, we drop all the index  $i = 1, 2$  for the whole corresponding notation (e.g., construction lines, candidate set) and the readers are asked to assign the notations to the corresponding problem.

The results we get for 1-SLP can be extended for  $N$ -SLP. We know that the optimal solution for 1-SLP is contained in the candidate set  $\mathcal{S}_{cand}$ . The next theorem shows that this holds also for  $N$ -SLP.

**Theorem 4.1:** For  $N$ -SLP, there exists an optimal solution  $S^* \subseteq \mathcal{S}_{cand}$ ,  $|S^*| \leq N$

**Proof:** Suppose we have already a set of solution  $\tilde{S}$  for the set of demand points  $\mathcal{P}$  with  $|\tilde{S}| \leq N$ ,  $\tilde{S} \not\subseteq \mathcal{S}_{cand}$ . Take  $\tilde{s} \in \tilde{S} \setminus \mathcal{S}_{cand}$  and consider only demand points that are covered by  $\tilde{s}$ . Solve this problem as 1-SLP, and due to Theorem 3.3 and 3.8 we obtain an optimal solution  $\tilde{s}' \in \mathcal{S}_{cand}$ . By reassigning the covering of the demand points, we will get at least the same objective value. Iterate this process until all stops in  $\tilde{S} \setminus \mathcal{S}_{cand}$  are replaced by stops in  $\mathcal{S}_{cand}$ . ■

However,  $N$ -SLP is  $\mathcal{NP}$ -hard even if we only consider the finite dominating set  $\mathcal{S}_{cand}$ .

**Theorem 4.2:**  $N$ -SLP is  $\mathcal{NP}$ -hard

**Proof:** The proof is a reduction of the rectangular  $p$ -median problem to the decision version of  $N$ -SLP.

**(Rectangular  $p$ -median problem)** Given a set  $\mathcal{P} = \{p_1, \dots, p_M\} \subset \mathbb{R}^2$  of points in the plane and positive rational number  $k$ . Is there a set  $S \subset \mathbb{R}^2$  of  $p$  points such that if  $l_1(p_m, S)$  is the length of shortest  $l_1$  travel path from  $p_m$  to the closest point in  $S$ , then  $\sum_{m=1}^M l_1(p_m, S) \leq k$ ?

Rectangular  $p$ -median problem is one of some common geometric location problems. Megiddo and Supowit [17] showed the  $\mathcal{NP}$ -hardness of this problem using the reduction from 3-satisfiability.

Given an instance of rectangular  $p$ -median we define the following instance of  $N$ -SLP :

- Leave  $\mathcal{P}$  as it is.
- $N := p$ .
- Define  $\mathcal{S}$  such that it contains sufficiently large parts of the construction lines, where the set of  $V$  consists of the end points of segments in  $\mathcal{S}$  are at infinity. PTN can now be defined by a node set  $V$  and by edges corresponding to the linear pieces given between the nodes.

Defining the instance in this way,  $N$ -SLP1 is equivalent to  $N$ -SLP2 and the  $\mathcal{NP}$ -hardness proof is valid for both problems.

To establish the correctness of this (polynomial) transformation we have to prove the following :

**Claim:**  $N$ -SLP has a feasible solution  $S$  with  $|S| \leq N$  and  $\sum_{m=1}^M l_1(p_m, S) \leq k$  if and only if exists  $S \subset \mathbb{R}^2$ ,  $|S| = p$  such that  $\sum_{m=1}^M l_1(p_m, S) \leq k$ .

- ( $\Rightarrow$ ) Let  $S$  be a feasible solution of  $N$ -SLP. Notice that  $S$  is also feasible for rectangular  $p$ -median problem. If  $|S| = N$ , the result follows directly. If  $|S| < N$ , add  $N - |S|$  arbitrary points. The total distance will be at least the same, and the result follows.
- ( $\Leftarrow$ ) Now let  $S$  be a solution of rectangular  $p$ -median problem. We construct a set of stops  $S \subset \mathcal{S}$  as follows. Take  $s \in S$  and define  $\text{cover}_s$  as in Definition 2.1. Consider only the demand points in  $\text{cover}_s$  and find the optimal solution  $s'$  for unrestricted 1-facility problem. In the case where optimal solution is not unique, we can take one arbitrarily. The point  $s'$  lies on the construction lines (see e.g. [4] or [9] for a proof) and therefore it is feasible for our  $N$ -SLP. Taking  $s'$  as a stop, the total distance for  $N$ -SLP is at least the same as the total distance for rectangular  $p$ -median problem and this completes our proof. ■

## V. HEURISTIC ALGORITHM

Due to Theorem 4.1, there exists a set of optimal stops  $S^*$ ,  $|S^*| \leq N$  such that  $S^* \subseteq \mathcal{S}_{\text{cand}}$ , i.e. we only need to consider the finite candidate points in  $\mathcal{S}_{\text{cand}}$  as defined in Definition 3.2 or 3.7. From Theorem 3.3 and 3.8 we also know that the optimal solution for 1-SLP is always in  $\mathcal{S}_{\text{cand}}$ , thus we can use the result from 1-SLP to build a heuristic algorithm for  $N$ -SLP.

Our idea of heuristic is based on node partitioning scheme that was previously introduced by Maranzana [16].

We divide the demand points into  $N$  disjoint clusters (partition of demand points), and then consider each cluster as 1-SLP. In other words, we will have  $N$  independent 1-SLPs (we call them as  $1\text{-SLP}_1, \dots, 1\text{-SLP}_N$ ) and by solving them we have an initial solution  $\tilde{S} = \{\tilde{s}_1, \dots, \tilde{s}_N\} \subseteq \mathcal{S}_{\text{cand}}$ , where  $\tilde{s}_i$  is the optimal solution of the 1-SLP $_i$ .

In the case where  $1\text{-SLP}_i$  does not have a unique optimal solution for some  $i \in \{1, \dots, N\}$ , we take a solution with maximum cover considering the whole demand points (not only demand points in cluster  $i$ ). In more detail, we do it as follows. Let  $\tilde{S}_i = \{\tilde{s}_{i_1}, \dots, \tilde{s}_{i_J}\}$  be a set of optimal solutions for 1-SLP $_i$ . We define

$$\text{cover}_{\tilde{s}_{i_j}, \tilde{S}_i} = \{p_m \in \mathcal{P} : \tilde{s}_{i_j} = \operatorname{argmin}_{s \in \tilde{S}_i} l_1(p_m, s)\}$$

of each optimal solution  $\tilde{s}_{i_j} \in \tilde{S}_i$ . It means, we define the closest stop  $\tilde{s}_{i_j} \in \tilde{S}_i$  to each demand point  $p_m \in \mathcal{P}$ . If

the closest stop is not unique, we assume that customers can go to any of the closest stations. Having the set of demand points that are covered by each optimal solution in  $\tilde{S}_i$ , we take the optimal solution  $\tilde{s}_{i_{j^*}}$  whose  $|\text{cover}_{\tilde{s}_{i_{j^*}}}|$  is maximal (for simplification, we drop the index  $\tilde{S}_i$  in denotation of cover). If this is still not unique, we take one arbitrarily.

Notice that several 1-SLPs may have the same optimal solution. In this case we have to interpret the answer to  $N$ -SLP in such a way, that we have found only a single location on which several stops will be established. However, if we establish a single stop on this location, it will not effect our objective value. Therefore we can reduce the number of stations to be established to  $N' < N$ .

After we get the initial solution  $\tilde{S}$ , we define  $\text{cover}_{\tilde{s}}$  for each stop  $\tilde{s} \in \tilde{S}$ . There is a slight difference for  $N$ -SLP1 and  $N$ -SLP2 in this step.

For  $N$ -SLP1, we define  $\text{cover}_{\tilde{s}}$  with respect to  $\tilde{S}$ ,

$$\text{cover}_{\tilde{s}, \tilde{S}} = \{p_m \in \mathcal{P} : \tilde{s} = \operatorname{argmin}_{s \in \tilde{S}} l_1(p_m, s)\}$$

All demand points that are served by the same stop will be put in one cluster and thus we have a second  $N$  clusters of demand points.

For  $N$ -SLP2, we define  $\text{cover}_{\tilde{s}}$  with respect to  $V \cup \tilde{S}$ ,

$$\text{cover}_{\tilde{s}, V \cup \tilde{S}} = \{p_m \in \mathcal{P} : \tilde{s} = \operatorname{argmin}_{s \in (V \cup \tilde{S})} l_1(p_m, s)\}$$

All demand points that are served by the same new stop will be put in one cluster and thus we have a second  $N$  clusters of demand points.

By reassigning the covering, some stops in the initial solution may become redundant. In this case, we have  $N''$  clusters of demand points where  $N'' < N$ . Reducing  $N$  to  $N''$  does not effect the objective value.

We repeat the steps of considering each cluster as 1-SLP and define the demand points covered by each stop until we do not get better by doing so.

Now we come to the question how to get the initial cluster of demand points. We propose two strategies to partition our demand points, both are quite intuitive. The first strategy is using Kruskal algorithm and the second one is using  $p$ -median on a tree network.

### A. Clustering Using Kruskal Algorithm

The first idea to divide the demand points into  $N$  clusters is to use the well known Kruskal algorithm. As we know, the Kruskal algorithm finds the minimum spanning tree in a given graph  $G = (V, E)$  with non negative length  $l(e)$  for each edge  $e \in E$  (see any book on basic network optimization, e.g. [10]).

In our case, we define the graph  $G = (V, E)$  as follows.

*Definition 5.1:*

$$V := \text{set of demand points } \mathcal{P}$$

$$E := \{e = (p_i, p_j) : \text{for every pair } p_i, p_j \in V\}$$

$$l(e) := l_1(p_i, p_j) \text{ for } e = (p_i, p_j).$$

We do not want to find a minimum spanning tree in this graph, but we want to find a minimum spanning forest consists of  $N$  trees.

The algorithm starts with a forest which consists of  $|\mathcal{P}|$  trees. Each tree contains only one node and nothing else. In every step of the algorithm, we choose the edge with the least weight that connects two different trees of this forest. If the chosen edge connects nodes which belong to the same tree, the edge is rejected, and not examined again because it could produce a circle which will destroy our tree. We continue with the next edge in the order of least weight which connects two small trees into a bigger one. It means, the number of trees decreases by one in each iteration. Since we want to have a spanning forest consists of  $N$  trees, we terminate the Kruskal algorithm after  $|\mathcal{P}| - N$  iteration.

Unsurprisingly, we consider demand points that belong to the same tree as one cluster and therefore we have an initial  $N$  disjoint clusters of our demand points.

From the discussion above, we get the following algorithm to divide demand points into  $N$  clusters.

<b>Alg.3 :</b>	Division of Demand Points into $N$ Clusters (Kruskal)
<b>Input :</b>	Set of demand points $\mathcal{P} = \{p_1, \dots, p_M\} \subset \mathbb{R}^2$ .
<b>Output :</b>	cluster <sub>1</sub> , ..., cluster <sub><math>N</math></sub>
<b>Step 1 :</b>	Construct a complete graph with the demand points as vertices (as in Definition 5.1).
<b>Step 2 :</b>	Find the minimum spanning forest consisting of $N$ trees using the Kruskal algorithm.
<b>Step 3 :</b>	The set of demand points (vertices) that belong to one tree is considered as one cluster.

Construction of a graph as in Definition 5.1 takes  $\mathcal{O}(M^2)$  steps, and the complexity of Kruskal is  $\mathcal{O}(|E| \log |E|)$ . Our graph has  $\frac{1}{2}(M(M - 1))$  edges since it is a complete graph with  $M$  vertices, therefore the complexity of Kruskal in this case is  $\mathcal{O}(M^2 \log M)$ . Altogether, the complexity of Alg.3 is  $\mathcal{O}(M^2 \log M)$ .

This idea of clustering is rather simple and intuitive,

but naturally it also has its weaknesses. One which is quite obvious, is that this method can lead to the situation where we have one cluster with many demand points and another cluster with only one demand point. This happens e.g. in a situation where one demand point is further away from the others (see Figure 10(a)) or in a situation where each pair of demand points has the same distance (see Figure 10(b)).

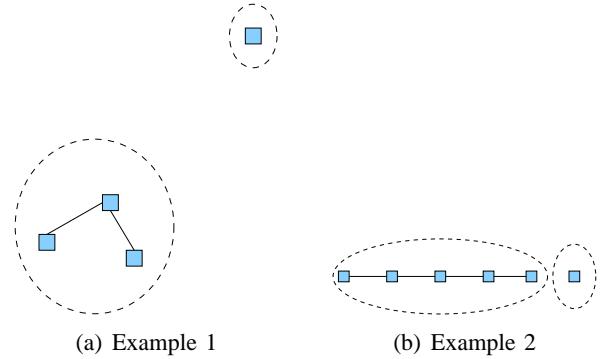


Fig. 10. Clustering using Kruskal

In the first situation, this kind of clustering is quite reasonable since we will establish one stop which covers one far away demand point and another stop which covers some demand points that are close to each other, as one will tend to do in reality. But in the second situation, clustering using Kruskal will produce an initial solution that is far from optimal, as we can easily see in Figure 10(b).

Based on this fact, we came to the idea to divide the demand points into  $N$  clusters by finding the  $N$ -median of our graph defined in Definition 5.1. Demand points that are served by the same facility (median) are then put in the same cluster, and therefore we get our initial  $N$  clusters and can continue with the procedure as explained in Section V.

A detailed discussion about this clustering strategy will be presented in the next section.

### B. Clustering Using $p$ -median on Tree Network

Determining the locations of  $p$  facilities on a network or space so as to minimize the total distance between users and facilities is commonly called the  $p$ -median problem. Hakimi [6] was the first to formulate the problem for locating a single and multi median on a network, and he also has shown that if the users are located only at vertices then an optimal solution to the  $p$ -median problems exists on the vertices (see [7]).

The problem is well known to be  $\mathcal{NP}$ -hard, even if we restrict the median (facilities) to be a subset of vertices (see [5]). Hakimi [8] has also shown that the

problem remains  $\mathcal{NP}$ -hard when the network has a simple structure, e.g. planar graph of maximum vertex degree three.

However, the  $p$ -median problem is solvable in polynomial time for some special structure of network. In particular, this is true for lines [12], [19] and trees [8], [13], [26].

We will use the algorithm in [26] to find the initial  $N$  clusters of demand points that we require to begin our heuristic algorithm. In this paper, Tamir proposed an algorithm to solve  $p$ -median problem on a tree by the "leaves to root" dynamic programming algorithm with complexity  $\mathcal{O}(pn^2)$  where  $n$  is the number of node in the tree.

As in the previous section, we consider demand points as the nodes of a complete graph as in Definition 5.1 and compute the minimum spanning tree of this network. Since our graph is a dense graph, we use Prim's algorithm to find the minimum spanning tree. We then use the algorithm presented in [26] to find set  $X \subseteq \mathcal{P}$ , the set of  $N$ -median on this tree. The next step is to define  $cover_x$  for each  $x \in X$  as in Definition 2.1 and then put all demand points that are covered by the same facility (median)  $x \in X$  into the same cluster.

We conclude this procedure to get initial  $N$  clusters of demand points using  $N$ -median on a tree network in the following algorithm:

**Alg.4 :** Division of Demand Points Into  $N$  Clusters ( $N$ -median on a tree)

<b>Input :</b>	Set of demand points $\mathcal{P} = \{p_1, \dots, p_M\} \subset \mathbb{R}^2$ .
<b>Output :</b>	cluster <sub>1</sub> , ..., cluster <sub><math>N</math></sub>
<b>Step 1 :</b>	Construct a complete graph with the demand points as vertices (as in Definition 5.1).
<b>Step 2 :</b>	Find the minimum spanning tree using Prim's algorithm.
<b>Step 3 :</b>	Find set $X \subseteq \mathcal{P}$ , the $N$ -median on this tree using algorithm presented in [26].
<b>Step 4 :</b>	Define $cover_x$ for each $x \in X$ .
<b>Step 5 :</b>	The set of demand points that are covered by the same facility is considered as one cluster.

From the previous subsection we know that the complexity of our first step is  $\mathcal{O}(M^2)$ . Prim's algorithm computes a minimum spanning tree also in  $\mathcal{O}(M^2)$  steps. As already mentioned above, the algorithm in [26] finds an  $N$ -median on a tree with  $M$  nodes in time  $\mathcal{O}(NM^2)$ . Thus the total complexity of Alg.4 is  $\mathcal{O}(NM^2)$ .

Figure 11 illustrates how this clustering strategy works for the same example we have in Figure 10(b).

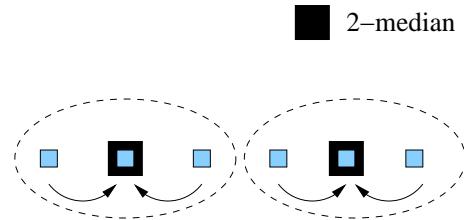


Fig. 11. Clustering using  $N$ -median on a tree

### C. The Algorithm

Summarizing all the above discussions, we get the following heuristic algorithm to solve  $N$ -SLP.

**Alg.5 :**  $N$ -SLP1

<b>Input :</b>	$PTN = (V, E)$ , demand points $\mathcal{P} = \{p_1, \dots, p_M\} \subset \mathbb{R}^2$ , number of stations to be established $N$ , stopping criterion $\epsilon$ .
<b>Output:</b>	$S^*$ optimal stops location, $ S^*  \leq N$ .
<b>Step 1:</b>	Divide demand points into $N$ clusters (using Alg.3 or Alg.4). Set $\tilde{S} := \emptyset$
<b>Step 2:</b>	For $i = 1, \dots, N$ calculate $\tilde{s}_i$ , the optimal single stop for cluster <sub><math>i</math></sub> using Alg.1. If the solution is not unique, take the one with maximum cover considering the whole demand points. If $\tilde{s}_i = \tilde{s}_j$ for some $j \in \{1, \dots, i-1\}$ , set $N := N - 1$ , else $\tilde{S} := \tilde{S} \cup \{\tilde{s}_i\}$ .
<b>Step 3:</b>	Calculate $f_{1_1}(\tilde{S}) = \sum_{i=1}^N f_1(\tilde{s}_i)$ where $f_1(\tilde{s}_i)$ is optimal objective value in cluster <sub><math>i</math></sub> , i.e. $f_1(\tilde{s}_i) = \sum_{p_m \in \text{cluster}_i} l_1(p_m, \tilde{s}_i)$ .
<b>Step 4:</b>	For $i = 1, \dots, N$ Define $cover_{\tilde{s}_i, \tilde{S}}$ for each $\tilde{s}_i \in \tilde{S}$ . If $cover_{\tilde{s}_i} = \emptyset$ , set $N := N - 1$ , $\tilde{S} := \tilde{S} \setminus \{\tilde{s}_i\}$ .
<b>Step 5:</b>	Calculate $f_{1_2}(\tilde{S}) = \sum_{p_m \in \mathcal{P}} l_1(p_m, \tilde{S})$ . If $f_{1_1}(\tilde{S}) - f_{1_2}(\tilde{S}) < \epsilon$ , set $S^* = \tilde{S}$ , otherwise consider $cover_{\tilde{s}_i}$ as cluster <sub><math>i</math></sub> for $i = 1, \dots, N$ and go to Step 2.
<b>Step 6:</b>	Output: $S^*$

The complexity of Step 1 in Alg.5 has been discussed in this section. Alg.1 finds an optimal solution for 1-SLP in  $\mathcal{O}(M^2R)$  time, where  $R$  is the number of tracks/rails

in the PTN. Therefore, finding optimal single stops for all clusters takes  $\mathcal{O}(NM^2R)$  steps. In the case where  $\tilde{s}_i$  is not a unique optimal solution for some  $i \in \{1, \dots, N\}$ , we have to determine the covering of each solution taking into account the whole demand points. This takes at most  $\mathcal{O}(MR)$  steps. Altogether the complexity of Step 2 is  $\mathcal{O}(NM^2R)$ . Defining  $\text{cover}_{\tilde{s}_i, \tilde{S}}$  for each  $\tilde{s}_i \in \tilde{S}$  requires  $\mathcal{O}(NM)$  steps, and this makes the complexity for each iteration of Alg.5  $\mathcal{O}(NM^2R)$ .

Several modifications of Alg.5 are needed for  $N$ -SLP2. For the convenience of the reader, we write these modifications separately in the following algorithm.

**Alg.6 :**  $N$ -SLP2

**Input :**

**Output:**

} As in Alg.5

**Step 1:**

**Step 2:** For  $i = 1, \dots, N$

calculate  $\tilde{s}_i$ , the optimal single stop for cluster $_i$  using Alg.2. If the solution is not unique, take the one with maximum cover considering the whole demand points.

If  $\tilde{s}_i \in \emptyset$  or  $\tilde{s}_i = \tilde{s}_j$  for some  $j \in \{1, \dots, i-1\}$ , set  $N := N - 1$ , else  $\tilde{S} := \tilde{S} \cup \{\tilde{s}_i\}$ .

**Step 3:** Calculate  $f_{2_1}(\tilde{S}) = \sum_{i=1}^N f_2(\tilde{s}_i)$  where  $f_2(\tilde{s}_i)$  is optimal objective value in cluster $_i$ , i.e.  $f_2(\tilde{s}_i) = \sum_{p_m \in \text{cluster}_i} l_1(p_m, V \cup \{\tilde{s}_i\})$ .

**Step 4:** For  $i = 1, \dots, N$

Define  $\text{cover}_{\tilde{s}_i, V \cup \tilde{S}}$  for each  $\tilde{s}_i \in \tilde{S}$ .

If  $\text{cover}_{\tilde{s}_i} = \emptyset$ , set  $N := N - 1$ ,  $\tilde{S} := \tilde{S} \setminus \{\tilde{s}_i\}$ .

**Step 5:** Calculate  $f_{2_2}(\tilde{S}) = \sum_{p_m \in \mathcal{P}} l_1(p_m, V \cup \tilde{S})$ . If  $f_{2_1}(\tilde{S}) - f_{2_2}(\tilde{S}) < \epsilon$ , set  $S^* = \tilde{S}$ , otherwise consider  $\text{cover}_{\tilde{s}_i}$  as cluster $_i$  for  $i = 1, \dots, N$  and go to Step 2.

**Step 6:** Output:  $S^*$

Until Step 3, the complexity of Alg.6 is the same as Alg.5. Defining  $\text{cover}_{\tilde{s}_i, V \cup \tilde{S}}$  for each  $\tilde{s}_i \in \tilde{S}$  requires  $\mathcal{O}(N(S + M))$  steps where  $S$  is the number of existing stops. Thus, the complexity for each iteration of Alg.6 is  $\mathcal{O}(N(S + M^2R))$ .

Notice that Alg.5 and Alg.6 is basically a heuristic algorithm for a general restricted median problem. Since the problem is  $\mathcal{NP}$ -hard, moreover it is  $\mathcal{NP}$ -hard in strong sense [18], it is unlikely that a (pseudo) polynomial-time algorithm for its computation can ever

be found [5]. Likewise, the total complexity of our heuristic algorithm is not known since we do not know beforehand the number of iteration needed until the program terminates.

## VI. COMPUTATIONAL EXPERIMENTS

The heuristic algorithms were implemented in C++ and the visualization of the output was implemented using OpenGL<sup>2</sup>. A simple input generator was also implemented in the program. All computations were done at the University of Kaiserslautern on a PC with an AMD Athlon 2200+ processor running under the Linux operating system.

The algorithm has been applied to several randomly generated problems, having the number of demand points  $M$  equal to 50, 100, 200, 300, 350, and 400. For problems up to 200 demand points, the number of existing stops  $S$  was set to 7, and the number of stations to be established  $N$  varies from 4 to 7. For larger number of demand points, we set  $S$  to 10 and  $N$  to 5, 7, 10, and 15. For every type of setting we set the stopping criterion  $\epsilon$  to 0.0001.

We try to compare the performance between our two clustering strategies, both with respect to heuristic performance and time performance. In order to do so, for each  $M$ ,  $S$ , and  $N$  several instances were generated, by randomly choosing the coordinates of demand points  $p_m \in \mathcal{P}$  and existing stops  $v_i \in V$  uniformly within the interval  $[0, 10]^2$ . In order to construct the PTN, for each pair of existing stops  $v_i$  and  $v_j$  we generated an additional parameter  $\alpha$  uniformly within the interval  $[0, 1]$ . If  $\alpha > 0.6$ , we construct a direct connection between  $v_i$  and  $v_j$ .

The computational times are always given in seconds of CPU, which excludes input and output. To get more accurate computational time the same instance was run several times and we took the average computational times.

In the case where some clusters have an identical optimal single stop, or where some optimal stops are redundant by reassigning the covering,  $N$  will be reduced and the program gives an optimal solution  $S^*, |S^*| = N' < N$ . The program gives also a notification that we can reduce the number of stations to be established without effecting the objective value.

During the computational experiment, clustering using Kruskal often gives an indication that we could reduce the number of stations to be established to obtain the same objective value. From a practical point of view, this is really good since it means that we can save

<sup>2</sup>Open Graphics Language.

the budget to establish one station. However, it often does not give us the right interpretation of what actually happens. Figure 12 illustrates a simple explanation why Kruskal often gave a warning that we could reduce  $N$  “without” effecting the objective value.

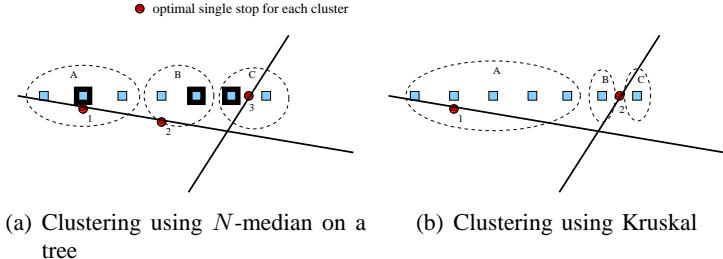


Fig. 12. A simple example with  $N = 3$ . Using  $N$ -median strategy, we get clusters A, B, C as in Figure (a). The single optimal stop for each cluster is stop 1, 2, and 3 respectively and the program continue with the next iteration. Using Kruskal, we get clusters A, B, C as in Figure (b). The single optimal stop for cluster A is stop 1, whereas stop 2 is the identical optimal single stop for both cluster B and C. The program gives an optimal solution  $S^*$ , with  $|S^*| = 2$  and notifies  $N$  can be reduced without effecting the objective value. However, reducing  $N$  to two does worsen the objective value.

In Section V-A we already mention that one drawback of clustering using Kruskal algorithm is the unbalanced number of demand points in the clusters. It means, one cluster might have almost all demand points whereas the other clusters may contain only one or two demand points in it. It is then likely to happen that two clusters (with only few demand points) will have an identical optimal solution, and therefore  $N$  is reduced.

On the contrary, clustering using  $N$ -median in a tree network in general obtains a fair distribution of demand points into  $N$  clusters. Therefore, a warning from the program that we can reduce the number of stations to be established to get the same objective value is more reliable rather than those we get from Alg.2.

During the computational experiment, clustering using Kruskal obtained better objective values in 28 % of the instances, clustering using  $N$ -median performed better in 68 % of the instances, and in the rest 4 % of the instances both strategies gave exactly the same solution.

Table I and II summarises the performance of these two clustering strategies during the computational experiments. For each algorithm we give the objective value obtained by the heuristics (Obj. Val) and CPU time in seconds (Time). The given values are averages over the instances in each setting ( $M, S, N$ ). The first group of columns (Kruskal) refers to the case where we use Kruskal clustering strategy, and the second ( $N$ -median) refers to the case where we use  $N$ -median on the spanning tree.

As one can expect, the objective values for  $N$ -SLP2

TABLE I  
COMPARISON OF THE CLUSTERING STRATEGIES FOR  $N$ -SLP1

$M$	$S$	$N$	Kruskal		$N$ -median	
			Obj. Val	Time	Obj. Val	Time
50	7	4	134,038	0,026	129,315	0,055
50	7	5	123,702	0,036	117,139	0,068
50	7	6	109,270	0,022	106,042	0,076
50	7	7	96,657	0,033	94,383	0,070
100	7	4	278,093	0,187	277,704	0,287
100	7	5	247,756	0,169	251,346	0,318
100	7	6	234,360	0,174	225,086	0,431
100	7	7	235,821	0,17	227,825	0,465
200	7	4	540,927	2,029	518,393	1,908
200	7	5	492,034	2,012	493,121	2,329
200	7	6	580,587	2,129	535,046	2,934
200	7	7	495,894	2,075	488,625	2,925
300	10	5	690,677	20,026	702,603	7,317
300	10	7	637,076	20,488	592,137	10,096
300	10	10	566,234	19,741	514,554	15,016
300	10	15	381,768	20,752	359,087	23,415
350	10	5	780,784	45,068	768,270	11,400
350	10	7	734,666	45,793	706,169	14,655
350	10	10	788,644	45,936	773,222	20,358
350	10	15	469,132	44,797	475,102	36,563
400	10	5	1019,484	87,766	1000,457	16,385
400	10	7	807,379	82,665	795,490	20,650
400	10	10	760,961	85,923	758,819	28,554
400	10	15	608,768	84,793	614,910	43,154

are almost always better compared to  $N$ -SLP1.

The comparison of time performance between these two clustering strategies for  $N$ -SLP1 are shown in Figure 13 and 14. For  $N$ -SLP2, the time performance of the clustering strategies does not differ significantly as for  $N$ -SLP1.

Figure 13 shows how the two algorithms perform if we keep  $N$  constant. When  $M$  is small, Alg.3 performs slightly better than Alg.4, but as the number of demand points grows larger, Alg.4 performs much faster than Alg.3.

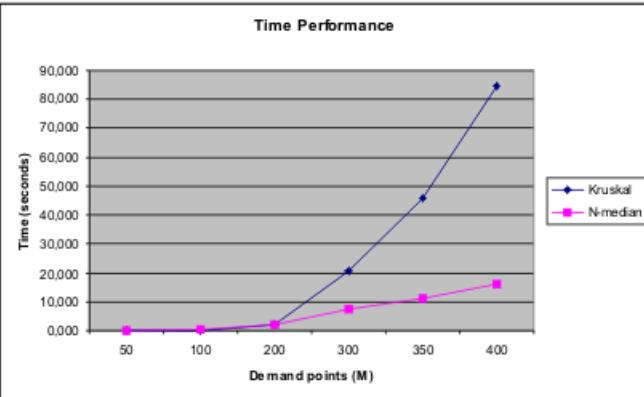
If we keep the number of demand points  $M$  constant, we see in Figure 14 that computational time of Alg.3 is relatively constant over  $N$  while the computational time of Alg.4 increases as  $N$  grows. However, the increase is not as fast as the increase of time performance of Alg.3 as  $M$  grows. Thus, for instances with large number

TABLE II

COMPARISON OF THE CLUSTERING STRATEGIES FOR  $N$ -SLP2

$M$	$S$	$N$	Kruskal		$N$ -median	
			Obj. Val	Time	Obj. Val	Time
50	7	4	104,187	0,025	98,605	0,070
50	7	5	102,546	0,027	98,240	0,067
50	7	6	87,953	0,030	86,215	0,095
50	7	7	85,298	0,020	82,808	0,090
100	7	4	243,761	0,150	235,196	0,315
100	7	5	202,997	0,165	202,534	0,360
100	7	6	191,470	0,160	188,649	0,495
100	7	7	203,195	0,155	199,4095	0,55
200	7	4	381,623	1,980	366,778	1,855
200	7	5	429,214	1,930	412,993	2,300
200	7	6	512,080	1,850	477,893	2,800
200	7	7	468,211	2,015	450,743	3,525
300	10	5	497,742	20,105	477,942	6,945
300	10	7	528,241	19,745	450,178	9,590
300	10	10	432,336	19,350	430,871	14,510
300	10	15	317,476	20,180	303,149	22,330
350	10	5	498,392	43,730	476,433	10,790
350	10	7	662,095	45,385	547,452	13,940
350	10	10	748,462	45,610	718,611	19,750
350	10	15	460,610	44,580	419,547	35,770
400	10	5	855,938	85,565	778,173	15,100
400	10	7	715,718	82,465	641,195	20,275
400	10	10	654,059	86,570	649,762	29,290
400	10	15	554,018	84,160	528,500	42,770

of demand points, Alg.4 still performs better even for relatively many stations to be established.

Fig. 13. Time performance,  $N = 5$ 

A visualization of the program is given in Figure 15. The smallest squares represent demand points, the squares in middle size are existing stops, and the big squares are the optimal solutions for  $N$ -SLP. The shade of the background represents the Voronoi region of each optimal solution. This gives us information about the covering of each demand point. See e.g. [14] for further discussion about Voronoi diagrams with respect to the rectangular metric.

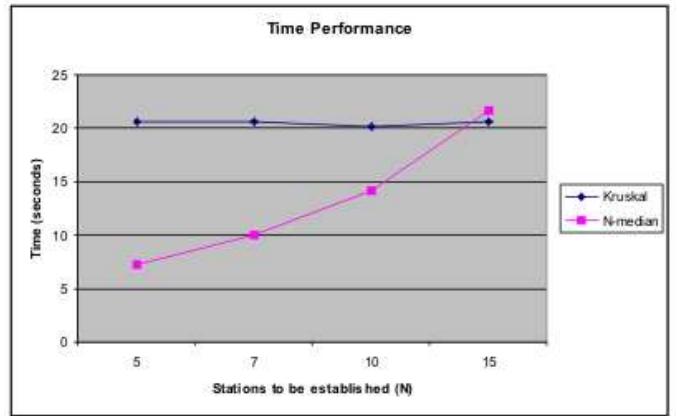
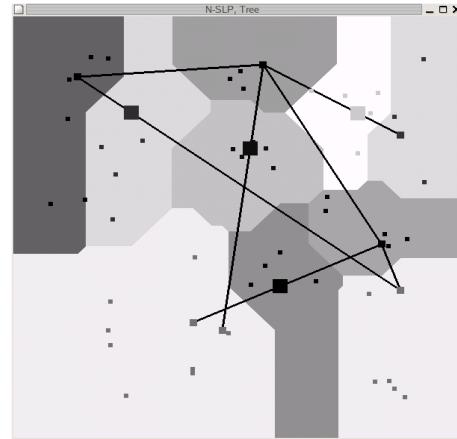
Fig. 14. Time performance,  $M = 300$ 

Fig. 15. Visualization of optimal solution

## VII. POSSIBLE EXTENSIONS AND FURTHER RESEARCH

This research can be extended in many directions. Some of them are (1) to introduce forbidden regions along the tracks in order to avoid the optimal solutions to be too close with the existing stops, (2) to introduce weights  $w_m$  for each demand point which represent e.g. the number of customers travelling from/to this point,

and (3) to consider other clustering strategies. These are left for future research.

#### ACKNOWLEDGEMENTS

The author would like to thank Prof. Dr. Horst W. Hamacher and Prof. Dr. Anita Schöbel for their comments, suggestions, and corrections which helped to improve the presentation of this paper.

#### REFERENCES

- [1] M. S. Daskin, *Network and Discrete Location: Models, Algorithms, and Applications*. New York: Wiley, 1995.
- [2] Z. Drezner, Ed., *Facility Location: A Survey of Applications and Methods*. Berlin: Springer, 1995.
- [3] Z. Drezner and H. W. Hamacher, Eds., *Facility Location: Application and Theory*. Berlin: Springer, 2002.
- [4] R. L. Francis, F. Leon, J. McGinnis, and J. A. White, *Facility Layout and Location: An Analytical Approach*, 2nd ed. New York: Prentice Hall, 1992.
- [5] M. R. Garey and D. S. Johnson, *Computers and Intractability - A Guide to the Theory of NP-Completeness*. San Francisco: Freeman, 1979.
- [6] S. L. Hakimi, "Optimum location of switching centers and absolute centers and medians of a graph," *Operation Research*, vol. 12, pp. 450–459, 1964.
- [7] ———, "Optimum distribution of switching centers in a communication network and some related graph theoretic problems," *Operation Research*, vol. 13, pp. 462–475, 1965.
- [8] S. L. Hakimi and O. Kariv, "An algorithmic approach to network location problems. part 2 : The p-medians," *SIAM Journal on Applied Mathematics*, vol. 37, pp. 539–560, 1979.
- [9] H. W. Hamacher, *Mathematische Lösungsverfahren für planare Standortprobleme*. Vieweg Verlag, 1995.
- [10] H. W. Hamacher and K. Klamroth, *Linear and Network Optimization - A Bilingual Textbook*, ser. Mathematics International. Vieweg Verlag, 2001.
- [11] H. W. Hamacher, A. Liebers, A. Schöbel, D. Wagner, and F. Wagner, "Locating new stops in a railway network," *Electronic Notes in Theoretical Computer Science*, vol. 50, no. 1, pp. 1–11, 2001.
- [12] R. Hassin and A. Tamir, "Improved complexity bounds for location problems on the real line," *Operation Research Letters*, vol. 10, pp. 395–402, 1991.
- [13] W. L. Hsu, "The distance-domination numbers of tree," *Operation Research Letters*, vol. 1, pp. 96–100, 1982.
- [14] D. T. Lee and C. K. Wong, "Voronoi diagrams in  $l_1$  ( $l_\infty$ ) metrics with 2-dimensional storage applications," *SIAM Journal on Computing*, vol. 9, pp. 200–211, 1980.
- [15] R. F. Love, J. G. Morris, and G. O. Wesolowsky, *Facility Location: Models and Methods*. New York: Elsevier Science Publishing Co., Inc., 1988.
- [16] F. E. Maranzana, "On the location of supply points to minimize transport costs," *Operational Research Quarterly*, vol. 15, pp. 261–270, 1964.
- [17] N. Megiddo and K. J. Supowit, "On the complexity of some common geometric location problems," *SIAM Journal on Computing*, vol. 13, pp. 182–196, 1984.
- [18] P. B. Mirchandani and R. L. Francis, *Discrete Location Theory*. New York: Wiley, 1990.
- [19] P. B. Mirchandani, R. Kohli, and A. Tamir, "Capacitated location problems on a line," *Transportation Science*, vol. 30, no. 1, pp. 75–80, 1996.
- [20] K. C. Mosler, *Continuous Location of Transportation Network*. Berlin: Springer, 1987.
- [21] S. Nickel and H. W. Hamacher, "Restricted planar location problem and applications," *Naval Research Logistics*, vol. 42, pp. 967–992, 1995.
- [22] D. R. Poertranto, "Stop location problem in public transportation networks," Master's thesis, Fachbereich Mathematik, Technische Universität Kaiserslautern, Germany, 2004.
- [23] A. Schöbel, "Optimization models in public transportation," Lecture Notes, 2003.
- [24] ———, "Locating stops along bus or railway lines – a bicriteria problem," *Annals of Operations Research*, 2005, to appear.
- [25] A. Schöbel, H. W. Hamacher, A. Liebers, and D. Wagner, "The continuous stop location problem in public transportation networks," Fachbereich Mathematik, Technische Universität Kaiserslautern, Report in Wirtschaftsmathematik 81/2002, 2002.
- [26] A. Tamir, "An  $\mathcal{O}(pn^2)$  algorithm for the  $p$ -median and related problems on tree graphs," *Operation Research Letters*, vol. 19, pp. 59–64, 1996.
- [27] U. O. Vaubel, *Standorttheoretische Methoden im Stadt- und Regionalverkehr*. Institut für Städtebauwesen, RWTH Aachen, 1981.

# A Column Generation Approach to the Capacitated Vehicle Routing Problem with Stochastic Demands

Christian H. Christiansen\* and Jens Lysgaard

Aarhus School of Business/Department of Accounting, Finance and Logistics

Fuglesangs Allé 4, DK-8210 Aarhus V, Denmark

Email: {chc,lys}@asb.dk

\*Corresponding author

**Abstract**—In this article we introduce a new exact solution approach to the Capacitated Vehicle Routing Problem with Stochastic Demands (CVRPSD). In particular, we consider the case where all customer demands are distributed independently and where each customer's demand follows a Poisson distribution.

The CVRPSD can be formulated as a Set Partitioning Problem. We show that, under the above assumptions on demands, the associated column generation subproblem can be solved using a dynamic programming scheme which is similar to that used in the case of deterministic demands.

To evaluate the potential of our approach we have embedded this column generation scheme in a branch-and-price algorithm. Computational experiments on a large set of test instances show promising results.

**Keywords**—Routing, Stochastic programming, Logistics

## I. INTRODUCTION

THE Capacitated Vehicle Routing Problem (CVRP) in a deterministic environment has been widely studied throughout the literature, and can be described as follows. A set of customers must be provided with known quantities of a common commodity from a single depot. To make the deliveries a fleet of identical vehicles, each with a given capacity, is available. The objective is to find a collection of routes of minimum total travel cost under the restrictions that i) each route begins and ends at the depot, ii) each customer is serviced exactly once, and iii) the total demand on any route does not exceed the vehicle capacity.

The CVRP has been extended in numerous directions for instance by incorporating time windows, multiple depots or maximum route duration. For thorough reviews of the CVRP with various extensions see [1], [21]. In this article we consider the Capacitated Vehicle Routing Problem with Stochastic Demands (CVRPSD), which may be viewed as a stochastic counterpart of the CVRP. The CVRPSD differs from the CVRP with respect to the following points:

- 1) In the CVRPSD, the customers' demands are stochastic variables of which only the probability distribution for each customer is assumed known at the time of planning.
- 2) In the CVRPSD, it is the *expected* total travel cost that must be minimized.
- 3) In the CVRPSD, the total actual demand on a route may exceed the vehicle capacity. In such cases a *failure* is said to occur. A strategy is required for updating the routes in case of such an event. The actual action resulting from this strategy is called a *recourse* action. The particular strategy affects the expected cost of a given route, so the strategy must be known at the time of planning.

The CVRPSD has not received nearly the same level of attention as the CVRP. In the literature there are given several reasons for the limited attention paid to the CVRPSD. Of these perhaps the most important reason is that the CVRP problem in itself is very hard to solve, and adding a stochastic dimension to the problem only makes it even more intractable.

Nonetheless, neglecting the stochastic nature of demands during the planning of the routes can incur substantially higher expected costs, than what would have been the result if the stochastic demands had been explicitly included in the route planning. For the TSP this has been thoroughly illustrated by Laporte and Trudeau in [7]. However, the effects causing the increased expected cost does not only relate to the customers' sequence on each route. To see this, consider the undirected network in Figure 1.

Each letter denotes a customer, whereas 0 denotes the depot. For each customer, the parentheses contain the possible actual demands associated with the customer. For customers *A* and *C*, the probability for each of the two possible actual demands is 0.5. Further, for each customer the value in square brackets is the expected value of the stochastic demand. If a vehicle is depleted

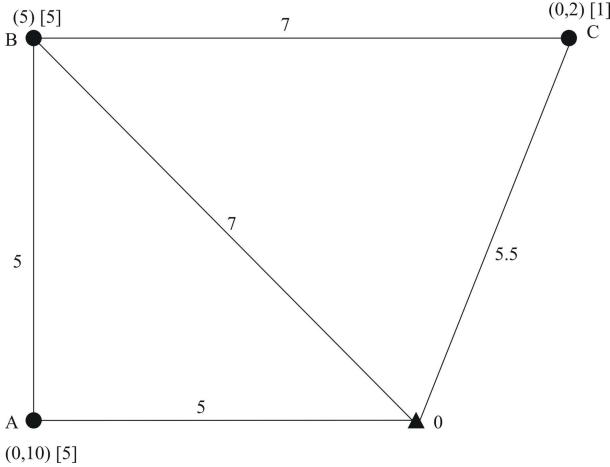


Fig. 1. Routing with stochastic demands

it must return to the depot to reload and then continue its route.

Considering only the expected values of the demands as input to a CVRP model would yield the collection of routes with minimum distance traveled, not including the expected penalty costs, namely the routes  $0 - A - B - 0$  and  $0 - C - 0$ . However, considering the stochastic demands as input to a CVRPSD model would yield the routes  $0 - B - C - 0$  and  $0 - A - 0$ .

The expected travel cost (including expected penalty cost) for the two solutions is 33 and 29.5, respectively. As this example illustrates, neglecting the stochastic nature of demands can cause a suboptimal solution, not only as a consequence of the sequence on each route, but also as a consequence of wrongful allocation of customers to the routes. Several other reasons have been given for incorporating stochastic demands into the route planning, for an overview of these see e.g. [4], [13], [18].

The article is organized as follows. Section II gives a brief literature review regarding research on a priori routing regarding CVRPSD. Section III focuses on the CVRPSD modeling. Section IV presents our new solution method, and Section V provides computational results obtained with the proposed algorithm. Concluding remarks are given in section VI.

## II. LITERATURE REVIEW

To the best of our knowledge the first to address the CVRPSD was Tillman in 1969 [20]. He considered a multi depot variant of the CVRP with Poisson distributed demands. The model considered a cost trade off between exceeding the vehicle capacity and finishing the route with excess capacity. The solution approach was a modification of the savings algorithm originally introduced

by Clarke and Wright in 1964 [6]. For further review of savings based approaches see [3], [8].

Several modeling approaches have been explored regarding the CVRPSD (see, e.g., [11]). Two frequently used approaches are chance constrained programming and two stage stochastic programming, respectively.

Chance constrained models implicitly incorporate the cost of failure. This is done by introducing a threshold value, limiting the maximum probability of failure for each route in the final route collection.

Two stage stochastic programming models, however, incorporate the cost of failure explicitly. The first stage contains the planning of routes, taking into account the expected failure costs incurred by the execution of the routes. The second stage contains the execution of the planned routes according to the chosen strategy for updating the routes in case of failure.

Yet another approach to the CVRPSD has been Markov decision process modeling. However, due to the large number of states this approach has received limited attention [7].

For a comparison of the chance constrained approach and the two stage stochastic programming approach see [2], [7], [19].

Regarding the two stage stochastic programming approach, Bertsimas [4] formulated two widely accepted recourse actions (A) and (B), respectively. These are based on two different assumptions regarding the time at which a customer's actual demand becomes known. Strategy (A) assumes that a customer's actual demand becomes known only upon arrival at the customer. Strategy (B), however, assumes that actual demands become known early enough to enable the vehicle to skip customers with zero actual demand. The recourse action under both strategies is to deplete the vehicle at the point of failure, return to the depot to reload and continue the originally planned route from the point of failure. In the particular case that a customer's demand equals the remaining load of the vehicle, the vehicle returns to the depot to reload before visiting the next customer.

In their article from 1993, Laporte and Louveaux developed an integer L-shaped method for stochastic programs with recourse [15]. Their approach is based on adding feasibility cuts to a relaxed flow formulation of a CVRPSD until an integer feasible solution is found. If the discrepancy between a lower bound on the expected cost of failure and the current value of the failure cost is above some threshold, an optimality cut is added to the formulation. However, in their article no computational results were presented. The method has been applied to the CVRPSD in 1995 [10], 1998 [16] and 2002 [17].

In the latter, instances with up to 100 customers were solved.

From the computational results it can be seen that the L-shaped method seems to perform best on problems with small expected demands relative to the vehicle capacity.

In this article we develop a new solution approach to the CVRPSD. Specifically, we formulate the CVRPSD as a Set Partitioning Problem and develop a dynamic programming algorithm for solving the associated column generating subproblem.

This approach is motivated by the tendency in deterministic vehicle routing that column generation approaches are particularly competitive in the case of tight restrictions (e.g. tight capacity restrictions). Therefore, column generation seems to be a promising approach to solving those instances that have proven most difficult to existing algorithms.

Indeed, we expect our approach to be competitive for solving CVRPSD instances in which it is optimal to have only a few customers per route. It is worth emphasizing that this is exactly the instance type for which the L-shaped approach in [17] seems relatively less effective. By the introduction of our approach, a wider range of problem instances is expected to be solvable, hence strengthening the practical ability of stochastic models.

### III. NOTATION AND MODEL FORMULATION

To formally describe the model let  $G = (V, E)$  be an undirected graph, with vertex set  $V = \{0, \dots, n\}$  and edge set  $E$ . Vertex 0 represents a depot, and each of the vertices in  $V_c = \{1, \dots, n\}$  represents a customer. With each edge  $\{i, j\}$  is associated a travel cost  $d_{ij}$ . Each customer  $i$  has a Poisson distributed demand with an expected value of  $\lambda_i > 0$ . We make the assumption that each  $\lambda_i$  is integer. The customers' demands are assumed to be independent. The vehicle capacity is denoted  $Q$ . For each customer  $i$ , we let  $q_i$  denote the stochastic variable describing the demand at customer  $i$ .

We define a feasible *elementary* route as a path  $(0, v_1, \dots, v_k, 0)$  where  $v_1, \dots, v_k$  are  $k$  different customers whose total expected demand does not exceed  $Q$ . (As in [17], we do not permit routes whose total expected demand exceeds  $Q$ , as such routes would systematically fail.) For any feasible elementary route  $r$ , let  $c_r$  denote its expected cost. Further, let  $\mathfrak{R}_e$  denote the set of all feasible elementary routes.

Let  $\alpha_{ir}$  be a binary parameter describing the number of times route  $r$  visits customer  $i$ , and let  $x_r$  be a binary variable of value 1 if route  $r$  is chosen and 0 otherwise. This leads to the following Set Partitioning formulation:

$$(P_{org})$$

$$\min: \sum_{r \in \mathfrak{R}_e} c_r x_r \quad (1)$$

$$\text{s.t.: } \sum_{r \in \mathfrak{R}_e} \alpha_{ir} x_r = 1 \quad \forall i \in V_c \quad (2)$$

$$x_r \in \{0, 1\} \quad \forall r \in \mathfrak{R}_e \quad (3)$$

The objective (1) minimizes the total expected distribution cost. Constraints (2) ensure that each customer is contained in exactly one route, whereas the constraints (3) are the binary constraints on the decision variables.

The recourse that we have chosen is identical to strategy (A), with one exception. If the vehicle, on a route  $(0, 1, \dots, k, 0)$ , at some customer  $i < k$  is exactly depleted, we assume that it continues, without an intermediate return to the depot, along the route until it at some customer  $j \in \{i + 1, \dots, k\}$  encounters a positive demand (or reaches the depot). The failure cost corresponding to this recourse is  $2d_{0j}$  (with  $d_{00} = 0$ ).

We now consider in detail how the expected cost  $c_r$  of a route  $r \in \mathfrak{R}_e$  can be calculated. For notational convenience, we assume that the route is  $(0, 1, \dots, k, 0)$ .

The total expected cost  $c_r$  can be decomposed into two elements. The first element ( $C_D$ ) is the deterministic cost of following the path  $(0, 1, \dots, k, 0)$ , which must be done irrespective of actual demands. This cost element is simply  $d_{01} + d_{k0} + \sum_{i=1}^{k-1} d_{i,i+1}$ .

The second element ( $C_S$ ) is the cost incurred by the stochastic nature of demands. Generally,  $C_S$  is the expected cost of travel to/from the depot as a result of failures, for the entire route as a whole. We note that the extra traveling represented by  $C_S$  must be done *in addition* to traveling the path represented by  $C_D$ , so that  $C_D$  and  $C_S$  are additive. Indeed, we obtain  $c_r = C_D + C_S$ .

The complicating part of  $c_r$  is the calculation of  $C_S$ . As it turns out, however, we are able to separate  $C_S$  into  $k$  additive terms, one for each customer  $1, \dots, k$ . This is shown in the following.

Let  $u > 0$  be an integer parameter describing the accumulated number of failures. The probability  $F(i|u, Q)$  that the total actual demand on the path  $(0, 1, \dots, i)$  does not exceed  $uQ$  can be calculated as follows:

$$F(i|u, Q) = \Pr\left(\sum_{h=1}^i q_h \leq uQ\right). \quad (4)$$

*Proposition 1:* The probability in (4) depends only on the total expected demand along the path to  $i$ , not on

how this total expected demand is divided among the customers on the path.

*Proof:* Our assumption that the demands are independent Poisson distributions implies that the expression  $\sum_{h=1}^i q_h$  is in itself a Poisson distributed variable with an expected value of  $\sum_{h=1}^i \lambda_h$ . As such, the distribution of  $\sum_{h=1}^i q_h$  does not depend on the individual expected demands, but only on  $\sum_{h=1}^i \lambda_h$ . ■

Proposition 1 is the key to our algorithm for solving the column generation subproblem in Subsection IV-B.

As a consequence of Proposition 1 we can for the remainder of this Section leave the assumption of the path  $(0, \dots, i)$ , and simply consider any elementary path from 0 to  $i$  on which the total expected demand is a given value, say,  $\Lambda$ . We let  $\text{Po}(\Lambda)$  represent any variable which follows a Poisson distribution with an expected value of  $\Lambda$ . In addition, we define  $F(\Lambda, U)$  as the probability that the total actual demand on a path, whose total expected demand is  $\Lambda$ , does not exceed  $U$ , where  $U$  is a positive integer:

$$F(\Lambda, U) = \Pr(\text{Po}(\Lambda) \leq U). \quad (5)$$

We now turn to consider *failures* in more detail.

*Definition 1:* For a given integer  $u \geq 1$  and any elementary path  $(0, \dots, j, i)$  we say that the  $u$ 'th failure occurs at customer  $i$  if and only if the total actual demand on the path  $(0, \dots, j, i)$  exceeds  $uQ$  and the total actual demand on the path  $(0, \dots, j)$  does not exceed  $uQ$ .

Let  $\Lambda$  denote the total expected demand on the path  $(0, \dots, i)$ . The probability that the  $u$ 'th failure occurs at customer  $i$  is then  $F(\Lambda - \lambda_i, uQ) - F(\Lambda, uQ)$ .

For any elementary path  $(0, \dots, i)$  with total expected demand  $\Lambda$ , the expected number of failures  $\text{FAIL}(\Lambda, i)$  at customer  $i$  can then be calculated by summing over all possible failures:

$$\text{FAIL}(\Lambda, i) = \sum_{u=1}^{\infty} F(\Lambda - \lambda_i, uQ) - F(\Lambda, uQ), \quad (6)$$

which in practical computations is approximated by replacing  $\infty$  with some sufficiently large number.

Since the failure cost  $2d_{0i}$  is incurred for every failure at customer  $i$ , the *expected failure cost*  $\text{EFC}(\Lambda, i)$  at customer  $i$ , for any elementary path  $(0, \dots, i)$  with total expected demand  $\Lambda$ , can be calculated as follows:

$$\text{EFC}(\Lambda, i) = 2d_{0i}\text{FAIL}(\Lambda, i). \quad (7)$$

This result is originally obtained in [7]. We note that our assumption of independent Poisson demands permits

a tractable calculation of expected failure costs, in the light of Proposition 1.

#### IV. SOLUTION PROCEDURE

Our solution procedure is based on Dantzig-Wolfe decomposition. As is usual in Set Partitioning based approaches to vehicle routing, we make a few modifications to the formulation in order to obtain a more tractable problem.

##### A. The master problem

To obtain the master problem denoted  $P_M$ , we i) relax the integrality constraints (3), ii) change the partitioning constraints to covering constraints in order to obtain a smaller dual solution space, and iii) enlarge the set of feasible routes by permitting non-elementary paths. This leads to the following master problem:

$$(P_M) \quad \min: \sum_{r \in \mathfrak{R}} c_r x_r \quad (8)$$

$$\text{s.t.: } \sum_{r \in \mathfrak{R}} a_{ir} x_r \geq 1 \quad \forall i \in V_c \quad (9)$$

$$x_r \geq 0 \quad \forall r \in \mathfrak{R} \quad (10)$$

In  $P_M$ , the set  $\mathfrak{R}$  contains all feasible elementary routes as well as all non-elementary routes without 2-cycles ( $i-j-i$ ) on which the total expected demand does not exceed  $Q$ . A customer  $i$  contributes  $\lambda_i$  to the total expected demand on every arrival at  $i$  on the route. The coefficient  $a_{ir}$  equals the number of times that customer  $i$  is visited on route  $r$ .

We initialize  $P_M$  by  $n$  single-customer routes and solve this LP. By solving  $P_M$  a vector of dual prices  $\pi_1, \dots, \pi_n$  is obtained related to the constraint set (9), so that the dual price associated with customer  $i$  is  $\pi_i$ . The dual prices are used in the subproblem in the search for one or more columns with negative reduced cost. If such columns are identified, they are added to the LP, which is then reoptimized. The steps of column generation and LP reoptimization are repeated until no further columns with negative reduced cost exist. The current solution is then optimal for  $P_M$ .

If the LP solution is integer, it is optimal. (If not all inequalities (9) are satisfied with equality in an integer solution, we change the inequalities to equations, resolve the LP, and continue the iterative procedure.) If the LP solution is fractional we resort to branching in order to eventually obtain an integer solution. The overall *branch-and-price* algorithm is a variant of a branch-and-bound

algorithm in which column generation is performed at each node in the branch-and-bound tree.

The two main ingredients: Column generation and Branching, respectively, are described in the following two subsections.

### B. Column generation

In the case of deterministic demands, the column generation subproblem has frequently been solved by a dynamic programming algorithm which effectively solves a shortest path problem on a particular acyclic network. This applies to, e.g., the approaches in [5], [12]. The generated paths are invariably restricted to those without 2-cycles, which can be done without increasing the computational complexity using the idea in [14].

Our column generation approach is quite similar to this, but with the modification that expected failure costs must be taken into account. We note that the calculation of expected failure costs is not affected by permitting non-elementary routes. In the following we describe our construction of the network.

We let  $G_S = (V_S, A_S)$  denote the graph that we construct for the purpose of solving the column generation subproblem.  $V_S$  contains  $(n+1)Q + 1$  vertices. Vertex  $v(0, 0)$  is the beginning of any generated path. Each vertex  $v(\Lambda, i)$ , for  $\Lambda = 1, \dots, Q$  and  $i = 0, \dots, n$ , represents all paths without 2-cycles from 0 to  $i$  on each of which the total expected demand equals  $\Lambda$ .

Beginning with an empty set  $A_S$ , we add arcs to  $A_S$  as follows:

- 1) For  $i = 1, \dots, n$ , add an arc from  $v(0, 0)$  to  $v(\lambda_i, i)$  and set its cost to  $d_{0i} + \text{EFC}(\lambda_i, i) - \pi_i$ .
- 2) For each ordered pair  $i, j \in V_c$ ,  $i \neq j$  and each  $\Lambda = 1, \dots, Q - 1$ , add an arc from  $v(\Lambda, j)$  to  $v(\Lambda + \lambda_i, i)$  (provided that  $\Lambda + \lambda_i \leq Q$ ) and set its cost to  $d_{ji} + \text{EFC}(\Lambda + \lambda_i, i) - \pi_i$ .
- 3) For each  $\Lambda = 1, \dots, Q$  and each  $j = 1, \dots, n$ , add an arc from  $v(\Lambda, j)$  to  $v(\Lambda, 0)$  and set its cost to  $d_{j0}$ .

The shortest path in  $G_S$  from  $v(0, 0)$  to  $v(\Lambda, 0)$ , for some  $\Lambda \in \{1, \dots, Q\}$ , represents the route of minimum reduced cost on which the total expected demand equals  $\Lambda$ . As such, computing the shortest path from  $v(0, 0)$  to  $v(\Lambda, 0)$  for all  $\Lambda = 1, \dots, Q$  effectively solves the column generation subproblem. This can be done in  $O(n^2Q)$  time, also in the case that 2-cycles are prohibited.

### C. The branching strategies

A traditional branching rule is to branch on single flow variables (as in [12]). However, the restriction of forcing two customers to be adjacent is typically more restrictive than the complementary restriction of forcing two customers not to be adjacent. This often leads to unbalanced branch and bound trees.

In order to obtain branch-and-bound trees that are more balanced, we adopt the branching strategy proposed by Gelinas et al. in 1995 for the Vehicle Routing Problem with Time Windows and Backhauls [9]. They introduced a branching procedure based on the time windows. In each branching the parent problem is split into two restricted problems each containing a restricted time window for some customer.

In a similar way, we branch on the capacity resource. If a solution to the master problem is fractional, then either a customer is visited more than once on the same route or a customer is visited on several routes. This means that at least one customer  $i$  will be visited on two paths of the form  $(0, \dots, i)$  with two different accumulated expected demands.

To illustrate this, consider a customer  $i$  which is visited twice, possibly on two different fractional routes, with different total expected demands on the path from the depot. These visits correspond to two vertices  $v(\Lambda_1, i)$  and  $v(\Lambda_2, i)$ , respectively, in  $G_S$  (see Subsection IV-B). As  $\Lambda_1 \neq \Lambda_2$ , this fractional solution can be eliminated by choosing a threshold  $\delta$  between  $\Lambda_1$  and  $\Lambda_2$ , and creating two restricted subproblems as follows. In the first problem, we permit only vertices  $v(\Lambda, i)$  with  $\Lambda \leq \delta$  to be visited on any path in  $G_S$ , and in the second problem, only vertices  $v(\Lambda, i)$  with  $\Lambda > \delta$  are permitted to be visited. In the remainder of the article this strategy is referred to as  $A$ , whereas the original flow variable branching strategy is referred to as  $B$ . Both strategies have been included in the computational testing in order to compare their relative performance.

## V. COMPUTATIONAL RESULTS

The algorithm was tested on several instances originally developed for the CVRP. The data for these CVRP instances are available at [www.branchandcut.org](http://www.branchandcut.org). To convert each CVRP instance into a stochastic instance, the original deterministic demand values were regarded as *expected* values of the stochastic (Poisson distributed) demands in the corresponding CVRPSD instance. For all instances the capacity and demands are divided by their largest common denominator. This decreases the time consumed by the algorithm. However, this must be taken into account when calculating the expected cost.

The instances chosen include all instances of the Augerat test sets A and P, and the Christofides and Eilon test set with up to 60 customers. For the latter; instances E-23-3 and E-30-3 are omitted due to their high vehicle capacity, which leads to computational inaccuracies regarding the calculation of the penalty costs. No test results for instances with more than 60 customers are included since preliminary testing showed that none of these were solvable.

Each instance was run on a Pentium Centrino 1500Mhz computer with 480MB of RAM. For each instance we set a time limit of 1200 seconds. For each instance, data were recorded when the root node was solved, and when the algorithm terminated, either due to timeout or optimality. These data are given in Table I. Columns 2 – 5 show the data collected after solving the root node, whereas columns 6 – 14 show the data collected after the algorithm had terminated. Each column (1-15) will now be discussed in detail.

Column 1 shows the name of the instance. Column 2 shows the objective value after solving the root node. Column 3 shows the number of columns generated at the root node, and column 4 shows the number of times the master problem has been solved. Column 5 shows the sum of the decision variables after having solved the root node. We note that this sum may be integer also for a fractional solution. Indeed, an integer solution is only obtained at the root for the instance P-23-8 in Table I.

Columns 6 and 7 show the total time spent by the algorithm in seconds, when using branching strategy *A* and *B*, respectively (the sign "##" indicates that the algorithm has reached the time limit). Columns 8 and 9 show the number of nodes solved in the branching tree for each of the two branching strategies. Columns 10 and 11 present lower bounds on the optimal objective value, for each of the alternative branching strategies. If a lower bound is marked with an upper case (\*) the solution is optimal. Column 12 shows an upper bound on the optimal objective value. This bound can originate from three different sources: 1) The optimal solution 2) The best integer solution found during branching 3) The best obtainable expected cost given the solution of the CVRP Counterpart (column 14). Column 13 shows the sum of the decision variables, hence also the number of routes in the optimal solution. Column 14 shows the lowest expected cost if the stochastic demand is not included in the route planning; this is calculated on the basis of the optimal solution of its CVRP counterpart. For each route in the optimal solution of the CVRP instance, the minimum expected route cost is calculated, given the customers and their sequence. The expected solution cost is then the sum of the minimum expected

route costs for all routes in the CVRP solution. (If alternative optimal CVRP solutions are known, the alternative resulting in the minimum expected cost is used.) Finally column 15 shows the ratio between the objective value found at the root node and the best known upper bound ( $\frac{LB_{root}}{UB}$ ).

From Table I it can be seen that our algorithm (with a few exceptions) solves all problem instances with up to 40 customers either to optimality or within 1 percent of the optimal solution. On several instances, the number of routes in the optimal solution exceeds the minimum possible for serving all customers (the minimum possible number of routes is the number after the second '-' in the name). For the instance P-55-15 this is particular evident in that three extra routes are formed.

Comparing the best expected solution cost given a deterministic route planning (Column 14) to the optimal expected solution cost, it is clear that neglecting the stochastic nature of demands during the route planning can incur large cost increases. This is especially evident for the instance E-33-4. For this particular instance the extra expected cost is more than 15 percent of the optimal solution cost. This proves that deterministic CVRP models applied to a situation containing stochastic demands could incur larger actual costs than a stochastic model applied to the same situation.

The largest instance successfully solved by the algorithm was an instance with 60 customers and 16 routes. This is considerable progress when comparing to the results obtained in [17] where only instances involving up to 4 vehicles were solved to optimality. Furthermore, the solved problems with more than 50 customers are all characterized by having a small average number of customers on each route. In general the algorithm seems to perform better when the number of required vehicles is large relative to the number of customers. This is contrary to the results in [17] where the computational effort required increases sharply with the number of routes. This leads to the conclusion that our algorithm broadens the range of VRPSD instances solvable, hence strengthens the applicability of VRPSD models.

The lower bound obtained at the root is very close to the optimal solution for all instances solved. Despite this fact the number of nodes in the branch-and-bound tree can be very large. This is particularly clear for the instance P-55-15. For this instance the solution at the root was within half a percent of the optimal solution, even so the number of nodes in the tree exceeded twenty thousand, when using strategy A and thirty thousand when using strategy B. This can be partly explained by the nature of the failure costs. Regardless of the direction

TABLE I  
COMPUTATIONAL RESULTS

Inst.	LB Root	Cols.	$P^M$	Routes Root	Time A	Time B	Nodes A	Nodes B	LB A	LB B	UB	Routes Opt	Best Det	$\frac{LB_{Root}}{UB}$
A-32-5	817.31	1221	67	5.00	257	##	2467	10223	856.3*	841.12	856.30	5	890.13	0.95
A-33-5	700.01	962	44	5.00	8	7	117	107	704.20*	704.20*	704.20	5	722.99	0.99
A-33-6	775.00	832	43	6.05	43	##	893	20329	793.9*	789.18	793.90	6	816.58	0.98
A-34-5	803.26	1212	62	5.31	##	##	11559	11355	824.34	822.42	826.87	-	839.95	0.97
A-36-5	838.83	1641	113	5.00	##	##	5809	5869	851.37	850.29	907.55	-	907.55	0.92
A-37-5	687.40	1758	95	5.00	##	##	6619	6521	706.54	702.35	708.34	-	709.83	0.97
A-37-6	1007.98	1237	67	6.48	##	##	12307	10673	1029.71	1020.96	1030.76	-	1069.32	0.98
A-38-5	739.19	1207	55	5.47	##	##	10631	10447	760.41	754.45	778.09	-	831.99	0.95
A-39-5	866.92	1722	73	6.07	2	3	9	23	869.18*	869.18*	869.18	6	903.26	1.00
A-39-6	850.09	1462	84	6.04	242	##	2453	8193	876.6*	870.15	876.60	6	960.81	0.97
A-44-7	1007.55	1871	59	6.43	##	##	8101	7405	1020.67	1016.80	1025.48	-	1047.18	0.98
A-45-6	984.38	1861	95	6.75	##	##	6631	7887	1005.95	1000.26	1096.19	-	1096.19	0.90
A-45-7	1254.23	1954	84	7.13	794	##	5365	6955	1264.83*	1262.63	1264.83	7	1302.20	0.99
A-46-7	986.39	1951	84	7.00	##	##	5433	6789	999.08	996.67	1069.66	-	1069.66	0.92
A-48-7	1160.52	2482	91	7.13	##	##	5237	6061	1179.37	1177.23	1248.27	-	1248.27	0.93
A-53-7	1093.64	3273	128	7.69	##	##	4003	4025	1108.67	1106.22	1180.10	-	1180.10	0.93
A-54-7	1262.49	2636	98	7.44	##	##	4347	4195	1279.46	1272.17	1342.87	-	1342.87	0.94
A-55-9	1148.40	1881	55	9.76	##	##	7427	7237	1172.58	1159.20	1264.18	-	1264.18	0.91
A-60-9	1489.82	3043	108	9.20	##	##	4777	4197	1502.90	1497.87	1608.40	-	1608.40	0.93
E-22-4	409.86	460	34	4.20	1	1	9	13	411.73*	411.73*	411.73	4	411.73	1.00
E-33-4	844.35	3030	148	4.00	74	##	73	879	850.27*	849.66	850.27	4	984.28	0.99
E-51-5	538.75	3290	138	5.48	##	##	2035	1925	544.34	543.38	553.26	-	553.26	0.97
P-16-8	511.27	79	8	8.50	1	1	11	19	511.00*	511.00*	511.00	8	512.82	1.00
P-19-2	210.90	620	63	2.17	131	138	1815	1529	224.06*	224.06*	224.06	3	229.68	0.94
P-20-2	221.11	871	88	2.18	298	450	3191	3819	233.05*	233.05*	233.05	2	233.05	0.95
P-21-2	217.75	1037	100	2.14	4	6	27	45	218.96*	218.96*	218.96	2	218.96	0.99
P-22-2	223.67	1129	103	2.21	186	297	1335	1903	231.26*	231.26*	231.26	2	231.26	0.97
P-22-8	677.97	205	18	8.92	1	1	63	55	680.10*	680.10*	680.10	9	707.80	1.00
P-23-8	619.52	22	19	9.00	1	1	0	0	619.52*	619.52*	619.52	9	662.31	0.94
P-40-5	471.47	1905	80	5.00	8	5	35	29	472.5*	472.5*	472.50	5	475.45	1.00
P-45-5	519.03	2153	101	5.14	##	##	3379	3219	527.34	526.24	546.05	-	546.05	0.24
P-50-7	573.66	2016	67	7.13	##	##	3925	3909	580.85	579.63	606.41	-	606.41	0.95
P-50-8	659.67	1522	55	8.78	##	##	7689	7921	666.54	666.53	724.69	-	724.69	0.91
P-50-10	750.27	1320	50	11.00	##	##	14453	14301	756.27	757.17	792.20	-	792.20	0.95
P-51-10	802.58	1436	57	10.99	440	1024	5347	12329	809.70*	809.70*	809.70	11	859.24	0.99
P-55-7	582.12	2782	102	7.00	##	##	2301	2533	585.43	585.28	616.44	-	616.44	0.94
P-55-8	599.67	2457	87	7.31	##	##	2777	2965	603.42	603.49	2438.00†	-	2438.00†	0.25
P-55-10	734.69	1747	56	10.17	##	##	8181	8779	739.75	741.00	797.21	-	797.21	0.92
P-55-15	1062.67	867	21	17.18	838	##	20355	32751	1068.05*	1067.90	1068.05	18	1191.34	0.99
P-60-10	793.71	2102	59	10.58	##	##	5817	5307	798.77	799.32	831.24	-	831.24	0.96
P-60-15	1080.85	1175	29	16.08	##	1167	20581	18483	1085.22	1085.49*	1085.49	16	1133.30	1.00

†upper bound = total expected cost if all customers were serviced on separate routes

in which a route is traveled the failure cost tends to be very low for the customers at the beginning of the route until the accumulated expected demand approaches  $Q$ . From here the failure cost increases rapidly. The route's total failure cost may change only slightly when changing the direction of the route. This element of near-symmetry seems to make the branching less effective. However, when considering the number of problems solved by the each of the two branching strategies, strategy A solves more problems than does strategy B.

Furthermore, when comparing the instances solved by both branching strategies, strategy A tends to search fewer nodes before reaching optimum than does strategy B. These two facts indicate that strategy A in general performs better than strategy B.

## VI. CONCLUDING REMARKS

In this article we have introduced a new branch-and-price algorithm for solving the Capacitated Vehicle Routing Problem with Stochastic Demands (CVRPSD),

where the objective is to minimize the expected solution cost. We show that under the assumption of independently Poisson distributed demands the column generation problem can be formulated as a shortest path problem on an acyclic network and solved by dynamic programming.

The algorithm was tested on a large number of CVRP instances that were converted into stochastic instances. The algorithm showed good results by solving almost all instances with up to around forty customers, and by solving a few instances with over fifty customers and more than 10 vehicles. This is a significant progress compared to previous work done on the CVRPSD. Moreover, we proposed a new branching strategy for the VRPSD based on accumulated demand, which shows some potential, when compared to a well known flow-variable based branching strategy.

## REFERENCES

- [1] M. Ball, T. Magnanti, C. Monma, and G. Nemhauser, Eds., *Network Routing*. North-Holland: Elsevier Science Publisher B.V., 1995.
- [2] C. Bastian and A. R. Kan, "The stochastic vehicle routing problem revisited," *European Journal of Operational Research*, vol. 56, pp. 407–411, 1992.
- [3] J. Beasley, "Fixed routes," *Journal of the Operational Research Society*, vol. 35, pp. 49–55, 1984.
- [4] D. Bertsimas, "A vehicle routing problem with stochastic demand," *Operations Research*, vol. 40, pp. 574–585, 1992.
- [5] N. Christofides, A. Mingozzi, and P. Toth, "Exact algorithm for the vehicle routing problem, based on spanning tree and shortest path relaxations," *Mathematical Programming*, vol. 20, pp. 255–282, 1981.
- [6] G. Clarke and J. W. Wright, "Scheduling of vehicles from a central depot to a number of delivery points," *Operations Research*, vol. 12, pp. 568–581, 1964.
- [7] M. Dror, G. Laporte, and P. Trudeau, "Vehicle routing with stochastic demands: properties and solution frameworks," *Transportation Science*, vol. 23, pp. 166–176, 1989.
- [8] M. Dror and P. Trudeau, "Stochastic vehicle routing with modified savings algorithm," *European Journal of Operational Research*, vol. 23, pp. 228–235, 1986.
- [9] S. Gélinas, M. Desrochers, J. Desrosiers, and M. Solomon, "A new branching strategy for time constrained routing problems with application to backhauling," *Annals of Operations Research*, vol. 61, pp. 91–109, 1995.
- [10] M. Gendreau, G. Laporte, and R. Séguin, "An exact algorithm for the vehicle routing problem with stochastic demands and customers," *Transportation Science*, vol. 29, pp. 143–155, 1995.
- [11] M. Gendreau, G. Laporte, and R. Séguin, "Stochastic vehicle routing," *European Journal of Operational Research*, vol. 88, pp. 3–12, 1996.
- [12] E. Hadjiconstantinou, N. Christofides, and A. Mingozzi, "A new exact algorithm for the vehicle routing problem based on  $q$ -paths and  $k$ -shortest paths relaxation," *Annals of Operations Research*, vol. 61, pp. 21–43, 1995.
- [13] M. Haughton, "Route reoptimization's impact on delivery efficiency," *Transportation Research Part e*, vol. 38, pp. 53–63, 2002.
- [14] D. Houck, J. Picard, M. Queyranne, and R. Vemuganti, "The travelling salesman problem as a constrained shortest path problem: theory and computational experience," *Opsearch*, vol. 17, pp. 93–109, 1980.
- [15] G. Laporte and F. Louveaux, "The integer L-shaped method for stochastic integer programs with complete recourse," *Operations Research Letters*, vol. 13, pp. 133–142, 1993.
- [16] —, "Solving stochastic routing problems with the integer L-shaped method," in *Fleet management and logistics*, T. Crainic and G. Laporte, Eds. United states: Kluwer Academic Publishers, 1998, ch. 7, pp. 159–167.
- [17] G. Laporte, F. Louveaux, and L. Hamme, "An integer L-shaped algorithm for the capacitated vehicle routing problem with stochastic demands," *Operations Research*, vol. 50, pp. 415–423, 2002.
- [18] M. Savelsbergh and M. Goetschalckx, "A comparison of the efficiency of fixed versus variable vehicle routes," *Journal of Business Logistics*, vol. 16, pp. 163–187, 1995.
- [19] W. Stewart and B. Golden, "Stochastic vehicle routing: a comprehensive approach," *European Journal of Operational Research*, vol. 14, pp. 371–385, 1982.
- [20] F. Tillman, "The multiple terminal delivery problem with probabilistic demands," *Transportation Science*, vol. 3, pp. 192–204, 1969.
- [21] P. Toth and D. Vigo, Eds., *The vehicle routing problem*. Philadelphia: SIAM Monographs on Discrete Mathematics and Applications, 2002.

# Integrating nurse and surgery scheduling

Jeroen Beliën\* and Erik Demeulemeester†

\*Katholieke Universiteit Leuven, Department of Applied Economics  
Naamsestraat 69, B-3000 Leuven, Belgium

Email: jeroen.beliën@econ.kuleuven.be

†Katholieke Universiteit Leuven, Department of Applied Economics  
Naamsestraat 69, B-3000 Leuven, Belgium  
Email: erik.demeulemeester@econ.kuleuven.be

**Abstract**—One common problem at hospitals is the extreme variation in daily (even hourly) workload pressure for nurses. The operating room is considered to be the main engine and hence the main generator of variance in the hospital. It is our belief that integrating the operation room scheduling process with the nurse scheduling process is a simple, yet effective way to achieve considerable savings in staffing costs. The purpose of this paper is threefold. First of all, we present a concrete model that integrates both the nurse and the operating room scheduling process. Secondly, we show how the column generation technique approach, often employed for nurse scheduling problems, can easily cope with this model extension. Thirdly, by means of a large number of computational experiments we provide an idea of the cost saving opportunities and required solution times.

**Keywords**—nurse scheduling, surgery scheduling, column generation, integer programming.

## I. INTRODUCTION

DURING the last decades, cost pressures on hospitals have increased dramatically. This emphasis on cost containment has forced hospital executives to run their organizations in a more business-like manner. The constant challenge is to provide high-quality service at ever reduced costs. In order to achieve this purpose, inefficient use of resources should be identified and actions should be taken to eliminate these sources of waste. Operations research techniques are increasingly being used to assist in this complicated task.

As nursing services account for an important part of a hospital's annual operating budget, concentrating on this resource can lead to substantial savings. The situation is exacerbated by an acute shortage of nurses in all western countries, said to be 120,000 today and expected to grow to 808,000 by 2020 in the United States (US) alone [24]. Hence, it is of vital importance that nurses are used as much as possible at the right time and at the right place. This goal is hard to achieve because of two reasons. The first one is inherent in service organizations

for which human resources outnumber all other types of resources. Unlike machines, staff schedules are restricted by collective agreement requirements. These form an important hindrance for the flexibility with which nurses are scheduled.

A second reason is the presence of variability. Variability is probably the main obstacle to efficient delivery of health care and reducing it is one of the major concerns in current health care management [19]. Compared with industrial environments, hospitals are much more stochastic by nature. One common problem at hospitals is the extreme variation in daily (even hourly) workload pressure for nurses. On days when the workload is too high, the quality of care decreases because it is too costly to staff for peak loads. On days when the workload is too low, there is waste. Fortunately, the situation is not as chaotic as it seems to be at first sight. As pointed out in [19], an important amount of the variability can effectively be managed and reduced by a thorough analysis of the existing system and by appropriate decision-taking. Special emphasis is put on the operating room since it is considered the main engine and hence the main generator of variance in the hospital. It is our belief that integrating the operation room schedule process into the nurse scheduling process is a simple yet effective way to achieve considerable savings in staffing costs.

This paper is organized as follows. In Section II a discussion of the background together with a brief literature review is given. In Section III a general overview of the model together with a branch-and-price solution approach is presented. Section IV provides more details on both pricing problems, while a general overview of the branch-and-price algorithm is given in Section V. Section VI discusses a specific branching scheme. In Section VII some computational issues are discussed and in Section VIII extensive computational results are given. Finally, Section IX draws conclusions and lists some topics for further research.

## II. BACKGROUND AND LITERATURE REVIEW

Nurse scheduling problems are frequently encountered in the operations research literature. Recently, a good bibliographic survey on medical staff rostering problems has appeared [13]. Several studies in the literature have utilized mathematical programming techniques to assist in finding efficient staff schedules (see e.g. [22], [28], [3], [8], [12], [4]). These problems typically involve some kind of set covering or set partitioning formulation. The main drawback, however, is that these models can have far more variables than can be reasonably attacked directly. Therefore, the linear program (LP) is often solved using column generation (see e.g. [18], [5] and [6], [21], [20]). To the best of our knowledge, all the proposed models consider the nurse scheduling problem as a separate problem, i.e. not related to any other activity in the hospital. In this paper we will describe a more general approach in which the demand constraints are dependent on the operation room schedule and hence become a part of the decision process.

The operations research literature is replete with examples of integer programming techniques being applied to operating room scheduling problems. This work can be categorized based on the stage of the scheduling process to which it applies. Developing operating room (OR) schedules can be seen as a three stage process. In a first stage the available OR time is divided over the different surgeons (or surgical groups). This first phase is also referred to as case mix planning, since it determines for which pathologies capacity will be preserved. Hughes and Soliman [17] propose a linear programming model to solve case mix planning problems. Dexter and Macario [14] argue that OR time should be allocated to maximize OR efficiency instead of "fixed hours" blocks based on historical utilization data. Blake and Carter [10] propose a methodology that uses two linear goal programming models. One model sets case mix and volume for physicians, while holding service costs fixed; the other translates case mix decisions into a commensurate set of practical changes for physicians.

Once the OR time allocated to each surgical group has been chosen, the second stage involves the development of a master surgery schedule. The master surgery schedule is a cyclic timetable that defines the number and type of operating rooms available, the hours that rooms will be open, and the surgical groups or surgeons who are to be given priority for the operating room time. Compared to case mix planning (first stage) and elective case scheduling (third stage), the literature on master surgery scheduling is rather scant. Blake et al. [11] propose an integer programming model that minimizes the weighted

average undersupply of OR hours (i.e. allocating to each surgical group a number of OR hours as close as possible to its target OR hours).

After the development of the master surgery schedule, elective cases can be scheduled. This third stage occurs on a daily base and involves detailed planning of each intervention. Each patient needs a particular surgical procedure, which defines the human (surgeon) and material (equipment) resources to use and the intervention duration. Guinet and Chaabane [15] define this problem as a general assignment problem and propose a primal-dual heuristic to solve it. Weiss [29] deals with the problem of determining the case orderings and presents both analytical and simulation results.

The methodology presented in this paper has some similarities with models for integrating the scheduling of project tasks and employees (Alfares et al. [1] and Alfares and Bailey [2]). Although several authors mention the interdependency between the surgery scheduling process and the development of nurse rosters, as far as we know, no models have been proposed to integrate both areas of decision-making. Litvak and Long [19] underline the negative impact of variability in hospital environments. They consider the operating room as the engine that drives the hospital. Consequently, the activities inside the operation room heavily determine the fluctuations in resource demands throughout the rest of the hospital. A poor operating room schedule could for instance be directly responsible for the occurrence of (contra-productive) peaks in the demand for certain types of resources. The authors distinguish between two types of variability: natural variability and artificial variability. Natural variability is inherent to the uncertain world of health care. This variability arises from uncertainty in patient show-ups, uncertainty in recovery time, uncertainty in the effectiveness of therapies etc.... Artificial variability originates from poor scheduling policies. Belien and Demeulemeester [9] have elaborated this idea. They propose a number of integer programming models for building robust surgery schedules for which the resulting expected bed shortage is minimized.

In this paper the master surgery schedule is being considered as the main generator of the workload of the nurses. In order to couple both scheduling environments, the objective in the surgery schedule process will be to construct a favorable workload distribution for the nurses.

## III. MODEL DESCRIPTION

### A. General idea

Figure 1 contains a schematic overview of the general idea elaborated in this paper.

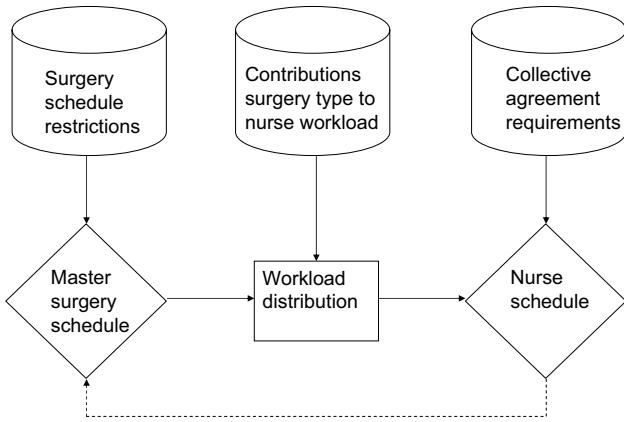


Fig. 1. Schematic overview of the general idea

First have a look at the nurse scheduling process at the right of this figure. The input for the nurse scheduling process consists of the restrictions implied on the individual nurse roster lines on the one hand and the workload distribution over time on the other hand. The workload distribution itself is determined by the master surgery schedule. In order to be able to deduce the workload from the surgery schedule one also has to know the workload contributions of each specific type of surgery. The dotted arrow at the bottom indicates the feedback that could be given from the nurse scheduling process to the surgery scheduling process in order to produce more favorable surgery schedules with respect to the resulting workloads. The freedom in modifying the surgery schedule is however limited, since the master surgery schedule itself is restricted by a set of specific surgery constraints (e.g. capacity and demand constraints). It must be clear, however, that integrating the surgery scheduling process with the nurse scheduling process provides more flexibility in building the nurse schedules, since one has an instrument to make the workload distribution fit for the nurse schedules.

In what follows we will describe a mathematical model for implementing this idea. Therefore, we start with stating the standard nurse scheduling problem and discuss the column generation solution procedure for solving it. Then, we extend this model with the extra decision of the nurse scheduling process and show how the column generation solution procedure can easily cope with this extension. Hereby, we focus on the minimization of the total required number of nurses. The reason for this objective is that it allows for a quantitative measure of the resulting benefits, i.e. the decrease in staffing cost. Obviously, this quantitative

benefit can easily be turned into a qualitative benefit by employing the saved nurse(s) on moments when they are most needed.

### B. The nurse scheduling problem

The nurse scheduling problem (NSP) consists of generating a configuration of individual schedules over a given time horizon. The configuration of nurse schedules is generated so as to fulfill collective agreement requirements and the hospital staffing demand coverage while minimizing the salary cost. An individual's *roster line* can be viewed as a sequence of *days on* and *days off*, where each day on contains a single *shift* identified by a label such as 'day', 'evening' or 'night'. Each such label coincides with a start and a finish time of the corresponding shift. Furthermore, a day is subdivided into several *demand periods* characterized by fixed starting and ending times. These demand periods do not necessarily coincide with the shifts. However, the demand per shift can easily be determined.

Coverage constraints imply how many nurses of appropriate skills have to be scheduled for each demand period. For ease of exposition and without loss of generalization we consider all nurses equally-skilled throughout the rest of this paper.

Collective agreement requirements are rules that define acceptable schedules for individual nurses in terms of total workload, holidays, weekends off and shift transitions (e.g. a morning shift after a night shift is not allowed). These rules cannot be violated and dramatically reduce the set of feasible individual roster lines. Obviously, when building nurse schedules also a set of individual constraints, often called preference constraints, have to be taken into account. For instance, some nurses prefer to do night shifts, others do not. Again, for ease of exposition and without loss of generalization, we make abstraction of these differences in individual preferences and only consider those restrictions which are stated in the collective agreement rules and consequently apply on all nurses. Hence, we present an integrated model that can be used to find optimal schedules for a homogeneous set of nurses.

In what follows we state the standard set covering model, which is often used for this type of problems. Let  $J$  be the set of feasible roster lines  $j$  and  $I$  be the set of demand periods  $i$ . Let  $d_i \in \mathbb{R}^+$ ,  $\forall i \in I$ , denote the required number of nurses scheduled during period  $i$ . Furthermore, let  $a_{ij}$  be 1 if roster line  $j$  contains an active shift during period  $i$  and 0 otherwise. The general integer decision variable  $x_j$ ,  $\forall j \in J$ , indicates the number of individual nurses which are scheduled by roster line  $j$ .

Then, the nurse scheduling problem (NSP) can be stated as follows:

$$\text{Minimize} \sum_{j \in J} x_j \quad (1)$$

subject to:

$$\sum_{j \in J} a_{ij} x_j \geq d_i \quad \forall i \in I \quad (2)$$

$$x_j \in \{0, 1, 2, \dots\} \quad \forall j \in J \quad (3)$$

### C. Solution procedure for the nurse scheduling problem

The integer program (IP) (1)-(3) is solved by first solving the linear programming relaxation and then using a branching scheme to drive the solution into integrality. As the number of possible roster lines an individual can work is usually too large to allow complete, a-priori enumeration, column generation is often applied to solve the LP relaxation. Typically, the pricing step involves the solution of a dynamic programming shortest path problem (also called the *subproblem*) to find the legal column with the most negative reduced cost. Let  $\pi_i, \forall i \in I$ , denote the dual price of constraint (2). Then, the reduced cost of a new column (roster line)  $j$  is given by:

$$1 - \sum_{i \in I} a_{ij} \pi_i \quad (4)$$

A brief discussion of the solution procedure for this subproblem is given in Section IV-A. The process of adding new columns continues until no more columns *price out*, i.e. no more columns with negative reduced cost can be found. However, at that point, the solution is not necessarily integral and applying a standard branch-and-bound procedure to the restricted master with its existing columns will not guarantee an optimal (or feasible) solution. Therefore, a branching scheme has to be applied to drive the solution into integrality. After branching, new columns might price out favorably and hence have to be added to the model.

Since it lies not in the scope of this paper to discuss effective branching schemes for the NSP, we will not go into details about this, but instead refer the reader to the specialized literature. Barnhart et al. [7] discuss appropriate branching strategies for solving a mixed integer program (MIP) using column generation. Since NSP (1)-(3) has identical restrictions on subsets (i.e. there are no subsets having a separate convexity constraint), elaborating a branching scheme is a complex

issue. Conventional integer programming branching on variables is not effective for reasons of symmetry and also because fixing variables destroys the structure of the subproblem. Vanderbeck and Wolsey [27] developed a general rule in which one is branching on the constraints (see also [26]). The drawback is that the branching constraints cannot be used to eliminate variables and have to be added to the formulation explicitly. Hence, each branching constraint will contribute an additional dual variable to the reduced cost, which complicates the pricing problem.

### D. The generalized nurse scheduling problem

In the NSP the right hand side values of the coverage constraints (i.e. the  $d_i$ 's in formulation (1)-(3)) are considered to be fixed. Nevertheless, coverage constraints are based on workload estimations which entail the summations of individual patient *workload contributions*. An individual patient workload contribution is determined by the *patient type*. The patient type can generally be described by three dimensions. The first dimension is the type of surgery the patient has undergone. The second is the number of periods the patient has already recovered. The third is the period to which the workload applies. For instance, some pathologies may require increased care during nights.

The number and type of the patients that are present in the hospital at each moment in time is largely determined by the operation room schedule. Obviously, due to emergency cases and uncertainty in patient show-ups, patient recovery times etc..., exact estimations are not possible. However, an in-depth analysis of the operation room schedule enables hospital executives to make a quite accurate prediction of the workload of the nurses. Moreover, they can reshape the workload distribution by modifying the operation room schedule. In the long term case mix planning decisions determine the overall workload. In shorter term the cyclic master surgery schedule determines the workload distribution over time.

The generalized nurse scheduling problem (GNSP) takes into account this extra dimension. Instead of assuming the demand values to be fixed, we consider them to be dependent on the number and type of patients undergoing surgery in the hospital at each moment. By manipulating the master surgery schedule, hospital management can create (and choose between) a number of different workload distributions, further referred to as *workload patterns*. Let  $K$  denote the set of possible workload patterns that could be generated by modifying the surgery schedule. These will be obtained by enumerating all possible ways of assigning operating blocks to

the different surgeons, subject to surgery demand and capacity restrictions (for more details see Section IV-B). Each workload pattern  $k$  is described by a number of periodic demands  $d_{ik} \in \{0, 1, 2, \dots\}$ ,  $\forall i \in I$ . Let  $z_k$  be 1 if the surgery schedule that corresponds to workload  $k$  is chosen and 0 otherwise. Then, the problem can be stated as follows:

$$\text{Minimize} \sum_{j \in J} x_j \quad (5)$$

subject to:

$$\sum_{j \in J} a_{ij} x_j \geq \sum_{k \in K} d_{ik} z_k \quad \forall i \in I \quad (6)$$

$$\sum_{k \in K} z_k = 1 \quad (7)$$

$$x_j \in \{0, 1, 2, \dots\} \quad \forall j \in J \quad (8)$$

$$z_k \in \{0, 1\} \quad \forall k \in K \quad (9)$$

Constraint (7), further referred to as the workload convexity constraint, implies that exactly one workload pattern has to be chosen. In a feasible solution all  $z_k$ 's but one equal 0. Hence, in constraint (6) only the corresponding  $d_{ik}$ 's are added in the right hand side values. It is easy to see that the NSP is a special case of the GNSP in which one  $z_k$  is fixed to be 1.

#### E. Solution procedure for the generalized nurse scheduling problem

In this part we show that the column generation approach to solve the LP relaxation of NSP can easily be extended to cope with the GNSP. Similarly to the roster lines, the number of possible workload patterns is usually too large to allow for complete, a-priori enumeration. Also here, the process starts with a limited subset of workload patterns and new patterns (columns) are added as needed. Therefore, a second subproblem has to be solved. The generation of a new workload pattern boils down to the construction of a new master surgery schedule. The subproblem is constrained by a set of specific surgery schedule restrictions. Its objective is the minimization of the reduced cost of a new workload pattern. Let  $\gamma$  denote the dual price of the workload pattern convexity constraint (7). Then, the reduced cost of a new workload pattern  $k$  is given by:

$$0 - \gamma + \sum_{i \in I} \pi_i d_{ik} \quad (10)$$

Obviously, the appropriate solution approach to price out a new workload pattern strongly depends on the characteristics of the master surgery schedule. In this

paper the workload pattern pricing problem is formulated as an IP and solved using a state-of-the-art optimization package (CPLEX). More details on this formulation can be found in Section IV-B.

## IV. PRICING PROBLEMS

### A. Generating a new roster line

Although the generation of a new roster line happens in a standard way (shortest path problem solved with recursive dynamic programming) (see e.g. [12]) and its exact implementation is not really necessary for understanding the general idea of this paper, we briefly discuss the procedure. First, we summarize the restrictions which apply to a roster line.

As already mentioned earlier, this work is only concerned with collective agreement requirements and leaves individual preferences out of consideration. Concretely, we take into account five types of requirements when building a new roster line. First of all, a nurse cannot work more than one shift per day. Secondly, the overall number of *active days*, i.e. days in which the roster line contains an *active shift* ("day", "evening" or "night"), cannot exceed a certain limit. Thirdly, the maximum number of *consecutive working days* is also constrained. The same holds for the maximum number of consecutive rest days. A sequence of working days is further referred to as a *block*. Fourthly, the number of so-called unpopular shifts (night shifts, weekend shifts) is limited per roster line. Fifthly, in a block, certain shift transitions are not allowed. For instance, a nurse cannot switch from, say, a night shift to a morning shift without having a rest first.

Generating a new roster line is typically done using a dynamic programming recursion. To this aim, we define a table giving the minimum cost that can be achieved in days 1 to  $d$  by a roster line that, starting from a situation in which on day  $d$  a shift  $s$  is scheduled and in which between days  $d$  to  $n$  a certain number of active shifts  $f$  occurred, a certain number of unpopular shifts  $g$  occurred and a number of consecutive working or rest days  $h$  (including day  $d$ ) is assigned. Formally, the entries of the table are of the form

$$\tau(d, f, g, s, h),$$

defined for  $d = 1..n$ ,  $f = 0..f_{max}$ ,  $g = 0..g_{max}$ ,  $s \in S$ ,  $h = 0..h_{max}$ . Hereby,  $n$  denotes the number of days in the scheduling horizon,  $f_{max}$  denotes the maximum number of working days in a roster line,  $g_{max}$  is the maximum penalty in terms of unpopular shifts,  $S$  is the set of shift types ("day", "evening", "night", "rest") and  $h_{max}$  is the maximum of both the maximum number

of consecutive working days ( $h_1^{max}$ ) and the maximum number of consecutive rest days ( $h_2^{max}$ ). Let  $p_{d,s}$  be the penalty cost for assigning an unpopular shift  $(d, s)$ . Let  $A$  denote the set of allowed shift transitions  $(s, s')$  between two consecutive days on. We consider demand periods as being subsets of the shifts, i.e. no demand period can be spread over more than one shift. However, a shift can consist of more demand periods. Let  $Q_{(d,s)}$  be the set of demand periods  $i$  that fall into shift  $(d, s)$ . Let  $\lambda_{d,s}$  be the total dual cost of a shift  $(d, s)$ , i.e.  $\lambda_{d,s} = \sum_{i \in Q_{(d,s)}} \pi_i$ .

The computation of the entries in the table is done by starting at the beginning of the time horizon and working forward by considering an insertion of a shift type  $s$  on the next day  $d$  of the roster line associated with an entry already computed. Therefore, we make use of recursive algorithm 1.

---

**Algorithm 1** RECURSION( $d, f, g, s, h$ )

---

```

if (d=0) then
    return 0; {beginning of time horizon reached}
else if ( $\tau(d, f, g, s, h) \neq 999999999$ ) then
    return  $\tau(d, f, g, s, h)$ ; {state already visited, can be pruned}
else
    cost  $\leftarrow +\infty$ ;
    min_cost  $\leftarrow +\infty$ ;
    for (all shifts  $\bar{s} \in S \setminus \{"rest"\}$ ) do
        if ( $g + p_{d-1, \bar{s}} \leq g_{max}$ ) AND  $((\bar{s}, s) \in A)$  AND  $(f < f_{max})$  then
            if ( $s \neq "rest"$ ) then
                if ( $h < h_{max}^1$ ) then
                    cost  $\leftarrow \lambda_{d,s} +$ RECURSION( $d-1, f+1, g+p_{d-1, \bar{s}}, \bar{s}, h+1$ );
                    {successive active shift}
                end if
            else if ( $s = "rest"$ ) then
                cost  $\leftarrow$ RECURSION( $d-1, f+1, g+p_{d-1, \bar{s}}, \bar{s}, 1$ ); {start active
shift}
            end if
        end if
        if ( $cost < min\_cost$ ) then
            min_cost  $\leftarrow cost$ ;
        end if
    end for
    if ( $s \neq "rest"$ ) then
        cost  $\leftarrow \lambda_{d,s} +$ RECURSION( $d-1, f, g, "rest", 1$ ); {start rest}
    else if ( $s = "rest"$ ) then
        if ( $h < h_{max}^2$ ) then
            cost  $\leftarrow$ RECURSION( $d-1, f, g, "rest", h+1$ ); {successive rest}
        end if
    end if
    if ( $cost < min\_cost$ ) then
        min_cost  $\leftarrow cost$ ;
    end if
    return  $\tau(d, f, g, s, h) \leftarrow min\_cost$ ;
end if

```

---

Before starting the recursion all entries of table  $\tau(d, f, g, s, h)$  are initialized to 999999999. The minimal reduced cost of a new roster line can now easily be calculated by starting the recursion on day  $n$  and minimizing over each shift type (see algorithm 2).

Once all the calculations are done, the best new roster line can easily be constructed backward. The overall

---

**Algorithm 2** FIND-NEW-ROSTER-LINE

---

```

{initialize all entries of  $\tau$ }
for (d = 1 to n) do
    for ( $f = 0$  to  $f_{max}$ ) do
        for ( $g = 0$  to  $g_{max}$ ) do
            for (all shifts  $s \in S$ ) do
                for ( $h = 0$  to  $h_{max}$ ) do
                     $\tau(d, f, g, s, h) \leftarrow 999999999$ ;
                end for
            end for
        end for
    end for
end for
cost  $\leftarrow +\infty$ ;
min_cost  $\leftarrow +\infty$ ;
{start the recursion}
for (all shifts  $\bar{s} \in S \setminus \{"rest"\}$ ) do
    if ( $p_{n, \bar{s}} \leq g_{max}$ ) then
        cost  $\leftarrow$ RECURSION( $n, 1, p_{n, \bar{s}}, \bar{s}, 1$ ); {end with an active shift}
    end if
    if ( $cost < min\_cost$ ) then
        min_cost  $\leftarrow cost$ ;
    end if
end for
cost  $\leftarrow$ RECURSION( $n, 0, 0, "rest", 1$ ); {end with a rest}
if ( $cost < min\_cost$ ) then
    min_cost  $\leftarrow cost$ ;
end if

```

---

space complexity of the dynamic programming recursion is

$$O(n \cdot f_{max} \cdot g_{max} \cdot |S| \cdot h_{max})$$

whereas the time complexity is (in the case that there are no forbidden shift transitions),

$$O(n \cdot f_{max} \cdot g_{max} \cdot |S| \cdot h_{max} \cdot |S|)$$

since each entry of the table is updated by considering up to  $O(|S|)$  other entries.

### B. Generating a new workload pattern

Each workload pattern corresponds to a particular surgery schedule. Hence, a new workload pattern can be obtained by building a new surgery schedule. Hereby, the capacity preserved for the different surgeons (or, more generally, surgery groups) is already determined by the case mix planning (first stage, long term) and considered to be fixed in our application. Elective case scheduling (third stage) is also left out of consideration because of two reasons. First of all, the impact of each specific elective case on the workload is rather scant. It is the type of surgery that determines the workload contribution, not the individual case. Secondly, it is very hard to predict the precise impact of the individual cases on the workload contribution at the moment that the nurse rosters have to be built. Often, at that moment, an important part of the elective surgery scheduling is still to be done.

The master surgery schedule is considered to be the tool for manipulating the workload distribution over time. This work is concerned with *cyclic* master surgery schedules. Cyclic schedules are schedules that are repeated after a certain time period (referred to as the cycle time). During such a cycle time there might be a number of time periods during which surgery cannot take place. These periods are referred to as the inactive periods, the others are active. Typically, cycle times are multitudes of weeks in which the weekends are inactive periods.

In our application, a new surgery schedule is built by solving an integer program. To find a new workload pattern with minimal reduced cost given the current set of roster lines and workload patterns, the objective function minimizes the dual price vector of the demand constraints (6) multiplied by the new demands. We deal with two types of constraints. Surgery demand constraints determine how many blocks must be preserved for each surgeon. Capacity constraints ensure that the number of blocks assigned during each period do not exceed the available capacity. Let  $y_{rt}$  ( $\forall r \in R$  and  $t \in T$ ) be the number of blocks assigned to surgeon  $r$  in period  $t$ . Hereby,  $T$  represents the set of active periods and  $R$  the set of surgeons. Let  $q_r$  be the number of blocks required by each surgeon  $r$ . Let  $b_t$  be the maximal number of blocks available in period  $t$ . Let  $w_{rti} \in \mathbb{R}^+$  denote the contribution to the workload of demand period  $i$  of assigning one block to surgeon  $r$  in period  $t$ . Then, the integer program to construct a new surgery schedule (and at the same time price out a new workload pattern  $k$ ) is as follows:

$$\text{Minimize } \sum_{i \in I} \pi_i d_{ik} \quad (11)$$

subject to:

$$\sum_{t \in T} y_{rt} = q_r \quad \forall r \in R \quad (12)$$

$$\sum_{r \in R} y_{rt} \leq b_t \quad \forall t \in T \quad (13)$$

$$\sum_{r \in R} \sum_{t \in T} w_{rti} y_{rt} \leq d_{ik} \quad \forall i \in I \quad (14)$$

$$y_{rt} \in \{0, 1, 2, \dots, \min(q_r, b_t)\} \quad \forall r \in R, \forall t \in T \quad (15)$$

$$d_{ik} \in \{0, 1, 2, \dots\} \quad \forall i \in I \quad (16)$$

The objective function (11) minimizes the reduced cost of a new workload pattern. Observe that the periodic demands  $d_{ik}$  are now an integral part of the decision process, whereas these are merely coefficients in the master problem (5)-(9). Constraint set (12) implies that each surgeon obtains the number of required blocks.

Constraint set (13) ensures that the number of blocks assigned does not exceed the available number of blocks in each period. Constraint set (14) triggers the  $d_{ik}$ 's to the appropriate integer values. Finally, constraint set (15) and (16) define  $y_{rt}$  and  $d_{ik}$  to be integer.

At first sight, constraint set (16) which requires the periodic demands  $d_{ik}$  to be integral, seems to be redundant from a formulation point of view. Indeed, due to constraint (6) and the fact that  $a_{ij} \in \{0, 1\}$  and  $x_j \in \{0, 1, 2, \dots\}$  fractional demand values  $d_{ik}$  would also be covered by the upper integer number of nurses. The reason why we require the  $d_{ik}$ 's to be integral is to improve the computational efficiency of the overall branch-and-price algorithm. We come back to this issue in Section VII-A.

## V. OVERVIEW OF THE BRANCH-AND-PRICE ALGORITHM

Algorithm 3 contains the pseudocode of the branch-and-price algorithm to solve the GNSP.

The algorithm starts with a heuristic in order to find an initial solution. The heuristic generates only one workload pattern. This is done by building a surgery schedule for which the sum of the resulting quadratic demand values is minimized. The idea is to level the workload distribution as much as possible over the time horizon and as such to avoid the occurrence of peaks in the workload. This approach turned out to be beneficial for the surgery scheduling problem in which the expected shortage of beds has to be minimized (see [9]). The surgery schedule is built with a mixed integer program (MIP) in which the constraints are given by (12)-(15) (replacing the  $d_{ik}$ 's by  $d_i$ 's) and the objective is:

$$\text{Minimize } \sum_{i \in I} d_i^2$$

with  $d_i$  the required number of nurses in period  $i$ . To speed up the heuristic, the  $d_i$ 's are not required to be integral. Instead, we round each  $d_i$  to the next upper integer after solution of the quadratic MIP. Given this workload pattern, new roster lines are added until the set of roster lines (one nurse scheduled by each roster line) completely satisfies the coverage constraints. A new roster line is found by solving exactly the same shortest path problem as in Section IV-A, but replacing the dual prices  $\pi_i$  by the remaining right hand side values  $d_i$ . As such each new roster line cuts the peaks in the remaining workload pattern until all demand is covered.

After detection of an initial solution, the objective value is saved as an upper bound and both the surgery schedule and the nurse schedule are registered. The columns making up the initial solution are entered into

---

**Algorithm 3** BRANCH-AND-PRICE

---

```

apply heuristic to find initial solution;
if (solution found) then
    register nurse schedule and surgery schedule;
    upper_bound ← best solution found;
    initiate master with the columns making up the initial solution and ( $|I| + 1$ ) supercolumns;
else
    upper_bound ←  $+\infty$ ;
    initiate master with  $|I| + 1$  supercolumns;
end if
lower_bound ←  $-\infty$ ;
stop ← FALSE;
while (stop=FALSE) do
    LP_opt_found ← FALSE;
    {solve LP with column generation}
    while (LP_opt_found=FALSE) do
        LP_opt_found ← TRUE;
        improving_roster_line_found ← TRUE;
        while (improving_roster_line_found=TRUE) do
            RCj ← FIND-NEW-ROSTER-LINE( $j$ );
            if ( $RC_j < 0$ ) then
                add new roster line to master;
                LP_opt_found ← FALSE;
                LP_opt ← SOLVE-MASTER-LP();
            else
                improving_roster_line_found ← FALSE;
            end if
        end while
         $RC_k \leftarrow$  FIND-NEW-WORKLOAD-PATTERN( $k$ );
        if ( $RC_k < 0$ ) then
            add new workload pattern to master;
            LP_opt_found ← FALSE;
            LP_opt ← SOLVE-MASTER-LP();
        end if
    end while {LP solved to optimality}
    if (fractional_z) then
        expand node; {replace node by two child nodes}
    else if (LP_opt<best_integral_z) then
        best_integral_z ← LP_opt;
    end if
    if (no more nodes) then
        stop ← TRUE;
    else
        explore next node; {best-first}
        lower_bound ← bound.best_node;
        if (lower_bound  $\geq$  upper_bound OR lower_bound  $\geq$  best_integer_z) then
            stop ← TRUE;
        end if
    end if
    IP_opt ← SOLVE-MASTER-IP();
    if (IP_opt < upper_bound) then
        upper_bound ← IP_opt;
        register nurse schedule and surgery schedule;
    end if
end while

```

---

the master together with a number of supercolumns, which are needed to ensure feasibility of the master in each stage of the branch-and-bound algorithm.

The algorithm starts with the LP optimization loop in which iteratively a number of new roster lines and one new workload pattern are added until no more columns price out. Observe that roster lines are added until no more lines with negative reduced cost can be found, whereas only one workload pattern is generated, after which the generation of new roster lines restarts. This approach turned out to be the most successful, given the generally larger computation times to price out a new workload pattern.

Upon detection of the LP optimum, the solution is checked for fractional  $z_k$ 's (workload patterns). If there still are fractional  $z_k$ 's, branching is applied in order to drive the solution into an integral  $z$  solution (i.e. with only one  $z_k$  equal to 1 and all other equal to 0). The algorithm does not branch until an integral  $x_j$  (roster line) solution, because branching schemes for the  $x_j$  variables are not straightforward to implement and significantly complicate the roster line subproblem. Moreover, it provides no extra value for the extended model, which is the subject of this paper. Instead, we report lower and upper bounds for the required number of nurses to cover demand. The lower bound is the best possible solution with exactly one  $z_k$  equal to 1, however one for which the  $x_j$ 's are not necessarily integral. Hence, the solution represented by the lower bound might not be interpretable in terms of the nurse schedule (e.g. schedule 2.5 nurses following roster line  $j$ ). The upper bound on the other hand is the best found overall integer solution (with also integrality of the  $x_j$ 's), which is fully interpretable.

In order to increase the lower bound as much as possible, the branch-and-bound tree is traversed in a best-search way. After each move in the tree, the master problem is solved with required integrality on both the  $x_j$ 's and the  $z_k$ 's. Because the integral master problem is often computationally very intensive, the MIP optimizer is interrupted after a specified time interval (e.g. 10 seconds). If a better solution is found, the upper bound decreases and as such the gap between the lower and upper bound tightens.

## VI. BRANCHING

For reasons that are explained earlier, this work is only concerned with a branching scheme for driving the  $z_k$ 's to integrality and leaves the  $x_j$ 's out of consideration. We apply a constraint branching scheme [23] which works as follows.

First we search for the highest fractional  $z_k$ . Let this be  $z_{k'}$ . Then we select another  $z_k > 0$ , say  $z_{k''}$ , and take the first period  $i$  for which  $d_{ik'} \neq d_{ik''}$ . If no such period exists, both  $z_k$ 's represent essentially the same workload patterns and hence one of them can be set to 0 while its fractional value is added to the other one. Suppose we found period  $i'$  as the branching period with  $d_{i'k'} < d_{i'k''}$ . Then, we create two nodes in the branch-and-bound tree. In the left node we imply  $d_{i'k} \leq d_{i'k'}$  and in the right node we imply  $d_{i'k} \geq d_{i'k'} + 1$ . Figure 2 visualizes this branching scheme. Else if  $d_{i'k'} > d_{i'k''}$  we imply  $d_{i'k} \leq d_{i'k''}$  in the left node and  $d_{i'k} \geq d_{i'k''} + 1$  in the right node.

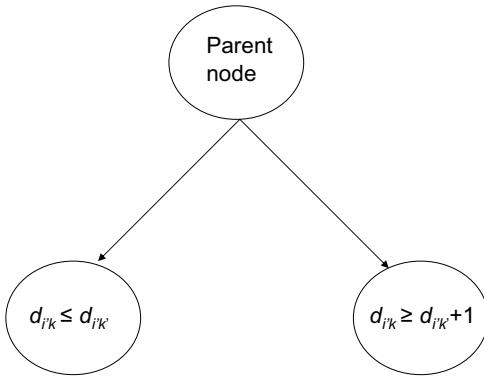


Fig. 2. Binary branching scheme in the case of  $d_{i'k'} < d_{i'k''}$

## VII. COMPUTATIONAL PERFORMANCE ISSUES

In this section we present some techniques which helped to improve the computational efficiency of the algorithm.

### A. Integral versus fractional demand values

It has already been mentioned at the end of Section IV-B that we imply the  $d_{ik}$ 's to be integral in the workload pattern pricing problem. Although this is not necessary from a formulation point of view, it has a substantially positive impact on the overall computational efficiency of the algorithm.

Implying integrality of the  $d_{ik}$ 's affects the computation time in two ways. On the one hand, there is a negative impact, because the pricing problem itself becomes more complex. On the other hand, there is a positive impact as far fewer columns can be found with negative reduced cost. Preliminary results indicate that this positive effect dramatically exceeds the negative effect. Consequently, the master LP is solved much faster when integrality of the  $d_{ik}$ 's is implied. Moreover, requiring integral demand values in the workload

patterns makes the LP optimal solution substantially less fractional in terms of the  $x_j$ 's. Hence, finding a global optimum (with both integrality on the  $z_k$ 's and on the  $x_j$ 's) turns out to be much easier. In our application the gap between the lower and upper bound becomes much smaller.

### B. Upper bound pruning for the workload pattern pricing problem

Basically, we are no longer interested in finding the column with the lowest reduced cost from the moment we know that this reduced cost will be positive anyway. Hence, we can act as if we already found a solution with reduced cost 0 by providing an appropriate upper bound. For the workload pattern subproblem, this observation yields dramatic time savings.

The reduced cost expression (4) consists of a fixed part and a variable part. By setting the upper bound equal to the fixed part with reverse sign, we act as if we found already a new column with reduced cost equal to 0. The reduced cost of a workload pattern is given by  $0 - \gamma + \sum_{i \in I} \pi_i d_{ik}$ . Consequently, we provide  $\gamma$  as an upper bound in the integer program (11)-(16).

Note that, since generating a new roster line is done using a backward dynamic recursion, upper bound pruning cannot be applied here. As an alternative, we wrote an A\* algorithm (enumeration approach entailing a forward recursion including both dynamic pruning and pruning based on bound comparisons). Dynamic pruning occurs if a state has already been visited at lower cost. For pruning based on bound comparisons we need an upper and lower bound for the best new roster line. Since the reduced cost of a new roster line is given by  $1 - \sum_{i \in I} a_{ij} \pi_i$ , we can provide -1 as an initial upper bound in the A\* algorithm. Obviously, this bound is decreased each time a better roster line is found. Starting from a certain day, a lower bound on the minimal cost path could be obtained by selecting for each remaining day the shift with the lowest total of corresponding dual prices, i.e:

$$\text{MIN} \left\{ \text{MIN}_{s \in S \setminus \{\text{"rest"}\}} \{ \lambda_{d,i} \}, 0 \right\} \quad \forall d$$

and summing up only the  $(f_{max} - f)$  lowest values amongst these. In other words, for calculating the lower bound, we relax all constraints but the not-more-than-one-shift-per-day constraint and the maximum number of active days constraint. Preliminary tests, however, indicated that the A\* algorithm is outperformed by the backward dynamic recursion. Hence, the time saved from upper bound pruning in the A\* algorithm is inferior to

the time won by visiting each state only once in the purely dynamic backward recursion.

### C. Two-phase approach for the workload pattern pricing problem

During the LP optimization loop it is not necessary to find the column with the most negative reduced cost, any column with negative reduced cost will do. Again, particularly for the computationally intensive workload pattern pricing problem, using this observation dramatically decreases the computation times. To guarantee optimality of the LP solution, a two-phase approach is applied for the workload pattern pricing problem. In the first phase, a certain time limit is set for the MIP optimizer. Only if no new workload pattern is found with negative reduced cost within this time limit, the algorithm enters the second phase. In this phase the time limit is undone and the optimizer is required to search until a feasible solution is found with negative reduced cost or it is proven that such a column does not exist.

### D. Lagrange dual pruning

It is well known that Lagrangian relaxation can complement column generation in that it can be used in every iteration of the column generation scheme to compute a lower bound to the original problem with little additional computational effort (see e.g. [25], [27]). If this lower bound exceeds an already found upper bound, the column generation phase can end without any risk of missing the optimum. Using the information from solving the reduced master and the information provided by solving the pricing problem for a new workload pattern  $k$ , it can be shown (see e.g. [16]) that a lower bound is given by  $\delta + RC_k \theta_k$  where  $\delta$  is the objective value of the reduced master,  $RC_k$  is the reduced cost of a newly found workload pattern  $k$  and  $\theta_k$  is a binary variable equal to 1 when  $RC_k$  is non-negative and set to zero, otherwise. This lower bound is referred to as the Lagrangian lower bound, since it can be shown that it equals the bound obtained by Lagrange relaxation.

Obviously, if the pricing procedure finds a negative reduced cost column during the first phase and hence does not enter the second phase (see Section VII-C) this lower bound cannot be used, because the workload pattern pricing problem has not been solved to optimality.

Using CPLEX, it is very easy to set upper bounds, time limits and limits on the number of feasible solutions. Moreover, it can easily be verified if either the problem has been solved to optimality or optimization has prematurely ended because of an insufficient time limit.

## VIII. COMPUTATIONAL RESULTS

### A. Test set

To test the algorithm, we started from the same set as the one introduced in [9] for their surgery scheduling application. All surgery scheduling problems in this set involve a cycle time of 7 days. The last two days are not available to allocate OR time (weekend), which is common practice. The problems differ with respect to five factors. These are: (1) the number of time blocks per day, (2) the number of surgeons, (3) the division of requested blocks per surgeon, (4) the number of operated patients per surgeon and finally (5) the length of stay (LOS) distribution. If we consider two settings for each factor and repeat each factor combination three times, we obtain  $2^5 * 3 = 96$  test instances. Table I contains the settings for these five factors. Some of the factor settings require some further explanation.

TABLE I  
FACTOR SETTINGS IN SURGERY SCHEDULING TEST SET

Factor setting	Nr. blocks per day	Nr. surgeons	Division req. blocks	Nr. patients per surgeon	LOS
1	3-6	3-7	evenly distributed	3-5	2-5
2	7-12	8-15	not evenly distributed	3-12	2-12

The number of blocks per day is drawn from a uniform distribution with bounds 3 and 6 in the first setting and 7 and 12 in the second setting. A block is defined as the smallest time unit for which a specific operating room can be allocated to a specific surgeon (or surgical group). Note that, due to large set-up time and costs, in real-life applications the number of blocks per day in one operating room is usually 1 or 2, i.e. each surgical group has the OR for at least half a day. Hence, considering more blocks can be seen as a way of considering more operating rooms as there is no difference from a computational point of view. The third factor indicates whether or not the requested blocks are evenly distributed among all surgeons; e.g. if there are 20 time blocks and 5 surgeons, each surgeon requires 4 time blocks in the evenly distributed case, whereas in the unevenly distributed case huge differences can occur. For the LOS in factor 5 we simulated exponential distributions (made discrete by use of binomial distributions) with mean dependent on the factor setting.

Next, we generated some weights  $w_{rti}$  defining the contributions to the workload of period  $i$  of allocating a block to surgeon  $r$  in period  $t$ . These weights

vary linearly with the number of patients of surgeon  $r$  operated in period  $t$  that are still in the hospital in period  $i$ . The patient's workload contribution generally decreases the longer the patient has already recovered in the hospital. In our test set the workload demand periods coincide with the shifts. Furthermore, we set the contribution to a "day" shift two times as large as the one to an "evening" shift and four times as large as the one to a "night" shift. Obviously, although attempting to represent realistic scenarios, these contributions are chosen somewhat arbitrarily.

Thirdly, we composed a set of collective agreement rules which apply on individual roster lines. The scheduling horizon amounted to 4 weeks or 28 days ( $= n$ ). The maximum days an active shift could be scheduled ("day", "evening" or "night") was set to 20 ( $= f_{max}$ ). Shifts during the weekends were marked as unpopular shifts: day and evening shifts got a penalty of 1, night shifts got a penalty of 2. The maximum number of consecutive working days was set to 6 ( $= h_1^{max} = h_{max}$ ) and the maximum number of consecutive rest days was set to 3 ( $= h_2^{max}$ ). Furthermore, we distinguished between two scenarios: a hard constrained scenario and a flexible one. Collective agreement rules in the hard constrained scenario differ from those in the flexible scenario on the following two points:

- In the hard constrained scenario, there is only one shift type allowed within each block. In other words, no shift transitions between different shift types can occur without scheduling a rest first. In the flexible scenario, all shift transitions are allowed, except the following three: a "night" shift followed by a "day" shift, a "night" shift followed by an "evening" shift or an "evening" shift followed by a "day" shift.
- In the hard constrained scenario, the maximal penalty with respect to unpopular shifts is set to 4, whereas in the flexible scenario it is set to 8 ( $= g_{max}$ ).

## B. Savings

Table II contains the lower and upper bounds for both the NSP and the GNSP. In the NSP, a surgery schedule is generated randomly. The resulting workload pattern contains the (fixed) right-hand side values of the coverage constraints. Then, the NSP is solved using column generation. In the GNSP, new surgery schedules (and hence resulting workload patterns) are generated during search if needed. We distinguish between the flexible and the hard constrained scenario. To give an idea of the variability, the detailed bounds are provided for the first 9 and the last 9 problems of the problem set.

The last line contains the average bounds over the whole set. Observe that the name of each problem ( $dijklm\_n$ ) contains the information about the surgery scheduling subproblem:  $i$  stands for the setting of the first factor in Table I (0 for the first setting, 1 for the second),  $j$  for the second one, etc..., and  $n$  for the iteration number.

From these results one may conclude the following. First have a look at the upper bounds, which are after all the solutions that will be worked with. Although it is not guaranteed that the upper bound will be better (one might be lucky in the NSP and find the same or even a better overall integer solution), the upper bounds for the GNSP are generally better than those for the NSP. We compared them using a one-tailed paired T-test. The extremely small p-values obtained indicate that the differences are statistically significant both for the flexible and for the hard constrained case. The same results are obtained for the lower bounds. Unlike the upper bounds, the GNSP lower bounds are of course guaranteed to be at least as good as the NSP lower bounds.

When comparing the lower bounds for the NSP with the upper bounds for the GNSP, both scenarios entail different conclusions. The average lower bound for the NSP is lower than the average upper bound for the GNSP in the flexible scenario, whereas the reverse is true in the hard constrained scenario. Both differences turned out to be significant using a one-tailed paired T-test (again extremely small p-values). This observation can easily be explained. The stricter the collective agreement rules, the harder it is to nicely fit the nurse rosters into the required workload pattern in the NSP. As the workload pattern can be adapted in the GNSP, the GNSP includes more possible savings in the case of severe collective agreement requirements.

## C. Interpretation of the savings

In the previous section we concluded that integrating the surgery scheduling process with the nurse scheduling process may yield important savings in terms of required nurses to hire. In this section we identify the source of these savings. Therefore, we provide an answer to the question: 'Where lies the waste if one is considering the surgery schedule (and hence the workload distribution) as being fixed?' It turns out that the origin of the waste is twofold.

First of all, an unfavorable workload pattern may contain many workload demands that slightly exceed the workforce of  $x$  nurses, but that are dramatically inferior to the workforce of  $x + 1$  nurses. In terms of the  $d_{ik}$ 's one could think of many  $d_{ik}$ 's having a small decimal part, like e.g. 6.1, 8.2, 4.05 etc... This type of waste is

TABLE II  
LOWER AND UPPER BOUNDS FOR THE NSP AND THE GNSP

Nr.	Problem	Flexible scenario				Hard constrained scenario			
		NSP		GNSP		NSP		GNSP	
		lb	ub	lb	ub	lb	ub	lb	ub
1	d00000_0	15	17	13	15	19	19	16	17
2	d00000_1	26	28	25	27	34	35	31	31
3	d00000_2	25	27	23	25	32	32	28	29
4	d00001_0	40	42	39	41	49	50	47	48
5	d00001_1	45	47	44	46	54	54	52	53
6	d00001_2	94	96	92	94	112	113	109	110
7	d00010_0	34	36	32	35	43	43	40	40
8	d00010_1	40	42	38	40	49	50	47	47
9	d00010_2	28	30	26	27	34	35	32	33
...	...	...	...	...	...	...	...	...	...
88	d11101_0	96	98	94	96	114	115	112	113
89	d11101_1	99	102	97	99	119	120	116	116
90	d11101_2	122	125	119	121	145	146	142	143
91	d11110_0	83	85	80	82	101	102	96	96
92	d11110_1	111	113	109	111	138	139	132	132
93	d11110_2	58	60	56	58	73	74	67	68
94	d11111_0	252	254	249	252	303	304	296	297
95	d11111_1	119	122	116	119	143	144	139	140
96	d11111_2	135	137	131	133	162	163	156	157
Average		70.18	72.43	68.33	70.44	86.07	86.73	81.91	82.61

referred to as the waste due to the workforce surplus per shift. In many hospitals this kind of waste is taken care of by simply scheduling  $x$  nurses instead of  $x+1$  nurses during those shifts. The result is a group of overworked nurses and an almost for sure decrease in the quality of care. This illustrates how the GNSP approach can also be very useful for optimizing qualitative instead of quantitative objectives.

Secondly, waste also originates from the inflexibility of the roster lines, due to strict general agreement requirements. Because of this, no set of roster lines can be found that perfectly fit with the workload demand. This source of waste is further referred to as waste due to the inflexibility of roster lines.

Table III gives an overview of the importance of both sources of waste. Hereby, we again distinguish between the flexible scenario and the hard constrained scenario. For each scenario there are three columns. The first column contains the total waste in terms of overstaffing in the NSP compared with the GNSP. These numbers are obtained by subtracting the upper bounds for the GNSP from those for the NSP. The second and third column indicate the parts of this total waste that are due to the workforce surplus per shift and to the inflexibility of roster lines. These numbers can easily be calculated as

follows. Firstly, for both the NSP and the GNSP we make the sum of the (integral) demands of the chosen workload pattern. Call this number the total required workforce ( $= \sum_{i \in I} d_i$  for the NSP and  $\sum_{i \in I} \sum_{k \in K} d_{ik} z_k$  for the GNSP). Next, divide this number by the workforce per nurse ( $= f_{max}$  in our application). This gives the minimal number of nurses that would be needed and can be obtained in the case of fully flexible roster lines. The difference between these numbers for the NSP and GNSP is the waste due to the workforce surplus per shift. The difference between the total waste and the waste due to the workforce surplus per shift is the waste due to the inflexibility of roster lines. Observe that these wastes may be negative (e.g. the waste due to workforce surplus per shift for problem d00000\_2 is -1). This situation occurs when the gain with respect to one source of waste is so large that the best found solution for the GNSP includes a limited sacrifice with respect to the other source of waste.

The results in Table III clearly indicate that the importance of the source of waste strongly depends on the strictness of the general agreement requirements. The stricter these requirements are, the larger is the share of the waste due to the inflexibility of the roster lines.

TABLE III  
INTERPRETATION OF THE SAVINGS

Nr.	Problem	Flexible scenario			Hard constrained scenario		
		Total waste	Waste due to workforce surplus per shift	Waste due to inflexibility of roster lines	Total waste	Waste due to workforce surplus per shift	Waste due to inflexibility of roster lines
1	d00000_0	2	1.2	0.8	2	1.2	0.8
2	d00000_1	1	1.2	-0.2	4	1.4	2.6
3	d00000_2	1	2	-1	3	1	2
4	d00001_0	1	1.2	-0.2	2	0.6	1.4
5	d00001_1	2	1	1	1	0.2	0.8
6	d00001_2	2	1.6	0.4	3	0	3
7	d00010_0	1	1.4	-0.4	3	1	2
8	d00010_1	1	1.6	-0.6	3	1.6	1.4
9	d00010_2	1	1.8	-0.8	2	-0.6	2.6
...	...	...	...	...	...	...	...
88	d11101_0	2	1.4	0.6	2	0.6	1.4
89	d11101_1	2	1.8	0.2	4	0.2	3.8
90	d11101_2	1	2.2	-1.2	3	0.2	2.8
91	d11110_0	2	1.6	0.4	6	0.8	5.2
92	d11110_1	2	0.8	1.2	7	0.6	6.4
93	d11110_2	1	2	-1	6	1.8	4.2
94	d11111_0	2	1.2	0.8	7	0.2	6.8
95	d11111_1	2	1.8	0.2	4	-0.6	4.6
96	d11111_2	1	2	-1	6	0.6	5.4
Average		1.58	1.43	0.16	4.11	0.28	3.84

#### D. Computational results

Table IV and Table V contain the computational results for the flexible respectively hard constrained scenario. For the NSP, both the computation time and the number of generated roster lines are given. For the GNSP also the number of generated demand patterns and the number of nodes in the branch-and-bound tree are provided.

Obviously, the required computation times for the GNSP exceed those for the NSP. However, taking into account the explosion of the feasible solution space for the GNSP compared to the NSP, the increase in computation time is rather small. We can conclude that column generation is an excellent technique for solving the GNSP.

If we compare the flexible scenario with the hard constrained scenario, a couple of things attract the attention. First of all, observe that for the NSP the computation times for the flexible scenario surpass those for the hard constrained scenario, whereas for the GNSP the computation times for the hard constrained scenario exceed those for the flexible scenario. For the NSP this difference is statistically significant (extremely small p-value for a two-tailed paired T-test) and easy to explain.

In the flexible scenario much more legal roster lines exist and hence much more roster lines with negative reduced cost are found during the search process (on average 207.25 versus 106.07). Moreover, the time needed to price out a new roster line is also larger since the feasible state space contains more legal states.

For the GNSP the difference in computation time is not statistically significant at the 5% level (p-value of 0.113 for a two-tailed paired T-test). As again the number of generated roster lines is significantly smaller (very small p-value for a two-tailed paired T-test), the higher computation times for the constrained scenario must be produced by the higher number of generated workload patterns and the higher number of nodes in the branch-and-bound tree. The differences in number of generated workload patterns and in nodes in the branch-and-bound tree are found to be significant (very small p-values for two-tailed paired T-tests). This can easily be explained as follows. In the flexible scenario, it is unlikely that an extra workload pattern improves the overall solution. Thanks to the flexibility in the roster lines, an already very good solution can be found using a limited set of workload patterns. In the hard constrained case on the other hand, the inflexibility of the roster lines might

obstruct the detection of a good solution. In this case, it is far more likely that adding a new workload pattern improves the overall solution. We can conclude that the GNSP is easier to solve if the collective agreement requirements are less strict, whereas the reverse is true for the NSP.

As a final remark we note that a large part of the computation time goes to the calculation of an overall feasible solution in order to detect an upper bound after each move in the branch-and-bound tree in the GNSP and at the end of the column generation process in the NSP.

TABLE IV  
COMPUTATIONAL RESULTS FOR THE FLEXIBLE SCENARIO

Nr.	Problem	Time (s)	NSP		GNSP		
			Roster lines	Time (s)	Roster lines	Workload patterns	Nodes
1	d00000_0	43484	150	44422	183	2	0
2	d00000_1	44063	174	51000	196	2	0
3	d00000_2	46423	235	45438	213	2	0
4	d00001_0	44078	173	46000	221	2	0
5	d00001_1	43829	167	45172	190	2	0
6	d00001_2	44844	212	48829	238	3	0
7	d00010_0	45266	211	70359	274	2	0
8	d00010_1	46311	237	185623	535	17	8
9	d00010_2	44594	208	166892	640	32	13
...	...	...	...	...	...	...	...
88	d11101_0	44390	213	47984	243	2	0
89	d11101_1	44953	228	52031	257	2	0
90	d11101_2	44734	230	56438	280	2	0
91	d11110_0	46203	252	358811	555	30	15
92	d11110_1	45265	238	1765257	815	128	59
93	d11110_2	47359	200	423125	507	28	14
94	d11111_0	46360	347	69266	381	2	0
95	d11111_1	45719	243	59063	319	2	0
96	d11111_2	45048	237	251970	512	14	6
Average		44146.04	207.25	99008.57	310.31	5.93	1.95

TABLE V  
COMPUTATIONAL RESULTS FOR THE HARD CONSTRAINED SCENARIO

Nr.	Problem	Time (s)	NSP		GNSP		
			Roster lines	Time (s)	Roster lines	Workload patterns	Nodes
1	d00000_0	453	46	66953	263	8	4
2	d00000_1	500	70	55359	304	18	6
3	d00000_2	422	64	11781	111	2	0
4	d00001_0	468	77	609	81	2	0
5	d00001_1	453	74	687	95	3	0
6	d00001_2	672	120	782	127	2	0
7	d00010_0	4250	113	216064	470	79	43
8	d00010_1	953	113	323236	448	129	47
9	d00010_2	750	80	201970	459	102	39
...	...	...	...	...	...	...	...
88	d11101_0	2125	122	1656	130	2	0
89	d11101_1	1531	126	2625	146	2	0
90	d11101_2	1610	149	2109	159	2	0
91	d11110_0	1938	123	456191	439	58	17
92	d11110_1	1500	152	1228851	508	92	45
93	d11110_2	5438	101	102470	310	10	1
94	d11111_0	8000	251	12265	264	2	0
95	d11111_1	4859	143	19359	185	2	0
96	d11111_2	4922	153	1809557	600	221	83
Average		1215.52	106.07	153927.85	226.05	28.08	10.81

## IX. CONCLUSIONS AND FURTHER RESEARCH

This paper presents an integrated approach for building nurse and surgery schedules. It has been shown how the column generation technique, often employed for solving nurse scheduling problems, can easily be extended to cope with this integrated approach. The approach involves the solution of two types of pricing problems, the first one is solved with a standard dynamic programming recursion, the second one by aims of a state-of-the-art mixed integer programming optimizer. A constraint branching scheme has been proposed to drive the solution into integrality with respect to the workload patterns while the integrality of the roster lines was left out of the scope of this paper. Finally, some techniques were presented that helped to improve the computational efficiency of the branch-and-price algorithm.

Our computational results indicate that considerable savings could be achieved by using this approach to build nurse and surgery schedules. We simulated problems for a large range of surgery scheduling instances and distinguished between a flexible and a hard constrained scenario with respect to the collective agreement requirements. Our conclusions can be summarized as follows. First of all, column generation is a good technique to

deal with the extra problem dimension of modifying surgery schedules. Secondly, the obtained gains originate from two sources of waste: waste due to the workforce surplus per shift and waste due to the inflexibility of roster lines. Thirdly, unlike the NSP, the GNSP turns out to become harder to solve when the collective agreement requirements are more strict.

Obviously, in real-life hospital environments it is not so easy to modify the master surgery schedule. As the surgery schedule can be considered to be the main engine of the hospital, it not only has an impact on the workload distribution for nurses, but also on several other resources throughout the hospital. Think for instance about anaesthetists, equipment, radiology, laboratory tests and consultation. This observation yields a negative as well as a positive note for the reasoning in this paper. The negative note is that the possible savings obtained through integrating the nurse and the surgery scheduling process are in real-life probably much smaller, due to the smaller flexibility with which surgery schedules can be modified. The positive note is that not only savings in nurse staffing costs are possible, but also in other related resource types, by integrating the scheduling of these resources with the surgery scheduling process. This is probably the main contribution of this paper. This work clearly shows the benefits of integrating scheduling processes in health care environments and moreover proposes a methodology for implementing the heart of a supporting ICT infrastructure.

Possible topics for further research include the application of this approach in a real-world environment involving a detailed report on the experienced merits and pitfalls. From a theoretical point of view, it would be interesting to elaborate this technique for one or more of the other resource types stated above.

#### ACKNOWLEDGEMENTS

We acknowledge the support given to this project by the Fonds voor Wetenschappelijk Onderzoek (FWO) - Vlaanderen, Belgium under contract number G.0463.04.

#### REFERENCES

- [1] H. Alfares and J. Bailey, "Integrated project task and manpower scheduling," *IIE Transactions*, vol. 29, pp. 711–718, 1997.
- [2] H. Alfares, J. Bailey, and W. Lin, "Integrating project operations and personnel scheduling with multiple labor classes," *Production Planning & Control*, vol. 10, pp. 570–578, 1999.
- [3] M. N. Azaiez and S. S. Al Sharif, "A 0-1 goal programming model for nurse scheduling," *Computers and Operations Research*, vol. 32, pp. 491–507, 2005.
- [4] J. F. Bard, C. Binici, and A. H. deSilva, "Staff scheduling at the United States Postal Service," *Computers and Operations Research*, vol. 30, pp. 745–771, 2003.
- [5] J. F. Bard and H. W. Purnomo, "A column generation-based approach to solve the preference scheduling problem for nurses with downgrading," *Socio-Economic Planning Sciences*, vol. 39, pp. 193–213, 2005.
- [6] ———, "Preference scheduling for nurses using column generation," *European Journal of Operational Research*, vol. 164, pp. 510–534, 2005.
- [7] 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.
- [8] N. Beaumont, "Scheduling staff using mixed integer programming," *European Journal of Operational Research*, vol. 98, pp. 473–484, 1997.
- [9] J. Beliën and E. Demeulemeester, "Integer programming for building robust surgery schedules," Katholieke Universiteit Leuven, Department of Applied Economics, Research Report OR 0446, 2004.
- [10] J. T. Blake and M. W. Carter, "A goal programming approach to strategic resource allocation in acute care hospitals," *European Journal of Operational Research*, vol. 140, pp. 541–561, 2002.
- [11] J. T. Blake, F. Dexter, and J. Donald, "Operating room manager's use of integer programming for assigning block time to surgical groups: A case study," *Anesthesia and Analgesia*, vol. 94, pp. 143–148, 2002.
- [12] A. Caprara, M. Monaci, and P. Toth, "Models and algorithms for a staff scheduling problem," *Mathematical Programming*, vol. 98, pp. 445–476, 2003.
- [13] B. Cheang, H. Li, A. Lim, and B. Rodrigues, "Nurse rostering problems - A bibliographic survey," *European Journal of Operational Research*, vol. 151, pp. 447–460, 2003.
- [14] F. Dexter and A. Macario, "Changing allocations of operating room time from a system based on historical utilization to one where the aim is to schedule as many surgical cases as possible," *Anesthesia and Analgesia*, vol. 94, pp. 1272–1279, 2002.
- [15] A. Guinet and S. Chaabane, "Operating theatre planning," *Int. J. Production Economics*, vol. 85, pp. 69–81, 2003.
- [16] E. W. Hans, "Resource loading by branch-and-price techniques," Ph.D. Dissertation, Twente University Press, Enschede, The Netherlands, 2001.
- [17] W. L. Hughes and S. Y. Soliman, "Short-term case mix management with linear programming," *Hospital and Health Services Administration*, vol. 30, pp. 52–60, 1985.
- [18] B. Jaumard, F. Semet, and T. Vovor, "A generalized linear programming model for nurse scheduling," *European Journal of Operational Research*, vol. 107, pp. 1–18, 1998.
- [19] E. Litvak and M. C. Long, "Cost and quality under managed care: Irreconcilable differences?" *The American Journal of Managed Care*, vol. 6, pp. 305–312, 2000.
- [20] A. J. Mason and M. C. Smith, "A nested column generator for solving rostering problems with integer programming," in *International Conference on Optimisation: Techniques and Applications*, 1998, pp. 827–834.
- [21] A. Mehrotra, K. E. Murphy, and M. A. Trick, "Optimal shift scheduling: A branch-and-price approach," *Naval Research Logistics*, vol. 47, pp. 185–200, 2000.
- [22] H. E. Miller, W. P.ierskalla, and G. J. Rath, "Nurse scheduling using mathematical programming," *Operations Research*, vol. 24, pp. 857–870, 1976.
- [23] D. M. Ryan and B. A. Foster, "An integer programming approach to scheduling," in *Computer Scheduling of Public Transport Urban Passenger Vehicle and Crew Scheduling*, A. Weren, Ed. North-Holland, Amsterdam, 1981, pp. 269–280.
- [24] USDHHS, *Projected supply, demand and shortages of registered nurses: 2000-2020*. National Center for Health Work-

- force Analysis. US Department of health and Human Services, Rockville, MD, 2002.
- [25] M. Van den Akker, H. Hoogeveen, and S. L. van de Velde, "Combining column generation and lagrangian relaxation to solve a single-machine common due date problem," *INFORMS Journal on Computing*, vol. 14, pp. 37–51, 2002.
  - [26] F. Vanderbeck, "On Dantzig-Wolfe decomposition in integer programming and ways to perform branching in a branch-and-price algorithm," *Operations Research*, vol. 48, pp. 111–128, 2000.
  - [27] F. Vanderbeck and L. A. Wolsey, "An exact algorithm for IP column generation," *Operations Research Letters*, vol. 19, pp. 151–159, 1996.
  - [28] D. M. Warner, "Scheduling nursing personnel according to nursing preferences: A mathematical programming approach," *Operations Research*, vol. 24, pp. 842–856, 1976.
  - [29] E. N. Weiss, "Models for determining estimated start times and case orderings in hospital operating rooms," *IIE Transactions*, vol. 22, pp. 143–150, 1990.

# A bi-objective coordination setup problem in a two-stage production system

Michele Ciavotta\*, Paolo Detti†, Carlo Meloni‡ and Marco Pranzo†

\*D.I.A. Università Roma Tre

Via Della Vasca Navale, 79, I-00146 Roma Italy

Email: ciavotta@dia.uniroma3.it

†D.I.I. Università di Siena

Via Roma, 56, I-53100 Siena Italy

Email: {detti, pranzo}@dii.unisi.it

‡D.E.E. Politecnico di Bari

Via E. Orabona, 4, I-70125 Bari Italy

Email: meloni@deemail.poliba.it

**Abstract**—This paper addresses a problem arising in the coordination between two consecutive departments of a production system, where parts are processed in batches, and each batch is characterized by two distinct attributes. Due to limited interstage buffering between the two stages, these departments have to follow the same batch sequence. In the first department, a setup occurs every time the first attribute of a new batch is different from the previous one. In the downstream department, there is a setup when the second characteristic changes. The problem consists in finding a set, as large as possible, of batch sequences optimizing the number of setups paid by each department. This case results in a particular bi-objective combinatorial optimization problem. Here we present a geometrical characterization for the solution space of the problem and we propose an experimental study of three different metaheuristics. The proposed approaches result in fast, accurate and effective algorithms which are able to find almost the whole Pareto front with an acceptable computation time.

**Keywords**—Setups, Multi-Objective Algorithms, Sequencing, Manufacturing systems.

## I. INTRODUCTION

THIS work deals with a problem of real-life manufacturing interest: the coordination of two consecutive production departments. Each department consists of a flexible machine, the first one deals in shaping items of raw wooden panels; the second concerns the painting of the just shaped items. To avoid unnecessary costs, each department works with batches of jobs, in which every job has the same shape and color. Since it is not possible for a job to change its own route in the production chain, as well as for a department to reschedule the sequence

of incoming batches, this problem belongs to the class of Permutation Scheduling.

When two consecutive jobs with different features have to be processed, at least one department must pay a setup. Each department tends to organize its own operations in order to minimize its setup number. From a global point of view, this problem is inherently multi-objective ([8], [14]) because each department has to take into account the requirements of the other one. At this aim, a set of optimization algorithms are proposed in this paper.

A graph model introduced in [1] is used to obtain a lower and upper bound setup number for each department, and a good estimation of all possible Pareto front points. At an operational level the goal is to find a, possibly large, set of sequences of batches solving a trade-off between different objectives. We tackled this multi-criteria problem using a metaheuristic approach. The goal of this algorithm is to obtain a good estimation of the Pareto front for the multi-objective problem. In the proposed approach first the total setup number of the production system is minimized, then a different procedure is employed to spread the setups over departments keeping constant the total number of setups. Different procedures to guide the algorithms toward the discovering of a greater part of the Pareto front are implemented. The proposed approach returns a set of non-dominated points achieving a high probability of covering almost the whole Pareto front in an acceptable computation time.

The paper is organized as follows. In Section II, literature results and applications are discussed. In Section III the industrial context is described, a formal description of the problem and a geometrical characterization

of the solution space are given. In Section IV, three metaheuristic algorithms are presented, and in Section V a large sample of experiments are reported, showing the effectiveness of the proposed approaches. Finally, in Section VI conclusions are drawn.

## II. LITERATURE AND APPLICATIONS

Minimizing the impact of changeovers has been widely described as a main component of modern production management strategies [16]. Pursuing high changeover performances is a way to enable agile and responsive manufacturing processes by improving line productivity and reducing downtime losses [10]. This aspect of the production management involving both organizational and economic aspects has received an increasingly attention also in fields as applied mathematics and operations research.

In particular, over the past few decades, there has been a significant effort associated with reducing the time required to perform setups and developing suitable changeover modeling processes. This process can be quite complicated, but yields important benefits in planning and scheduling a production system [16] improving both the production capacity and the system manageability. Important surveys on changeovers and setups are proposed in [3], [11] where classifications of setups are also given. A wide discussion on MIP models is given in [17]; while [12] offers a review of heuristics for setup problems in serial systems. The models and algorithms presented in this paper particularly refer to the furniture production case first addressed in [1]. Despite the problem is quite specific, it may arise in different production contexts in which one is interested in minimize the setups. Typically, in such cases, setup costs are a prominent part of the overall production costs. In successive works, some strongly related problems have been considered, by the same authors, under both theoretic and algorithmic point of view [4], [5]. In particular, these works deal with graph problems such as the Hamiltonian Completion Number (*HCN*) problem and the Minimum Cardinality Dominating Trail Set (*MCTS*) problem. The literature also presents other different real world applications of some variants of the problem addressed in this paper. As examples, we cite the optimization of the line scheduling for production of components for catalytic converters considered in [13], the setup sequencing problems arising in the weaving industry [?], and the part loading scheduling in a forging machine addressed in [15].

## III. PROBLEM DESCRIPTION

### A. The industrial context

This paper addresses a problem arising in the coordination between two consecutive production departments of an industrial system. Often in multi-stage serial manufacturing systems there not exists an interstage mechanism for re-scheduling the items. Hence a very important task becomes to select an appropriated sequence of jobs. Such sequence has to take into account the requirements of every stage. In particular, we consider the problem of sequencing production batches of jobs in two consecutive manufacturing departments. Due to the lack of interstage buffering, departments must process the batches in the same order. Batches to produce are characterized by two different attributes, say  $A_1$  and  $A_2$ , the actual cardinality of each batch (i.e., the number of parts that compose each batch) is of no interest at all in this kind of problems. In the first (the second) department, a setup occurs when the attribute  $A_1$  (attribute  $A_2$ ) of the next batch to be processed changes. Therefore this kind of setups can be classified as sequence dependent [3], [11].

Each given sequence of batches results in a number of setups to be paid by each department, and the problem is finding a collection of batch sequences minimizing the costs related to changeovers (i.e., setups). In this way it is possible to offer to the decision-maker a wide range of choices. Moreover it may be useful to re-schedule the production work on line holding unchanged the quality of the schedule; this could be obtained by selecting (where it is possible) another sequence of the found solution set. Since, in the context under study, the activities involved in a changeover are very similar, regardless of the particular changeover, the total number of setups is, in this case, a meaningful index of performance. We refer to the real industrial context addressed in [1], in which a large number of different slabs of wood are cut, painted and assembled to build kitchen furniture. In this system, two consecutive departments are considered and no resequencing is possible between the two stages. The two departments are the cutting and the painting departments and batches are characterized only by shapes and a colors. In the cutting department, a setup occurs when a batch has a different shape from the preceding one (cutting tools and machinery must be reconfigured). Similarly, in the painting station a setup occurs when a new color is used (the equipment and the pallets must be thoroughly cleaned in order to eliminate the residuals of the previous color). Hence, in both cases costs are paid in terms of time and manpower. In particular, we consider the case in which all the setups have the same cost. Hence, we focus our attention on

the number of setups. Since each item to be produced has its own shape and color, all the items having the same shape and color form a single batch. In fact, there is no convenience, on either side, in splitting such batches, while the actual cardinality of each batch is of no interest for us here. Hence, in processing two consecutive batches, at least a department has to pay a changeover. Minimizing the changeover cost for a department often implies the increasing of the setup costs for the other one. It leads that this problem is inherently multi-objective. More in details, we analyze a bi-criteria version of the described problem, where the two objectives we consider are the minimization of the number of the setups paid by each department. This bi-objective problem is NP-hard [1] (even if the single-objective problem requiring to minimize the setup cost of a single department is clearly an easy task) and it calls for a heuristic solution approach. Hence, our aim is to find in a reasonable computation time a good approximation of the Pareto-optimal front. Thus, in general, a solution of the problem is a set of trade-off solutions, i.e., non dominated solutions. In this paper we propose three metaheuristic approaches for the problem which are able to attain a good approximation of the Pareto front in reasonable amount of time even for large instances.

### B. Notation and problem formulation

The problem we consider in this paper can be more formally formulated as follows. Let  $A$  be a set of batches to be produced. The batches must be processed by two departments of the plant, called  $D_S$  and  $D_C$ , in the same order. Each batch is characterized by two attributes, say *shape* and *color*. Let  $S$  and  $C$  denote the sets of all possible shapes and colors respectively. We denote the shapes as  $s_i$ ,  $i = 1, \dots, |S|$  and the colors as  $c_j$ ,  $j = 1, \dots, |C|$ . Each batch is therefore defined by a pair  $(s_i, c_j)$ . If batch  $(s_i, c_j)$  is processed immediately after batch  $(s_h, c_k)$ , a setup is paid in department  $D_S$  if  $s_h \neq s_i$ , and a setup is paid in department  $D_C$  if  $c_k \neq c_j$ . We can represent the input as a bipartite graph,  $G = (S, C, A)$ , where nodes in  $S$  correspond to shapes, nodes in  $C$  to colors and each edge of  $A$  corresponds to a batch to be produced. The problem is to sequence the batches in a profitable way from the viewpoint of the number of setups. This means that we must find some particular ordering  $\sigma$  of the edges of  $G$ . If two consecutive edges  $(i, j)$  and  $(h, k)$  in  $\sigma$  have no nodes in common, this means that both departments have to pay one setup when switching from batch  $(i, j)$  to batch  $(h, k)$ . We refer to this as a *global changeover*. On the

other hand, if  $i = h$  ( $j = k$ ), only department  $D_C$  ( $D_S$ ) pays a setup. This is called *local changeover*. For a given sequence  $\sigma$ , we can therefore easily compute the number of setups incurred by each department, call them  $N_S(\sigma)$  and  $N_C(\sigma)$  respectively. In fact, let  $\delta_{ih}$  be equal to 1 if  $i \neq h$  and 0 otherwise, and let  $s(\sigma(q))$  denote the shape of the  $q$ -th batch in the sequence  $\sigma$ , and let  $c(\sigma(q))$  denote its color. Hence,

$$N_S(\sigma) = 1 + \sum_{q=1}^{|A|-1} \delta_{s(\sigma(q)), s(\sigma(q+1))} \quad (1)$$

$$N_C(\sigma) = 1 + \sum_{q=1}^{|A|-1} \delta_{c(\sigma(q)), c(\sigma(q+1))} \quad (2)$$

The two objectives of the problem addressed in this paper are exactly  $N_S(\sigma)$  and  $N_C(\sigma)$  (for sake of simplicity, we use  $N_S$  and  $N_C$  when the context does not require to remark the dependence of  $\sigma$ ). The problem we address, denoted as  $N_S$ - $N_C$  problem, consists in finding a set of batch sequences minimizing the two objectives  $N_S$  and  $N_C$ .

### C. Geometrical properties of the solution space of $N_S$ - $N_C$ problem

In this section, a geometrical characterization of the solution space of  $N_S$ - $N_C$  problem is presented. Such characterization is useful to determine a good estimation of the Pareto front.

In the literature, a heuristic approach for another bi-criteria optimization problem with the same combinatorial structure but with a different pair of objective functions has been presented in [6]. In particular, in [6] the minimization of following criteria is considered:  
 $SUM = N_S + N_C$  and  
 $MAX = \max\{N_S, N_C\}$ .

In the following, we refer to this bi-objective problem as *SUM-MAX* problem. The above criteria strongly depend on those (i.e.,  $N_S$  and  $N_C$ ) considered in the problem under study in this paper. Some observations on the solution space of the *SUM-MAX* problem give important information about the solution space of the  $N_S$ - $N_C$  problem. In the computational experiments presented in [6] and carried on a wide set of instances, the effectiveness of the proposed heuristic is shown. Moreover, the computational experiments highlight that the Pareto front always contains few (often one) non dominated points. This is due to the strong correlation between the objectives *SUM* and *MAX*. Note that, in general, to the same point in the criteria space may correspond several batch sequences  $\sigma$ . In the following,

we give a geometric representation of the solution space of *SUM-MAX* problem and  $N_S-N_C$  problem.

In [5], [6], the authors proposed a pair of useful lower (upper) bounds  $LB_{SUM}$  ( $UB_{SUM}$ ) and  $LB_{MAX}$  ( $UB_{MAX}$ ) for the *SUM* and *MAX* objective, respectively. The lower bounds define an ideal point  $IP = (LB_{SUM}, LB_{MAX})$  for *SUM-MAX* problem, whereas the upper bounds are used to define a nadir point  $(UB_{SUM}, UB_{MAX})$ .

The solution space of the *SUM-MAX* problem has the following geometrical properties. Given a batch sequence  $\sigma_p$ , let  $p = (SUM_p, MAX_p)$  be a point in the solution space of the *SUM-MAX* problem, where  $SUM_p = N_S(\sigma_p) + N_C(\sigma_p)$  and  $MAX_p = \max\{N_S(\sigma_p), N_C(\sigma_p)\}$ . Let  $p$  be a point in the solution space of the *SUM-MAX* problem, and let  $SUM_p$  and  $MAX_p$  be the coordinates of  $p$  in the solution space *SUM-MAX*. It can be noted that the following relations hold:

$$SUM_p \leq 2 * MAX_p \quad (3)$$

$$MAX_p + \eta \leq SUM_p \quad (4)$$

where  $\eta = \min_{\sigma} \{N_S(\sigma), N_C(\sigma)\}$  is a constant that can be easily computed.

Obviously, inequality (3) holds for all points  $p$  representing a feasible solution for the *SUM-MAX* problem. In fact, for each solution point, the correspondent value of the objective function *SUM* is less than twice of the value assumed by the second objective *MAX*. Note that Relation (4) holds for all the feasible points of the *SUM-MAX* problem, too. Hence, inequalities (3), (4) and the lower and upper bounds previously introduced describes the whole solution space of the problem *SUM-MAX* (see Figure 1). It is important to note that while coordinates  $SUM_p$  and  $MAX_p$  depend on the particular  $\sigma_p$  associated to a point  $p$ ,  $\eta$  is a constant which depends only on the instance of the problem.

In the Figure 1,  $IP$  is the ideal point determined by  $LB_{SUM}$  and  $LB_{MAX}$ ; while the point  $A$  is the unfeasible solution point individuated by the intersection of the two lines (i.e., 3 and 4) bounding the solution space.

These observations give us some useful information about the solution space of the  $N_S-N_C$  problem, object of this paper.

Each batch sequence  $\sigma_p$  has a representative point  $p = (SUM_p, MAX_p)$  in the solution space of *SUM-MAX* problem, while in the solution space of  $N_S-N_C$  problem, the same sequence has an image given by the point  $\pi = (N_S(\sigma_p), N_C(\sigma_p))$ , where:

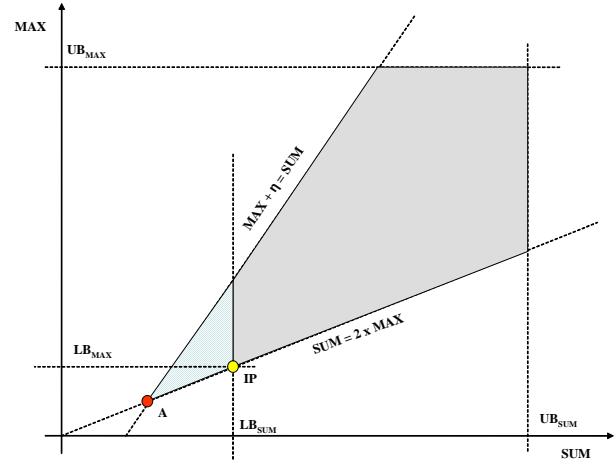


Fig. 1. Geometric aspects of the feasible region of the *SUM-MAX* problem.

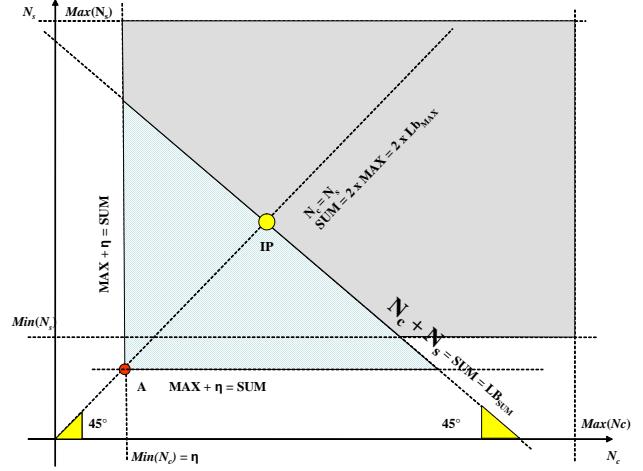


Fig. 2. Geometric aspects of the solution space of the  $N_S-N_C$  problem.

$$SUM_p = N_S(\sigma_p) + N_C(\sigma_p), \text{ and} \\ MAX_p = \max\{N_S(\sigma_p), N_C(\sigma_p)\}.$$

Thus, several characteristics of one space can easily be mapped in the other one.

In Figure 2, the solution space of  $N_S-N_C$  problem is depicted and, in Figure 3, the ideal Pareto front of the problem is shown.

It can be observed that  $LB_{SUM}$  is quite useful also in the space of the  $N_S-N_C$  problem giving an estimation of the whole Pareto front. In fact, in this space the relation  $SUM \geq LB_{SUM}$  can be re-reviewed as  $N_S + N_C \geq LB_{SUM}$  and hence it gives a lower bound for the Pareto front of the problem. Note that in Figure 2 the case with  $\eta = \min_{\sigma} \{N_C(\sigma)\} = \min_{\sigma} \{N_S(\sigma), N_C(\sigma)\}$  is depicted.

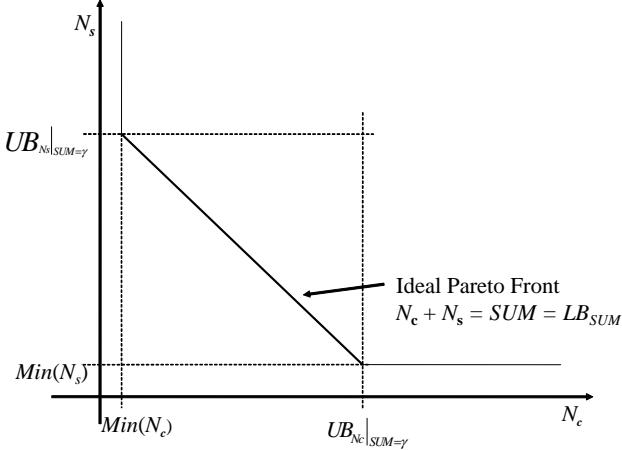


Fig. 3. Bounds for the Pareto front for the  $N_S$ - $N_C$  problem.

The estimation of the Pareto front can be easily detailed determining an estimation of the number and the position of elements (i.e., integer solutions) contained in it.

More information about the solution space of the  $N_S$ - $N_C$  problem can be obtained mapping other geometric objects from the *SUM-MAX* solution space as shown in Figures 2 and 3.

Since the minimum setup costs for a department (as single objective) can be easily attained by ordering the batches in the sequence by grouping them by the same color or shape, we can calculate, in closed form, the minimum values  $\min_\sigma\{N_S(\sigma)\}$  and  $\min_\sigma\{N_C(\sigma)\}$  for a given instance of the problem. Given an estimated Pareto front  $N_S + N_C = SUM = \gamma$  for an instance of  $N_S$ - $N_C$  problem, upper bounds  $UB_{N_S}|_{SUM=\gamma}$  and  $UB_{N_C}|_{SUM=\gamma}$  on the values of  $N_S$  and  $N_C$ , respectively, when  $SUM$  is fixed to the value  $\gamma$  are:

$$\begin{aligned} UB_{N_S}|_{SUM=\gamma} &= \gamma - \min\{N_C\}; \\ UB_{N_C}|_{SUM=\gamma} &= \gamma - \min\{N_S\}. \end{aligned}$$

The above values delimit the estimated Pareto front.

Moreover, using the information given by these geometric observations and the values of the bounds previously introduced, it is possible to calculate, in closed form, the cardinality  $|PF_\gamma|$  of the set of all points lying on the estimated Pareto front  $PF_\gamma$  described by

$$N_S + N_C = SUM = \gamma.$$

In fact, due to the geometric properties of the bounds on the solution space, the number of points contained in the estimated Pareto front is:

$$|PF_\gamma| = UB_{N_C}|_{SUM=\gamma} - \min\{N_C\} + 1, \quad (5)$$

or, in an equivalent way,

$$|PF_\gamma| = UB_{N_S}|_{SUM=\gamma} - \min\{N_S\} + 1. \quad (6)$$

and the position of each point belonging to the Pareto front can be easily computed.

We use this geometric characterization in the design and development of the proposed algorithms. In fact, the estimated number and position of the ideal solutions is used to guide the algorithms' behavior toward the Pareto front.

#### IV. ALGORITHMS

In this section we describe three metaheuristic approaches developed to tackle the problem. A metaheuristic is an iterative solution procedure, combining subordinate heuristic tools into a more sophisticated framework [7], [9]. All the developed algorithms are based on the same basic subordinate procedures.

- **Constructive Heuristic:** This heuristic is a greedy procedure. It starts from the empty sequence  $\sigma = \emptyset$  and at each step it adds to the partial sequence  $\sigma$  all the batches sharing a given attribute. The behavior of this routine is controlled by an input parameter in order to produce promising starting points for the improving procedures.
- **Improving procedures:** As solution improving tools we adopted an advanced local search procedure able to attain solutions belonging to the Pareto front or close to it. Besides the local search we developed also an exploration procedure that, given a solution, it generates a set of neighbors adjacent to it in the solution space having the same quality.
- **Update routine:** This helper function updates the Pareto front adding, if necessary, the new Pareto solutions found by the improving procedures.

These subordinate procedures are used by some *Master procedure* to guide the search process toward the exploration of the Pareto front. In the following subsections, first we describe in details the tools that constitute our framework, and next we introduce two Master procedures employing the described subordinate tools, and finally we illustrate the metaheuristic algorithms developed.

##### A. Constructive heuristic

The constructive heuristic is an iterative greedy procedure used to generate starting solutions. The heuristic starts from the empty sequence  $\sigma = \emptyset$  and at each step it selects one attribute from a department. Then it adds to the partial sequence  $\sigma$  all the batches sharing the selected

attribute. This process is repeated until all the batches are sequenced.

More in details, the behavior of the heuristic is guided by an input parameter ( $\rho$ ) determining which department is selected at each iteration. In particular, if  $\rho > 0$  ( $\rho < 0$ ) then attributes from department  $D_S$  (department  $D_C$ ) are chosen in the next  $\rho$  iterations. After these  $\rho$  iterations one attribute from department  $D_C$  (department  $D_S$ ) is selected. Given a selected department the attribute with the smaller number of occurrences among the unsequenced batches is selected. In terms of the bipartite graph  $G = (S, C, A)$  the chosen attribute corresponds to the node having smaller degree. The node degree is updated every time a batch is added to the sequence  $\sigma$ .

Observe that the solution generated by the constructive heuristic may have a poor quality. A global changeover is paid at every iteration of the algorithm because the sequence  $\sigma$  switches from two batches with both departments operating a setup.

### B. Solution improving procedures

As improving procedures we consider two different procedures minimizing the single objective problems, SUM and MAX independently. The two single objective problems minimize the overall number of setups and the balanceness of the setup among the two departments, respectively.

Observe that, minimizing the SUM objective, i.e., minimizing  $N_S + N_C$ , corresponds to a move toward the Pareto front. In fact, given a solution  $p = (SUM_p, MAX_p)$ , the reduction of the SUM objective by one generates a new solution  $p'$  which is either  $p' = (SUM_p - 1, MAX_p)$  or  $p' = (SUM_p, MAX_p - 1)$ . On the other hand, the MAX objective represents the balance among the setups paid by the departments. Varying the MAX objective, while maintaining constant the value of MAX, allows the exploration of new solutions having the same SUM quality. In fact, the reduction (the increase) of the MAX objective generates a new solution  $p'$  in the solution space toward the center (extremity) of the Pareto front. In other words  $p'$  is either  $p' = (SUM_p + 1, MAX_p - 1)$  or  $p' = (SUM_p - 1, MAX_p + 1)$ .

More in details, this is done by the following procedures:

- An Iterated Local Search (first introduced in [4]) minimizes the SUM objective. It is used to attain a solution belonging to the Pareto front or close to it. Given an initial solution the ILS performs a Variable Neighborhood Descent (based on two complex neighborhoods) as local search step. The Perturbation phase applies random moves to the incumbent

solution. The actual number of perturbing moves is randomly drawn in the  $[1; 0.15|A|]$  interval. A solution is accepted as the new incumbent solution only if it strictly decreases the SUM objective. Each run of the ILS is stopped whenever the optimality of the solution is proved or when the computational bounds (3 seconds or 50 iterations) are reached. The lower bound used to prove the optimality and to stop the search process is described in details in [5].

- Once a Pareto point is obtained, COVER, a diversification procedure in the MAX objective, is performed. This procedure basically consists, given a solution possibly on the Pareto front, in both minimizing and maximizing the MAX objective and it permits a fast exploration of the Pareto front. In particular, given a solution  $p$  the COVER procedure attains a set of non dominated solutions that can be reached in a simple neighborhood [6] starting from  $p$ . Let  $p_{min}$  and  $p_{max}$  be the two extreme solutions that can be reached in the neighborhood starting from  $p$ , then all the solutions in the interval  $[p_{min}, p_{max}]$  are found by COVER.

### C. Update routine

This helper function updates the Pareto front adding, if necessary, the new Pareto solutions found by the solution improving procedures. Since COVER returns a set of non dominated solutions that are reachable by the solution found by the ILS phase the knowledge of the two extreme solutions ( $p_{min}$  and  $p_{max}$ ) is sufficient to calculate the exact amount of Pareto front covered in the iteration.

### D. Master procedures

The subordinate procedures are used by *Master procedures* to guide the search process toward the exploration of the Pareto front. Starting solutions are generated by the constructive greedy and they are improved by local search procedures previously described. In the following two Master procedures are introduced:

- The *Sweep* procedure (Figure 4) uses the constructive greedy to generate an *Equally Spaced Initial Solutions Set* (ESISS) which is used to control the search process. Starting from each solution in ESISS the improving procedures (ILS and COVER) are applied in cascade aiming to reach a Pareto point and consequently covering a portion of the Pareto front. The number of solutions in the ESISS is a parameter for the algorithm.
- The *Fill* procedure uses a different strategy. Starting from the knowledge of the coverage of the Pareto

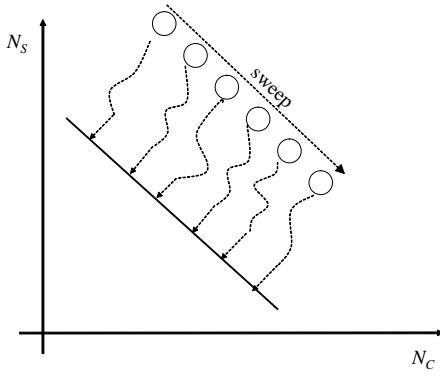


Fig. 4. The Sweep procedure.

front, an *hole* is identified, i.e., a portion of the Pareto front not yet explored by the algorithm. Given an hole, Fill pilots the constructive heuristic to build an initial solution having the same balance-ness of the hole (Figure 5). In this way it creates a new initial solution trying to drive the exploration to fill up the hole. The search process, in this case, is guided by the knowledge of the Pareto front found during the search process, and no additional parameters are required.

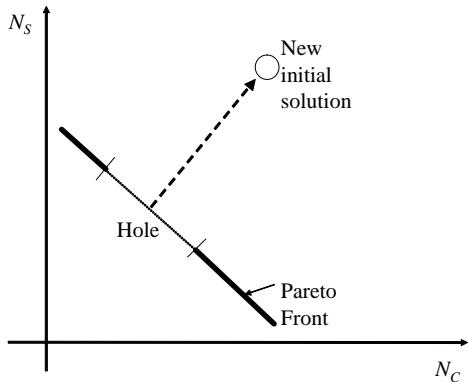


Fig. 5. The Fill procedure.

#### E. Metaheuristics

Using the two defined Master procedures we developed three different metaheuristic algorithms.

- The *Multi-Sweep* (MS) algorithm repeatedly applies the Sweep procedure. Once all the solutions of the

ESISS are considered, it starts again from the same sequence of piloting starting solutions. This process is iterated until the time limit  $t_{max}$  is reached or the whole Pareto front is covered.

- The *Sweep and Fill* (SF) algorithm uses a different strategy. Like Multi-Sweep it creates an Equally Spaced Initial Solution Set and it uses it as a pool of initial solutions. Once all the solutions in the ESISS are considered the algorithm applies the Fill procedure trying to fill the holes in the Pareto front until the time limit  $t_{max}$  is reached or the whole estimated Pareto front is covered.
- The *Front Fill* (FF) algorithm relies only on the Fill procedure. Clearly at the beginning no solution in the Pareto front has been yet identified, therefore it generates the initial solution corresponding to the central element of the Pareto front. The Fill procedure is repeated until the whole Pareto front is identified or the time limit  $t_{max}$  is reached.

## V. COMPUTATIONAL EXPERIMENTS

In this section we describe the experiments carried out to evaluate the behavior of the three proposed algorithms.

### A. Instances description

The algorithms have been tested on one set of 32 real-life instances (GSET) and on twelve sets of 20 randomly generated problems (RND10B–RND10E, RND30B–RND30E, RND60B–RND60E) [1]. The 32 real-life instances consist of unbalanced bipartite graphs  $G = (V_1, V_2, E)$  where  $|V_1| = 32$  and  $|V_2| = 14$ . In these instances  $|E|$  ranges from 150 to about 300. The other sets of instances consist of balanced bipartite graphs  $G = (V_1, V_2, E)$ , with cardinality  $n = |V_1| = |V_2|$  and graph density  $d$  ranging from 10% to 40%. In particular, 10, 30 and 60 nodes have been considered for each department, and 10, 20, 30, 40% for the density  $d$ . For each pair  $(n, d)$  a set of 20 connected instances has been generated.

In a preliminary test phase we tuned the proposed algorithms. On all the tests  $t_{max}$  is set to 300 seconds. The algorithms stop when one of the following conditions holds: (i) the maximum number of starting solutions is reached (at most 500,000); (ii) the computation time exceeds  $t_{max}$ ; (iii) the whole Pareto front is found.

The cardinality of the ESISS (the number of initial solution in the Sweep routine) for MS and SF is set to  $((|PF| * d)/2) + 1$ , where  $|PF|$  is the cardinality of the ideal Pareto front calculated according to Equations 5 and 6. The knowledge of the cardinality of the ESISS permits to calculate easily the parameter  $\rho$  controlling

the constructive algorithm. Since the algorithms are stochastic, for each instance 10 runs are executed in order to obtain a meaningful representation of the behavior of the developed algorithms.

All the experiments have been performed on a 3.2 GHz Pentium 4 laptop equipped with 512 MB ram, and the algorithms are coded in standard C language.

### B. Performances Analysis

In Table I, we summarize the results for each algorithm on each test set showing the computation time in seconds (time), the number of initial solutions generated (Iter.) and the Pareto front covering percentage (%). Each row refers to the average results over 20 instances and 10 repetitions. On average a large part of the Pareto front is covered by all algorithms (%) in a small computation time. In general, MS performs better on sparse instances (sets RND30B and RND60B). On the other hand, as the graph density grows, SF and FF require a smaller amount of time to cover the same percentage of the front. The second algorithm (SF) implements a more sophisticated strategy, in fact once all the initial solutions in the ESISS are explored, other promising initial solutions are generated by the Fill routine. Hence, in general, a lower number of improving procedures are needed to attain the same covering percentage of the Pareto front. Finally, the last algorithm (FF) generates initial starting solutions dynamically guided by the Pareto front and its performances are very similar to those of SF.

It is important to note that for real-life instances, which have a relatively low number of nodes and high density, all the proposed algorithms achieve a covering percentage close to 100% ( $\geq 99\%$ ). Regarding the balanced and small sets (RND10B–RND10E), the average percentage of covering is quite high ( $\geq 85\%$ ), even if a very high number of starting solutions are generated.

In Tables II–III we report, for each instance and for each algorithm, the computation time in seconds (time), the total number of initial solutions generated (Iter.) and the Pareto front covering percentage (%). Each row refers to the average results over 10 repetitions. Observe that in Table III (small and sparse instances) the number of iterations to attain all the Pareto front is, in general, small; but there are some remarkable exceptions where the number of iterations reached is maximum (i.e., 500,000), while the Pareto coverage is smaller than 100%. In other words, the algorithms try without succeeding to cover all the front, generating a great number of initial solutions. In fact, some points in the Pareto front may not correspond to a solution (see for example RND10B.006). Moreover, the Pareto

front calculated as described in Section III-B may even be unreachable, since the  $LB_{MAX}$  value is only lower bound on the optimal value. In fact some instances (see for example RND10B.007) have a null Pareto coverage.

### VI. CONCLUSIONS AND FUTURE RESEARCH

In this paper a two objective setup coordination problem arising in a two stage serial manufacturing system is addressed. The problem consists in finding a common sequence of batches to be produced such that the number of setup paid on each department is minimized. In particular the case in which all the setups are identical has been considered. Since the peculiarity of the problem it is possible to give a geometrical description of the Pareto front of the search space. Three heuristic algorithms for the problem have been proposed and they have been extensively tested on a wide set of instances. The proposed algorithms are able to cover large portions of the optimal Pareto front in average, within a reasonable computation time.

Future research directions may aim to improve the lower bound procedure and to try to establish a priori that some points of the Pareto front do not corresponds to feasible solutions. Such improvements could guarantee better performance of the algorithms both in terms of quality and computation time.

From the decision maker point of view it could be also interesting to find different solutions (i.e. sequences) corresponding to each Pareto point. In fact, the proposed approach is only interested in finding only a solution for each Pareto point.

### REFERENCES

- [1] Agnetis, A., Detti, P., Meloni, C., Pacciarelli, D., (2001), *Setup coordination between two stages of a supply chain*, Annals of Operations Research, 107, 15–32,
- [2] Al-Haboubi, M.H., Selim, S.Z., (1993), *A sequencing problem in the weaving industry*, European Journal of Operational Research, 66, 65–71.
- [3] Allahverdi, A., Gupta J.N.D., Aldowaisan, T., (1999), *A review of scheduling research involving setup considerations*, Omega, 27, 219–239.
- [4] Detti, P., Meloni, C., Pranzo, M., (2003), *Local Search Algorithms for the Minimum Cardinality Dominating Trail Set of a Graph*, Technical report RT-DIA-84-2003, Dipartimento di Informatica e Automazione, Università Roma Tre, Roma, Italy.
- [5] Detti, P., Meloni, C., Pranzo, M., (2004), *Simple bounds for the minimum cardinality dominating trail set problem*, Technical report RT-DIA-87-2004, Dipartimento di Informatica e Automazione, Università Roma Tre, Roma, Italy.
- [6] Detti, P., Meloni, C., Pranzo, M., (2005), *Minimizing and balancing setups in a serial production system*, Technical report 01/05, Dipartimento di Ingegneria dell'Informazione, Università di Siena, Italy.
- [7] Glover, F., Kochenberger, G., (2003), *Handbook of Metaheuristics*, Kluwer Academic Publishers.

TABLE I  
RESULTS

Instance set	MS			SF			FF		
	time	Iter.	%	time	Iter.	%	time	Iter.	%
GSET	0.24	229.11	100	12.84	2971.02	99.99	15.03	3814.18	99.99
RND10B	45.39	125266.59	86.67	41.71	103605.94	88.33	40.41	104701.19	88.33
RND10C	11.12	25875.71	95	25.63	51308.02	92.50	19.59	40884.75	93.50
RND10D	1.56	2941.03	100	1.24	2866.12	100	1.74	3730.37	100
RND10E	<0.01	10.15	100	<0.01	12.35	100	<0.01	10.42	100
RND30B	14.51	4314	99.85	22.77	5322.02	99.81	21.40	5159.41	99.83
RND30C	0.06	42.75	100	0.10	43.38	100	0.35	169.23	100
RND30D	0.18	131.68	100	0.11	79.35	100	0.09	68.51	100
RND30E	0.38	242.17	100	0.26	177.25	100	0.24	166.06	100
RND60B	3.93	175.13	100	27.90	875.08	99.98	28.14	887.33	99.98
RND60C	17.32	511.68	100	12.56	369.25	100	12.58	373.33	100
RND60D	58.16	1412.07	100	36.68	837.44	100	38.82	906.08	100
RND60E	131.15	2423.39	99.99	95.31	1642.47	100	115.18	2015.77	99.98

TABLE II  
GSET

Instance name	MS			SF			FF		
	time	Iter.	%	time	Iter.	%	time	Iter.	%
GSET1	0.32	316.80	100	0.06	81.80	100	0.11	157.90	100
GSET2	0.28	290.40	100	0.08	101.20	100	0.01	23	100
GSET3	0.41	422.40	100	0.10	140.50	100	0.03	50.30	100
GSET4	0.20	202.40	100	0.09	111.80	100	0.08	117.80	100
GSET5	0.42	405	100	0.07	78.70	100	0.20	274.10	100
GSET6	0.23	207	100	0.09	108.20	100	0.09	123.50	100
GSET7	0.35	280.60	100	26.16	11347.10	100	64.62	26763.30	100
GSET8	0.11	99	100	0.15	183.60	100	0.13	179.80	100
GSET9	0.41	395.60	100	0.07	81	100	0.06	89	100
GSET10	0.22	216.20	100	0.12	162	100	0.11	145	100
GSET11	0.14	138	100	0.15	200.80	100	0.09	119.80	100
GSET12	0.19	184	100	0.13	168	100	0.15	225.50	100
GSET13	0.71	441.80	100	115.61	22887.40	99.88	174.37	36870.20	99.84
GSET14	0.52	414	100	160.20	33102.40	99.88	74.47	16850.30	100
GSET15	0.24	174.80	100	84.99	18175.80	99.96	82.80	18889.80	99.92
GSET16	0.23	179.40	100	18.50	4432.20	100	81.20	18535.20	100
GSET17	0.08	96	100	1.98	1456.30	100	0.64	469.30	100
GSET18	0.02	30	100	0.03	75	100	0.04	106.40	100
GSET19	0.14	175.50	100	1.35	967.90	100	0.44	324.10	100
GSET20	0.04	75	100	0.02	51	100	0.03	83.40	100
GSET21	0.14	135	100	0.11	131.90	100	0.15	182.60	100
GSET22	0.20	198	100	0.10	123.20	100	0.08	138.80	100
GSET23	0.46	450	100	0.08	96.60	100	0.05	83.10	100
GSET24	0.27	265.50	100	0.07	91	100	0.08	108.80	100
GSET25	0.30	328.50	100	0.12	161.40	100	0.10	132.90	100
GSET26	0.09	99	100	0.09	123	100	0.07	115.50	100
GSET27	0.35	382.50	100	0.05	70	100	0.18	253.50	100
GSET28	0.19	198	100	0.05	59.10	100	0.08	118.20	100
GSET29	0.06	63	100	0.07	92.40	100	0.13	209	100
GSET30	0.05	45	100	0.07	92.40	100	0.09	120.10	100
GSET31	0.17	162	100	0.05	53	100	0.08	104.20	100
GSET32	0.27	261	100	0.06	65.80	100	0.06	89.40	100
Avg	0.24	229.11	100	12.84	2971.02	99.99	15.03	3814.18	99.99

- [8] Ehrgott, M., (2000), *Multicriteria Optimization*, Lecture Notes in Economics and Mathematical Systems, Springer–Verlag.
- [9] Hoos, H.H., Stützle, T., (2004), *Stochastic Local Search: Foundations and Applications*, Morgan Kaufmann Publishers.
- [10] McIntosh, R.I., Culley, S.J., Mileham, A.R., Owen , G.W., (2001), *Changeover improvement: A maintenance perspective*, International Journal of Production Economics, 73, 153–163.
- [11] Potts, C.N., Kovalyov, M.Y., *Scheduling with batching: A review*, European Journal of Operational Research. **120**, 228–249, 2000.
- [12] Ríos–Mercado, R.Z., Bard, J.F., (1998), *Heuristics for the flow line problem with setup costs*, European Journal of Operational Research, 110, 76–98.
- [13] Spina, R., Galantucci, L.M., Dassisti, M., (2003), *A hybrid approach to the single line scheduling problem with multiple products and sequence-dependent time*, Computers & Industrial Engineering, 45, 573–583.
- [14] T'kindt, V., Billaut, J.C., (2002), *Multicriteria scheduling: theory, models and algorithms*, Springer, Berlin.
- [15] Tsujimura, Y., Gen, M., (1999), *Parts loading scheduling in a flexible forging machine using an advanced genetic algorithm*, Journal of Intelligent Manufacturing, 12 (3), 413–420.
- [16] Voß, S., Woodruff, D.L., (2003), *Introduction to computational optimization models for production planning in a supply chain*,

TABLE III  
RND10B

Instance name	MS			SF			FF		
	time	Iter.	%	time	Iter.	%	time	Iter.	%
RND10B0	<0.01	2	100	<0.01	2	100	<0.01	6.30	100
RND10B1	0.19	336	100	0.69	1545.20	100	0.16	474.40	100
RND10B2	0.01	41.60	100	0.01	15	100	0.02	55.90	100
RND10B3	<0.01	2	100	<0.01	2	100	<0.01	5	100
RND10B4	<0.01	2	100	<0.01	2	100	<0.01	1	100
RND10B5	<0.01	4	100	<0.01	8	100	<0.01	2	100
RND10B6	170.30	500000	66.67	14.66	68028.40	100	19.83	90517	100
RND10B7	202.46	500000	0	203.74	500000	0	196.64	500000	0
RND10B8	140.95	500000	66.67	184.67	500000	66.67	157.68	500000	66.67
RND10B9	2.39	4139.40	100	0.91	2359.80	100	0.78	2033.50	100
RND10B.010	0.01	2	100	<0.01	2	100	<0.01	1	100
RND10B.011	<0.01	4.80	100	<0.01	4	100	<0.01	9	100
RND10B.012	0.11	232.80	100	<0.01	2	100	0.26	842.30	100
RND10B.013	0.01	2	100	<0.01	7	100	<0.01	2	100
RND10B.014	0.14	528.40	100	0.04	116.80	100	0.03	64	100
RND10B.015	0.01	6	100	<0.01	8	100	<0.01	2.10	100
RND10B.016	191.99	500000	50	203.21	500000	50	198.59	500000	50
RND10B.017	<0.01	16.80	100	0.01	10.50	100	0.01	3.20	100
RND10B.018	199.21	500000	50	226.24	500000	50	234.13	500000	50
RND10B.019	<0.01	2	100	<0.01	2	100	0.04	1	100
Avg	45.39	125266.59	86.67	41.71	103605.94	88.33	40.41	104701.19	88.33

Springer–Verlag, Berlin.

- [17] Wolsey, L.A., (1997), *MIP modelling of changeovers in production planning and scheduling problems*, European Journal of Operational Research, 99, 154–165.

# Two unifying frameworks in voting theory

Estefanía García, José Luis Jimeno and Joaquín Pérez  
 Universidad de Alcalá/Dept. de Fundamentos de Economía e Historia Económica  
 Plaza de la Victoria 2, 28802 Alcalá de Henares. Madrid. Spain.  
 Email: [estefania.garcia@uah.es](mailto:estefania.garcia@uah.es), [josel.jimeno@uah.es](mailto:josel.jimeno@uah.es), [joaquin.perez@uah.es](mailto:joaquin.perez@uah.es)

**Abstract**—Two unifying frameworks (conceptual structured framework inside of which some voting methods lie) are proposed and described, and two new voting methods, one obtained in each framework, are defined and analysed with respect to the properties of Monotonicity and Participation.

**Keywords**—Voting; Monotonicity; Participation

## I. INTRODUCTION

Voting methods have in general been defined directly, by means of algorithms, and only further investigations have allowed to notice some similarities among apparently distant algorithms. The point of view from which these similarities are revealed is an enriching one because it facilitates a better understanding of algorithms and methods and their properties.

Some times this point of view is a kind of conceptual structured framework inside of which some voting methods lie. This framework, which will be called **Unifying Framework**, depends explicitly on some characteristics or parameters, in such a way that every particularization of these parameters leads to a particular voting method.

Some contributions from this point of view to the voting and decision literature are Marcotorchino and Michaud [9], who classify aggregation procedures based on the minimization of distances, Barthelemy and Monjardet [2], who study general relations and distances, and Lerer and Nitzan [7] and Campbell and Nitzan [4], who characterize some voting methods by means of distances between preference profiles.

It is worth to note that the subject of this paper is perfectly relevant, in a wide sense, to the operations research field. Since a voting method is (apart from the possible strategic aspects, not studied here) a process of aggregation of preferences, it is formally analogous to a multicriteria decision process, where the preferences of voters correspond to evaluation criteria. Therefore, there is a close relationship between the field of social choice and voting and

the field of operations research, not only in the subfield of preference aggregation but also in that of preference modelling. In fact, many studies in the ordinal multicriteria decision literature make use of the terminology, results and intuitions of the social choice and voting literature. See Marchant [8] and Arrow and Raynaud [1] for a study of the relation between these fields, and see Bouyssou et al [3], Arrow and Raynaud [1], Pérez [12], Pérez and Barba-Romero [14] and Pomerol and Barba-Romero [15] for studies in multicriteria decision contexts which make use of the voting methodology and insights.

The aim of this paper is twofold. Firstly, we will study two such unifying frameworks (the first one is a generalization, studied in Jimeno [5], of the idea leading to the well-known Dogdson voting method, while the second one is an extension of some distance frameworks, started in Pérez [11]), and on the other hand we will identify two relatively new voting methods, one from each framework, and analyse them from the point of view of Monotonicity and Participation Properties. Although we are dealing with voting theory, this work is also applicable to other aggregation contexts like ordinal multicriterion decision.

## II. TERMINOLOGY AND PRELIMINARIES

Let  $X = \{x_1, x_2, \dots, x_n\}$  be a finite set with two or more candidates. Preferences of any voter are supposed to take the form of a complete ranking, that is to say, a linear (strict and complete) order  $L$  over  $X$ .

We say  $L = xyzt\dots$  to denote the preference order in which  $x$  is the most preferred candidate,  $y$  is the second one, and so on, and  $a L b$  means that  $a$  is preferred to  $y$  in  $L$ .

Given the set of candidates  $X = \{x_1, x_2, \dots, x_n\}$ , and any finite set  $V = \{1, 2, \dots, m\}$  with one or more voters, we call a **Situation** any pair  $(X, p)$ , where  $p$  is a preference profile over  $X$  from  $V$ , that is to say, a  $m$ -tuple of orders over  $X$ , each one meaning the preferences of a voter over  $X$ .

We call **Voting Correspondence** (from now on **VC**) any function  $f$  which maps any situation  $(X, p)$  to a non-empty subset of  $X, f(X, p)$ . The elements of  $f(X, p)$  are the chosen candidates (the winners) over  $X$  from the preference profile  $p$ . Given any two VCs  $f$  and  $g$ , we call  $f$  a **refinement** of  $g$  when  $f(X, p) \subseteq g(X, p)$  for every  $(X, p)$ . A VC which, for any situation, chooses only one candidate is called **Voting Function**.

We will consider only anonymous VCs. Thus, a preference profile over  $X$  from  $V$  can also be described by specifying how many of the  $m$  voters from  $V$  sustain any of the  $n!$  linear orders on  $X$ .

Given any  $X$ , any two disjoint sets of voters  $V_1 = \{1, 2, \dots, m_1\}$  and  $V_2 = \{m_1+1, m_1+2, \dots, m_1+m_2\}$ , and any two preference profiles  $p_1$  and  $p_2$  over  $X$  from, respectively,  $V_1$  and  $V_2$ , we can merge these two profiles in order to obtain a new profile over  $X$ , but now originated from  $V_1 \cup V_2$ . This new profile will be called  $p_1 + p_2$ .

Given any situation  $(X, p)$  with  $n$  candidates and  $m$  voters,  $r(x, j)$  means the number of voters in  $p$  for which  $x$  is the  $j$ -th preferred candidate (that is to say,  $j-1$  candidates are preferred to  $x$  and  $n-j$  candidates are less preferred than  $x$ ). The square  $n \times n$  matrix  $R_p$ , whose entries are  $r(x, j)$ , will be called the **Rank Matrix** for  $(X, p)$ . Given  $(X, p)$  and any two different candidates  $x, y$  from  $X$ ,  $p(x, y)$  means the number of voters in  $p$  which prefer  $x$  to  $y$ . Because ties are not allowed in any voter's ballot,  $p(x, y) + p(y, x) = m$ . The square  $n \times n$  matrix  $M_p$ , whose entries are  $p(x, y)$ , will be called the **Comparison Matrix** for  $(X, p)$ . For every candidate  $x$ , the sum of the off-diagonal row entries in  $M_p$  is called the **Borda Score** of  $x$ .

Candidate  $x$  is said to **beat**  $y$ , denoted by  $xW_p y$ , when  $p(x, y) > p(y, x)$ . If we use  $\geq$  instead of  $>$ , we have the relation  $x$  **beats or ties**  $y$ , which is denoted by  $xU_p y$ . If  $x$  beats any other, then  $x$  is called the **Condorcet** candidate. A VC  $f$  is said to be a **Condorcet correspondence** if  $f(X, p) = \{x\}$  for every situation  $(X, p)$  in which  $x$  is a Condorcet candidate.

We call **Victories Matrix**, denoted  $V_p$ , to the square  $n \times n$  matrix with entries  $v(x, y) = 1$  if  $p(x, y) > p(y, x)$  and 0 in any other case

For every order  $L: x_1 x_2 \dots x_n$  of the candidates, the sums  $K(L) = \sum_{i < j} p(x_i, x_j)$  and  $S(L) = \sum_{i < j} v(x_i, x_j)$  will be called, respectively, the **Kemeny Score** of  $L$  and the **Slater Score** of  $L$ .

Any change in  $M_p$  by which a unit is added to the off-diagonal entry  $p(x, y)$  and subtracted from  $p(y, x)$ , is called an **elementary interchange** in  $M_p$ . Any change in  $(X, p)$ , by which two consecutive

candidates  $x$  and  $y$  in a voter's order interchange their position in that voter's order, is called an **elementary interchange in  $(X, p)$** .

Given any nonempty set  $\Omega$ , we call a **distance** in  $\Omega$  to any map  $d$  which assigns a nonnegative real number  $d(a, b)$  to any pair  $(a, b)$  of elements of  $\Omega$ , satisfying:

- 1)  $d(a, b) = 0$  if and only if  $a = b$ .
- 2)  $d(a, b) = d(b, a)$  for every  $a, b \in \Omega$ .
- 3)  $d(a, c) \leq d(a, b) + d(b, c)$  for every  $a, b, c \in \Omega$ .

Any such a map satisfying 2) and 3) is called a **pseudodistance**.

In the space  $\mathbf{R}^h$ , the best known family of distances is the **p-norm** family, defined as follows for any real positive  $p$ :

$$d_p(\mathbf{x}, \mathbf{y}) = (\lvert x_1 - y_1 \rvert^p + \lvert x_2 - y_2 \rvert^p + \dots + \lvert x_h - y_h \rvert^p)^{\frac{1}{p}}$$

where  $\mathbf{x} = (x_1, x_2, \dots, x_h)$  and  $\mathbf{y} = (y_1, y_2, \dots, y_h)$ .

The most frequently used are the 1-norm distance  $d_1$ , also called the taxicab or the **Manhattan** distance and the 2-norm distance  $d_2$ , also called the **Euclidean** distance. Another distance closely related to this family is the infinity norm distance  $d_\infty$ , which is defined as the limit of  $d_p$  as  $p$  goes to infinity. It is easy to see that:

$$d_\infty(\mathbf{x}, \mathbf{y}) = \lim_{p \rightarrow \infty} \left( \sum_{i=1}^h \lvert x_i - y_i \rvert^p \right)^{\frac{1}{p}} = \max_{i=1,2,\dots,h} \lvert x_i - y_i \rvert$$

Another interesting distance is the following:

$$d_{cd}(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^h \text{sgn} \lvert x_i - y_i \rvert, \text{ that is to say, the number of components in which } \mathbf{x} \text{ and } \mathbf{y} \text{ differ.}$$

A natural and easy way of constructing pseudodistances in  $\mathbf{R}^h$ , is to apply a distance to a proper subset of the components of the vectors. For

example,  $\delta(\mathbf{x}, \mathbf{y}) = (\lvert x_1 - y_1 \rvert^p + \lvert x_2 - y_2 \rvert^p)^{\frac{1}{p}}$  is a pseudodistance in  $\mathbf{R}^h$  constructed from the distance  $d_p$  in  $\mathbf{R}^2$  applied to the first and second components of the vectors  $\mathbf{x}$  and  $\mathbf{y}$ .

*Some well-known correspondences and some monotonicity and participation properties*

Let us define the following correspondences identifying, for every situation  $(X, p)$ , the set of winning candidates.

**Definition 1:**

a) **Plurality** correspondence ( $f_{PLU}$ ).

$f_{PLU}(X, p) \equiv \{x \in X : r(x, 1) \geq r(y, 1) \text{ for every } y \in X\}$

b) **Borda** correspondence ( $f_{BORDA}$ ).

$f_{BORDA}(X, p) \equiv \{x \in X : \sum_{j=1 \dots n} (n-j-1)r(x, j) \geq \sum_{j=1 \dots n} (n-j-1)r(y, j) \text{ for every } y \in X\}$

Or, equivalently,

$f_{BORDA}(X, p) \equiv \{x \in X : x \text{ has a maximal Borda score}\}$

c) **Kemeny** correspondence ( $f_{KEM}$ ).

$f_{KEM}(X, p) \equiv \{x \in X : x \text{ is at the top of an order } L \text{ with maximal Kemeny Score}\}$

d) **Simpson-Cramer Minmax** correspondence ( $f_{MINMAX}$ ).

$f_{MINMAX}(X, p) \equiv \{x \in X : \text{The minimal off-diagonal term of row } x \text{ in } M_p \text{ is maximal}\}$

e) **Dogdson** correspondence ( $f_{DOG}$ ).

$f_{DOG}(X, p) \equiv \{x \in X : \text{The number of elementary interchanges in } (X, p) \text{ needed by } x \text{ to become a Condorcet candidate is minimal}\}$

**Definition 2:** A VC  $f$  satisfies the **Monotonicity** property if for any given pair of situations  $(X, p)$  and  $(X, p'_x)$ , where  $p'_x$  is a profile obtained from  $p$  by just making an elementary interchange favourable to  $x$ , If  $x \in f(X, p)$ , then  $x \in f(X, p'_x)$ .

We say  $f$  satisfies **Strict Monotonicity** when If  $x \in f(X, p)$ , then  $f(X, p'_x) = \{x\}$ .

We say  $f$  satisfies **Semi-Strict Monotonicity** when it satisfies Monotonicity and also satisfies that

If  $z \notin f(X, p)$ , then  $z \notin f(X, p'_x)$ .

In other words, Monotonicity requires that if a chosen candidate  $x$  is favoured by an elementary interchange,  $x$  will remain chosen. Strict Monotonicity also requires  $x$  will be the only chosen, while Semi-Strict Monotonicity also requires that no candidate not chosen before will be chosen now.

**Definition 3:** A Voting Function  $f$  satisfies the **Participation** property if for any given pair of situations  $(X, p)$  and  $(X, v)$ , where profile  $v$  has only one voter,  $f(X, p) = \{x\}$  and  $x$  is preferred to  $y$  in  $v$ ) implies  $f(X, p+v) \neq \{y\}$ .

In words, if  $x$  is the winner for a situation and a new voter who prefers  $x$  to  $y$  is added, candidate  $y$  will not become the winner. Thus, the new voter could not do better abstaining. Moulin [10] shows that no Condorcet voting function satisfies Participation.

The following properties are adaptations of this property to the voting correspondences framework.

**Definition 4:** A VC  $f$  satisfies the **Optimist Participation** property if for any given pair of situations  $(X, p)$  and  $(X, v)$ , where profile  $v$  has only

one voter, If  $x \in f(X, p)$ , then  $(x \in f(X, p+v))$  or there exists a candidate  $z$  preferred to  $x$  in  $v$  such that  $z \in f(X, p+v)$ .

We say  $f$  satisfies **Strict Optimist Participation** when it satisfies Optimist Participation and also satisfies that If  $x \in f(X, p)$ , then there is no  $y$  less preferred than  $x$  in  $v$  such that  $y \in f(X, p+v)$ .

We say  $f$  satisfies **Semi-Strict Optimist Participation** when it satisfies Optimist Participation and also satisfies that If  $x \in f(X, p)$ , then there is no  $y$  less preferred than  $x$  in  $v$  such that  $y \in f(X, p+v)$ , unless  $x \in f(X, p+v)$ .

In other words, Optimist Participation requires that if candidate  $x$  is chosen,  $x$  or another candidate  $z$  preferred by  $v$  to  $x$ , will be chosen when a new voter  $v$  is added. Strict Optimist Participation also requires that no  $y$  less preferred than  $x$  will be chosen, while Semi-Strict Optimist Participation also requires that no candidate less preferred than  $x$  not chosen before will be chosen now.

Jimeno et al [6] shows that no Condorcet voting function satisfies Optimist Participation.

**Definition 5:** A VC  $f$  satisfies the **Positive Participation** or **Positive Involvement (PI)** property if for any given pair of situations  $(X, p)$  and  $(X, v)$ , where profile  $v$  has only one voter, If  $x \in f(X, p)$  and  $x$  is preferred to any  $y$  in  $v$ , then  $x \in f(X, p+v)$ .

We say  $f$  satisfies **Strict Positive Involvement (SPI)** when If  $x \in f(X, p)$  and  $x$  is preferred to any  $y$  in  $v$ , then  $f(X, p+v) = \{x\}$ .

We say  $f$  satisfies **Semi-Strict Positive Involvement (S-SPI)** when If  $x \in f(X, p)$  and  $x$  is preferred to any  $y$  in  $v$ , then  $x \in f(X, p+v)$  and  $f(X, p+v) \subseteq f(X, p)$ .

In other words, Positive Involvement requires that if candidate  $x$  is chosen,  $x$  will remain chosen when a new voter is added whose favourite is  $x$ . Strict Positive Involvement also requires that  $x$  will be the only chosen, while Semi-Strict Positive Involvement also requires that no candidate not chosen before will be chosen now.

Pérez [13] shows that almost no Condorcet voting function satisfies Positive Participation. Cramer-Simpson Minmax is an exception.

Failing to satisfy any of the above Participation properties means that a **No Show paradox** sets in.

**Definition 6:** A VC  $f$  satisfies the **Consistency** property if for any two situations  $(X, p_1)$  and  $(X,$

$p_2)$ ,  $f(X, p_1) \cap f(X, p_2) \neq \emptyset$  implies  $f(X, p_1 + p_2) = f(X, p_1) \cap f(X, p_2)$ .

In other words, if some candidates are chosen for profile  $p_1$  and profile  $p_2$ , they, and only they, are chosen when the two profiles are merged.

This property, sometimes called Young's Consistency, is used in Young [16] in order to make a characterization of the Borda rule where the Consistency property plays a fundamental role.

The positional voting correspondences, whose best known examples are the **Plurality rule** and the **Borda rule** are known to satisfy this property.

### III. UNIFYING FRAMEWORKS

The following two frameworks, although not completely new, have been less studied than others.

#### A. A first unifying framework

On an informal way, it seems reasonable to define a voting method by means of the conjunction of the following elements:

- i) A Perfection Criterion whose fulfilment determines the set of winners, that is derived from a binary relation or comparison on the set of candidates of each situation  $(X, p)$ .
- ii) An Improvement Criterion or a unit of measurement on  $(X, p)$  that allows, in absence of candidates who fulfil the perfection criterion, to determine the set of winners, as the ones that are nearest to the fulfilment of the perfection criterion, according to the established improvement criterion.

This way, we can define voting methods on an analogous way to as some Condorcet methods are defined. In order to clarify the idea we consider some Condorcet methods. In them, the binary relation of strict majority allows to define as perfection criterion the Condorcet principle (according to which whenever a Condorcet candidate exists this candidate is the only winner). Nevertheless, if this candidate does not exist in a situation  $(X, p)$  each method establishes its own rules. Thus, for example, the Dogson method considers elementary interchanges in voters preferences and takes as winner that candidate who requires less elementary interchanges to become Condorcet, since that candidate is, according to this criterion, closest to perfection. On the other hand, the Young method considers as winner that candidate who requires eliminating fewest voters to become Condorcet.

Although we have used some Condorcet methods to exemplify the idea, in fact it is valid for

other voting methods, depending on the binary relation and perfection criterion we use, and on the way we define the improvement criterion that allows measuring the distances to the supposedly perfect candidates for each criterion.

Therefore, in a certain way, we are talking about the application of what in the multicriterion context means to minimize the distance to the ideal, considering that the ideal is defined by a perfection criterion and the distance is measured using an improvement criterion.

#### 1) Binary relations and Perfection Criterion

Let us consider a situation  $(X, p)$  with  $X = \{x_1, x_2, \dots, x_n\}$  and  $m$  voters. Call  $R$  an irreflexive and asymmetric relation, not necessarily transitive, between candidates that we will interpret on the following way: "If  $xRy$ , candidate  $x$  has more merits than  $y$  to become winner in  $(X, p)$ , when we restricted  $X$  to the set  $\{x, y\}$ ". An Example of this kind of binary relation already constitutes the relation defined of strict majority.

Given a relation  $R$ , we can define the concepts of  $R$ -Perfect candidate and  $R$ -Perfection criterion:

**Definition 7 ( $R$ -Perfect Candidate).** Given a situation  $(X, p)$  and a relation  $R$ , we say that a candidate  $x$  is  $R$ -Perfect in  $(X, p)$  if  $xRy$  for every  $y \in X \setminus \{x\}$ .

In words, any candidate is  $R$ -perfect if it overcomes in relation  $R$  to all the other candidates. If we consider, for example, the strict majority relation, we will say that  $x$  is  $R$ -Perfect if  $x$  is a Condorcet candidate.

Thus, in the same way that the majority relation allows us to define the Condorcet criterion like the angular stone of a family of voting methods, each possible relation  $R$  allows us to define a  $R$ -Perfection criterion that will be used as a base for the definition of different families of voting methods:

**Definition 8 (Respecting the  $R$ -Perfection).** Given a relation  $R$ , we say that a voting method  $f$  respects the  $R$ -perfection if in all situations  $(X, p)$  in which a  $R$ -perfect candidate exists, that candidate is the only one chosen.

The asymmetry of  $R$  guarantees that if in a given situation  $(X, p)$  exists a  $R$ -perfect candidate, this candidate is unique.

It is worth to note that if the relation  $R$  is transitive we can say that, by definition, a method

$f$  respects the  $R$ -relation if in all situations in which  $xRy$ , the candidate  $y$  is not chosen. In that case, it is clear that if  $f$  respects the  $R$ -relation then  $f$  also respects the  $R$ -perfection. If, on the contrary,  $R$  is not transitive, in such a way that it presents/displays cycles, the property " $f$  respects the  $R$ -relation" is not well defined, because in all situation  $(X, p)$  in which  $x_1Rx_2R \dots Rx_nRx_1$ , there would be no winner at all.

**Definition 9 (Respect to  $R$ -Relation).** Given an irreflexive, asymmetric and transitive relation  $R$ , we say that method  $f$  respects  $R$  if and only if, for all situation  $(X, p)$  in which  $xRy$ , candidate  $y$  is not chosen.

Although we can define a lot of perfection criteria based on different binary relationship, we focus on the domination relations, the stronger of which is the Pareto domination.

## 2) Perfection Criteria based on Domination Relations.

The domination relations we define next take their base in the candidate comparison two by two and therefore in the majority relation.

### Pareto Domination Relation.

**Definition 10 (Pareto Domination Relation,  $R_P$ ).** We say that  $R$  determines a domination relation in the sense of Pareto, denoted by  $R_P$ , if given any situation  $(X, p)$  and given any pair of candidates  $x, y \in X$ , " $x R_P y$  if and only if  $p(x, y) = m$ ", that is to say,  $x$  dominates  $y$  in the sense of Pareto (all voter prefers  $x$  to  $y$ ).

This way, a candidate is  $R_P$ -Perfect if he is the unique Pareto optimum. This means that a voting procedure  $f$  respects  $R_P$ -Perfection if, whenever a unique Pareto optimum exists,  $f$  selects it as the only winner.

Two generalizations of the concept of Pareto domination can be expressed in the following way:

### C2-Domination Relation.

**Definition 11 (C2-Domination Relation,  $R_{C2}$ ).** We will say that  $R$  determines a C2-Domination Relation, and we will denote it by  $R_{C2}$ , if given any situation  $(X, p)$  and any pair of candidates  $x, y \in X$ , " $x R_{C2} y$  if and only if  $\{p(x, y) > p(y, x)\}$  and  $p(x, z) \geq p(y, z), \forall z \in X \setminus \{x, y\}$ ", that is to say,  $x$  dominates  $y$  in C2 sense.

This relation determines that a candidate is  $R_{C2}$ -Perfect when he dominates in C2 sense to all the

others. In this sense, a voting procedure  $f$  respects  $R_{C2}$ -Perfection if it selects the C2-dominant candidate as the only winner whenever it exists.

### C1-Domination Relation.

**Definition 12 (C1-Domination Relation,  $R_{C1}$ ).** We will say that  $R$  determines a domination relation in C1 sense, and we will denote it by  $R_{C1}$  if, given any situation  $(X, p)$  and any pair of candidates  $x, y \in X$ , " $x R_{C1} y$  if and only if  $\{p(x, y) > p(y, x)\}$  and, in addition,  $\forall z \in X \setminus \{x, y\}$ , if  $p(y, z) \geq p(z, y)$  then  $p(x, z) > p(z, x)\}$ ", that is to say,  $x$  dominates  $y$  in the C1 sense.

This relation determines that a candidate is  $R_{C1}$ -Perfect when he dominates in C1 sense all the others. In this sense, a voting procedure  $f$  respects  $R_{C1}$ -Perfection, if it selects the C1-dominant candidate as the only winner whenever it exists.

By the definition of  $R_{C1}$ , a  $R_{C1}$ -Perfect candidate is a Condorcet candidate, and therefore, all voting procedures  $f$  that respects  $R_{C1}$ -Perfection are in fact Condorcet procedures, thus choosing the Condorcet candidate in all situation in which it exists.

### Proposition 1.

Let  $f$  be a voting correspondence:

- a) If  $f$  respects  $R_{C1}$ -Perfection, then  $f$  respects  $R_{C2}$ -Perfection.
- b) If  $f$  respects  $R_{C2}$ -Perfection, then  $f$  respects  $R_P$ -Perfection.

### Proof:

a) If  $f$  does not respect  $R_{C2}$ -Perfection, then there exists a situation  $(X, p)$  with  $m$  voters, in which a  $R_{C2}$ -Perfect candidate  $x$  exists so that  $\{x\} \neq f(X, p)$ . This means that a  $R_{C1}$ -Perfect candidate exists, the same  $x$ , which is not the only winner. Thus,  $f$  does not respect  $R_{C1}$ -Perfection.

b) If  $f$  does not respect  $R_P$ -Perfection, then there exists a situation  $(X, p)$  with  $m$  voters, in which a  $R_P$ -Perfect candidate  $x$  exists so that  $\{x\} \neq f(X, p)$ . This means that a  $R_{C2}$ -Perfect candidate exists, the same  $x$ , which is not the only winner. Thus,  $f$  does not respect  $R_{C2}$ -Perfection. ■

On the other hand, in the case of the domination relations  $R_P$ ,  $R_{C1}$ , and  $R_{C2}$ , if  $f$  respects the  $R$ -relation, then  $f$  respects  $R$ -Perfection, because they are transitive relations.

Another class of generalizations of the Pareto Domination Relation appears if we relax the idea

of Pareto dominant candidate by a relation based on the number of first positions in the preference profile. A candidate is Pareto dominant if he is in the first position in the preference of all voters. Thus a relaxation of Pareto dominant candidate is a  $q$ -majority of first positions candidate, so that he is favourite for more than  $qm$  voters, where  $q \in [1/2, 1]$ . In the following we only consider the case  $q = 1/2$ .

#### *First Positions Majority Perfection*

This perfection criterion requires the use of the rank matrix to be expressed:

**Definition 13 (First Positions Majority Perfection Criterion,  $R_{FPM}$ ).** We will say that a candidate  $x \in X$  satisfies *First Positions Majority Perfection Criterion* ( $R_{FPM}$ ) in a situation  $(X, p)$  if and only if  $r(x, I) > n/2$ .

This criterion determines that a candidate will be  $R_{FPM}$ -Perfect whenever he appears as the first one in a strict absolute majority of voter preferences. In this sense, a voting procedure  $f$  respects  $R_{FPM}$ -Perfection if it selects such a candidate as the only winner whenever it exists.

#### *3) Improvement Criteria and Actions.*

The search of social election correspondences based on comparison relations requires, not only to establish a binary relation with a perfection criterion, but also to determine what candidate or candidates will be chosen as winners when such criterion is not fulfilled. In our framework, this means to define an improvement criterion or action that allows to select the candidates based on its proximity in terms of this established criterion to the  $R$ -perfect candidate. Thus the winner minimizes the distance to the  $R$ -perfect candidate in the sense that, with the minimum of improvement actions, becomes perfect.

Let  $R$  be a comparison relation or criterion and  $I$  be a type of improvement action. We define a voting method  $f_{(I, R)}$  in the following way:

**Definition 14 ( $f_{(I, R)}$ -Winner Candidate).** Given the situation  $(X, p)$  we say that candidate  $x$  is a  $(I, R)$ -Winner Candidate, or  $x \in f_{(I, R)}(X, p)$ , if  $x$  is the candidate who needs less improvement actions to become a  $R$ -Perfect one.

Intuitively speaking, this definition determines that, given a Perfection Criterion and one distance or improvement action, we can define a voting correspondence that selects the  $R$ -perfect candidate

as the only winner, when such a candidate exists. If there is no such a candidate, it selects as winner all candidates who minimize his distance to perfection, that is to say, who require the minimal number of improvement actions to become  $R$ -perfect.

We have already established the perfection criteria, on reasonable bases of the social election. What distances or improvement actions seem reasonable to define in consonance with these criteria?

Between the multiple improvement criteria that can be selected, we want to emphasize those that find their motivation in participation and monotonicity properties, from the variation of the elements that determine a situation  $(X, p)$ .

This implies that the improvement criteria that we are considering take into account three basic variations:

a) Modifications in the Voters Preferences. As we will see later, we can define an elementary modification in the preference profile and count for every candidate  $x$  how many elementary modifications we need to convert  $x$  in a perfect candidate, and select as a winner those candidate that require a minimal number of modifications.

b) Variations in the set of voters. Another improvement criterion that on a natural way allows transforming a situation  $(X, p)$  with the purpose of obtaining the fulfilment of a perfection criterion is the introduction of new voters (or the elimination of some of them). In this sense, we can count for every candidate  $x$  how many voters (with a specific preferences) we need to add (or delete) to convert  $x$  in a perfect candidate, and select as a winner those candidates that require adding (or deleting) a minimal number of voters.

c) Variations in the set of candidates. Finally we can consider another improvement criterion consistent on the introduction of new candidates dominated on some way, in relation to a particular candidate, or in the elimination of some of them, and then we can determine, like in the variation of the set of voters, the winner candidates.

The improvement criteria based on variations of the set of voters or candidates have some limitations. They are not applicable to all the perfection criteria defined.

In the following and for simplicity we only concentrate in the first case.

#### *Voters Preferences Modifications.*

A first criterion or improvement action is one that, on a neutral and anonymous way, considers the minimum improvement that can be done to a candidate in a preferences profile. That is to say, what have been called above elementary interchange, and that in this context we can define in the following way:

**Definition 15 (Direct Elementary Interchange).**

Given a situation  $(X, p)$  and a candidate  $x$ , we call *direct elementary interchange* in favour of candidate  $x$  in the preferences profile  $p$  to any improvement in the preferences consisting on interchanging the candidate  $x$  position with the one of the immediately preceding candidate in voter preferences of any  $i \in V$ .

Thus, given a Perfection Criterion, if we take as Improvement Criterion to favour the candidates in the voters preferences, we can select as winners all those candidates who require a minimum number of direct elementary interchanges (that is to say, to its favour) to become  $R$ -Perfектs according to the given perfection criterion.

**Definition 16 (Direct Elementary Interchange Criterion,  $I_{DEI}$ ).** Given a situation  $(X, p)$  and a  $R$ -Perfection criterion:

a) We denote by  $i(R, x)$  the minimum number of direct elementary interchanges that allow candidate  $x$  to reach the Perfection Criterion based on relation  $R$ .

b) We define the voting correspondence  $f_{(I, R)}$  where  $I = I_{DEI}$  as follows:

$$f_{(I, R)}(X, p) = \{x \in X : i(R, x) \leq i(R, y), \forall y \in X\}.$$

This definition only considers direct elementary interchanges, that is to say, the measurement used for each candidate consider only the interchanges in which this candidate improves his position in voters preferences and does not consider interchanges that, affecting only other candidates, could help a candidate to reach a perfection criterion.

From this improvement criterion and the perfection criteria that we have defined, we can generate some well-known voting methods.

Thus, the combination of the  $I_{DEI}$  improvement criterion with the respect to the  $R_P$ -Perfection define the following voting correspondence:

$$f_{(I_{DEI}, R_P)}(X, p) = \{x \in X : i(R_P, x) \leq i(R_P, y), \forall y \in X\}.$$

that selects as the only winner any candidate who Pareto-dominates to all the others, whenever it exists, and if none exists, the candidate who requires the smaller number of direct elementary interchanges to become Pareto dominant. And this voting correspondence is the Borda method as shown in the following proposition:

**Proposition 2**

The voting correspondence  $f_{(I, R)}$ , where  $R$  is the  $R_P$ -Perfection criterion and  $I$  is the  $I_{DEI}$ , improvement criterion, is the Borda method.

**Proof:**

Let  $f_{(M, R)} = f_{(I_{DEI}, R_P)}$  and  $(X, p)$  a situation with  $m$  voters, where candidate  $R_P$ -Perfect does not exist. Method  $f_{(I_{DEI}, R_P)}$  selects as winner any candidate  $x$  who becomes  $R_P$ -Perfect in the  $(X, p')$  situation, where profile  $p'$  has been obtained from  $p$  by making  $i(R_P, x)$  direct elementary interchanges (improvements in candidate  $x$  position) so that  $i(R_P, x) \leq i(R_P, y) \forall y \in X$ .

We can relate the whole value of  $i(R_P, x) \geq 0$  (null when candidate  $x$  Pareto dominates the others), with the components of the  $x$  row in the Positions Matrix on the following way:  $i(R_P, x) = \sum_{j=1}^n r(x, j) \cdot (j - 1)$ .

$$\text{Therefore, } f_{(I_{DEI}, R_P)} = \{x \in X : \sum_{j=1}^n r(x, j) \cdot (j - 1) \leq$$

$$\sum_{j=1}^n r(y, j) \cdot (j - 1), \forall y \in X\},$$

$$\text{but } \{x \in X : x \in \arg \min_{x \in X} \sum_{j=1}^n r(x, j) \cdot (j - 1)\} =$$

$$\{x \in X : x \in \arg \max_{x \in X} \sum_{j=1}^n (n - j)r(x, j)\} =$$

$\{x \in X : x \in \arg \max_{x \in X} R_{BORDA}(x)\}$ . So that this method selects as winner any candidate who maximizes the Borda count. Therefore,  $f_{(I_{DEI}, R_P)}$  corresponds with the Borda method. ■

In the same way, the combination of the  $I_{DEI}$  improvement criterion with the  $R_{CI}$ -Perfection defines the following voting correspondence:

$$f_{(I_{DEI}, R_{CI})}(X, p) = \{x \in X : i(R_{CI}, x) \leq i(R_{CI}, y), \forall y \in X\}$$

that is the Dogson method, if we take into account that a  $R_{CI}$ -Perfect candidate is a Condorcet candidate.

Nevertheless, the combination of the  $I_{DEI}$  improvement criterion with the respect to the  $R_{C2}$ -Perfection or  $R_{FPM}$ -Perfection criteria allows defining two new voting correspondences. In the following section we analyze the voting method that appears with the  $R_{FPM}$ -Perfection criterion, which will be called **Dogdson-like Majoritarian (DM)**:

$$f_{(I_{DEI}, R_{FPM})}(X, p) = \{x \in X : i(R_{FPM}, x) \leq i(R_{FPM}, y), \forall y \in X\}.$$

This voting correspondence selects as a winner the candidate that appears as the favourite for more than the half of voters, whenever exists such candidate, and in other case the candidates requiring a minimal number of elementary interchanges in the voter's preferences to become the favourite for at least a half of voters.

In the same form, we can combine the perfection criteria defined with other improvement criteria that we can define varying the candidate or voter's set. A well-known positional method, Plurality method, can be obtained if we use the Pareto Domination as the Perfection criterion and a specific improvement criterion based on deleting a minimal number of voters. In other way, if we use the  $R_{CI}$ -Perfection criterion combined with this improvement criterion based on the elimination of a minimum number of voters, we get the Young method. And if the  $R_{CI}$ -Perfection criterion is implemented with the introduction of new voters (in a specific form and with a restricted preferences) as the improvement criterion, we obtain the Minmax method. For more details, see Jimeno [5].

### B. A second unifying framework

This framework will not require the full preferences profile, but just a summary matrix of the situation, like the comparison matrix or the victories matrix.

Given a set of candidates  $X = \{x_1, x_2, \dots, x_n\}$  and a situation  $(X, p)$ , let  $M_p$  be its comparison matrix, whose entries are  $p(x_i, x_j)$ . The reference here will be the comparison matrices of the unanimous profiles in which all  $m$  voters have the same preferences order. For an order  $O = x_1 x_2 \dots x_n$ , the corresponding comparison matrix will be denoted  $M_{U(O)}$ , and their entries are

$$p_O(x_i, x_j) = \begin{cases} m & \text{if } i < j \\ 0 & \text{if } j < i \end{cases}$$

In other words, if we order the rows and columns of  $M_{U(O)}$  according to  $O$ , every entry above the diagonal

is  $m$  and every entry below the diagonal is 0 (we can say that every diagonal entry is 0 by convention).

We will construct voting correspondences by mean of distances or, more generally, pseudodistances between comparison matrices, in the following way:

**Let  $d$  be a pseudodistance function defined for every two comparison matrices whose rows and columns are ordered according to the same linear order of candidates. Then, we define  $f_d(X, p) = \{x \in X : \text{there exists an order } O = x_1 x_2 \dots x_n \text{ with } x = x_1 \text{ such that } d(M_p, M_{U(O)}) \leq d(M_p, M_{U(O')}) \text{ for every order } O'\}$ .**

**That is to say, the correspondence  $f_d$  declares as a winner of a given situation  $(X, p)$  to every candidate who is at the top of some order  $O$  such that the distance  $d(M_p, M_{U(O)})$  is minimal.**

Therefore, every pseudodistance that can be defined between  $n \times n$  matrices produces a voting correspondence which, in some way, transports to the voting problem the specificities of the pseudodistance formula. The use of a distance will be required to produce voting methods whose primary output is a ranking of candidates, while the use of a pseudodistance will be enough to produce voting methods whose primary output is a set of winning candidates

The main particular cases of voting correspondences obtained in this way are the Kemeny and Borda correspondences, both produced using the 1-norm distance.

Kemeny is the correspondence  $f_d$  when  $d$  is the 1-norm distance  $d(M_p, M_{U(O)}) = \sum_{i=1}^n \sum_{j=1}^n |p(x_i, x_j) - p_O(x_i, x_j)|$ , where both matrices are ordered according to  $O = x_1 x_2 \dots x_n$ . Indeed, since  $\sum_{i=1}^n \sum_{j=1}^n |p(x_i, x_j) - p_O(x_i, x_j)| = 2 \sum_{i>j} (m - p(x_i, x_j)) = 2 \sum_{i>j} (m - K(O))$ ,  $d(M_p, M_{U(O)})$  is minimal when the

Kemeny score of order  $O$ ,  $K(O)$ , is maximal.

Borda is the correspondence  $f_d$  when  $d$  is the pseudodistance defined (from the 1-norm distance)

as  $d(M_p, M_{U(O)}) = \sum_{j=1}^n |p(x_1, x_j) - p_O(x_1, x_j)|$ , where both matrices are ordered according to  $O$ . Since  $\sum_{j=1}^n |p(x_1, x_j) - p_O(x_1, x_j)| = \sum_{j=1}^n (m - p(x_1, x_j)) =$

$nm - \sum_{j=1}^n p(x_1, x_j)$ ,  $d(M_p, M_{U(O)})$  is minimal when the Borda score of  $x_1$ , the first candidate in  $O$ , is maximal.

If in these two cases, we replace the comparison matrix  $M_p$  with the victories matrix  $V_p$ , we will obtain, respectively, the Slater and Copeland correspondences.

Another well-known voting correspondence obtained in this way is the Simpson-Cramer Minmax correspondence. It is the correspondence  $f_d$  when  $d$  is the pseudodistance defined (from the  $\infty$ -norm distance) as  $d(M_p, M_{U(O)}) = \max_{j=1,2,\dots,n} |p(x_1, x_j) - p_O(x_1, x_j)|$ , where both matrices

are ordered according to  $O$ . Indeed, since  $\max_{j=1,2,\dots,n} |p(x_1, x_j) - p_O(x_1, x_j)| =$

$$\max_{j=1,2,\dots,n} (m - p(x_1, x_j)) = m - \min_{j=1,2,\dots,n} p(x_1, x_j),$$

$d(M_p, M_{U(O)})$  is minimal when the minimal entry in the row of  $x_1$ , the first candidate in  $O$ , is maximal.

It is worth to explore the consequences of choosing other distances or pseudodistances, analysing the correspondences so produced. If we use distances applying to every entry of the matrices (and thus to every component of  $\mathbf{R}^{nxn}$ ), we will get Kemeny-like correspondences, while if we use pseudodistances applying only to the first row of the matrices (and thus to just the first  $n$  components of  $\mathbf{R}^{nxn}$ ), we will get Borda-like or Minmax-like correspondences.

In this paper we will explore a correspondence in an intermediate position between Borda and Minmax. It is  $f_{d(\rho)}$  when  $d(\rho)$  is the pseudodistance defined, from the  $\rho$ -norm distance,

$$d(\rho)(M_p, M_{U(O)}) = \left( \sum_{j=1}^n |p(x_1, x_j) - p_O(x_1, x_j)|^\rho \right)^{\frac{1}{\rho}}$$

where both matrices are ordered according to  $O = x_1 x_2 \dots x_n$ , and being  $nm \ll \rho < \infty$  (thus,  $\rho$  is a big positive number as compared to  $nm$ ).

The following proposition give us an equivalent definition of  $f_{d(\rho)}$ , which has a much simpler and easy to use expression.

### Proposition 3

Given a situation  $(X, p)$ , with  $X = \{x_1, x_2, \dots, x_n\}$  and comparison matrix  $M_p$ . For every  $x_j$ , let  $q(x_j) = (q_{j1}, q_{j2}, \dots, q_{jn-1})$  be the vector formed by the

nondiagonal row entries  $p(x_j, x_r)$  of  $x_j$  ordered in a nondecreasing way. Then, candidate  $x_i$  is a winner of  $f_{d(\rho)}(X, p)$  if and only if for every other candidate  $x_k$ ,  $q(x_i) \geq_{\text{LEX}} q(x_k)$  (that is to say,  $q_{i1} > q_{k1}$  or ( $q_{i1} = q_{k1}$  and  $q_{i2} > q_{k2}$ ) or ( $q_{i1} = q_{k1}$  and  $q_{i2} = q_{k2}$  and  $q_{i3} > q_{k3}$ ) or ...).

### Proof:

Let us first prove the following intermediate result:

Given two vectors  $\mathbf{a} = (a_1, a_2, \dots, a_{n-1})$  and  $\mathbf{a}' = (a'_1, a'_2, \dots, a'_{n-1})$ , with positive integer entries nonincreasingly ordered and bounded from above by  $m$  ( $m \geq a_i \geq a_{i+1}$  and  $m \geq a'_i \geq a'_{i+1} \forall i=1,\dots,n-2$ ), and being  $\rho$  a high enough positive number,  $\|\mathbf{a}\|_\rho = (a_1^\rho + a_2^\rho + \dots + a_{n-1}^\rho)^{1/\rho} \geq \|\mathbf{a}'\|_\rho = (a'_1^\rho + a'_2^\rho + \dots + a'_{n-1}^\rho)^{1/\rho}$  if and only if  $\mathbf{a} \geq_{\text{LEX}} \mathbf{a}'$ .

To prove this result, let us compute  $\lim_{p \rightarrow \infty} (\|\mathbf{b}\|_p - \|\mathbf{b}'\|_p)$  for the extreme particular case in which  $\mathbf{b} = (b_1, 0, \dots, 0)$  and  $\mathbf{b}' = (b_1-r, b_1-r, \dots, b_1-r)$  where  $r \geq 1$ .

$$\begin{aligned} \lim_{p \rightarrow \infty} (\|\mathbf{b}\|_p - \|\mathbf{b}'\|_p) &= \lim_{p \rightarrow \infty} ((b_1^p)^{\frac{1}{p}} - ((n-1)(b_1-r)^p)^{\frac{1}{p}}) = \\ \lim_{p \rightarrow \infty} ((b_1) - (n-1)^p(b_1-r)) &= b_1 - (b_1-r) \lim_{p \rightarrow \infty} (n-1)^{\frac{1}{p}} = \\ b_1 - (b_1-r) &= r \geq 1. \end{aligned}$$

Now consider the general case for  $\mathbf{a}, \mathbf{a}'$  in which  $a'_1 = a_1-r$ .  $\lim_{p \rightarrow \infty} (\|\mathbf{a}\|_p - \|\mathbf{a}'\|_p) \geq \lim_{p \rightarrow \infty} (\|\mathbf{b}\|_p - \|\mathbf{b}'\|_p) \geq \lim_{p \rightarrow \infty} (\|\mathbf{b}\|_p - \|\mathbf{b}'\|_p) \geq 1$ . Therefore, there is a real number  $\alpha_n$ , depending on  $n$ , such that for every  $\rho \geq \alpha_n$ ,  $\|\mathbf{a}\|_\rho > \|\mathbf{a}'\|_\rho$ . This argument is also valid for the the general case for  $\mathbf{a}, \mathbf{a}'$  in which  $a'_1 = a_1, \dots, a'_h = a_h$ , and  $a'_{h+1} = a_{h+1}-r$ , where  $1 < h \leq n-1$ .

In order to prove now proposition 3, let  $\rho$  be a positive number such that  $\rho \geq \alpha_n$ , and let us compare  $d(\rho)(M_p, M_{U(O)})$  where  $x_i$  is at the top of the order  $Oi$  with  $d(\rho)(M_p, M_{U(Ok)})$  where  $x_k$  is at the top of the order  $Ok$ .

$$\begin{aligned} d(\rho)(M_p, M_{U(O)}) &= \left( \sum_{j=1}^n |p(x_i, x_j) - p_{Oi}(x_i, x_j)|^\rho \right)^{\frac{1}{\rho}} \\ &= \left( \sum_{j=1, j \neq i}^n (m - p(x_i, x_j))^\rho \right)^{\frac{1}{\rho}} \\ d(\rho)(M_p, M_{U(Ok)}) &= \left( \sum_{j=1}^n |p(x_k, x_j) - p_{Ok}(x_k, x_j)|^\rho \right)^{\frac{1}{\rho}} \\ &= \left( \sum_{j=1, j \neq k}^n (m - p(x_k, x_j))^\rho \right)^{\frac{1}{\rho}}. \end{aligned}$$

Let us call  $\mathbf{D}_i = (D_{i1}, D_{i2}, \dots, D_{in-1})$  and  $\mathbf{D}_k = (D_{k1}, D_{k2}, \dots, D_{kn-1})$  the vectors  $(m - p(x_i, x_j))_{j=1,2,\dots,n, j \neq i}$  and  $(m - p(x_k, x_j))_{j=1,2,\dots,n, j \neq k}$ , after permuting their entries in such a way that they are nonincreasingly ordered. Since  $d(p)(M_p, M_{U(Oi)}) = \|\mathbf{D}_i\|_p$  and  $d(p)(M_p, M_{U(Ok)}) = \|\mathbf{D}_k\|_p$ , by the intermediate result just proved,  $d(p)(M_p, M_{U(Oi)}) > d(p)(M_p, M_{U(Ok)})$  if and only if  $\mathbf{D}_i \geq_{LEX} \mathbf{D}_k$ , which, since  $\text{Max } \{m - s_h\}_{h \in H} = m - \text{Min } \{s_h\}_{h \in H}$  is equivalent to say  $q(x_i) \geq_{LEX} q(x_k)$ . ■

Observe that we have obtained a correspondence which is the more natural refinement of Minmax, so we will call it the **Natural Lexicografic Refinement of Minmax**, abbreviated **NLRM**.

#### IV. ANALYSIS OF TWO VOTING CORRESPONDENCES

##### A. Dogdson-like Majoritarian (DM)

In the previous section we have defined this new voting correspondence as a voting method that selects as a winner the candidate who needs a smaller number of elementary interchanges to become perfect.

The following proposition proves that the DM correspondence satisfies Semi-strict Monotonicity (although not Strict Monotonicity), but not Positive Involvement and Young's Consistence properties, nor Condorcet Criterion.

##### Proposition 4

The voting correspondence DM:

- a) Satisfies Semi-Strict, but not Strict, Monotonicity.
- b) Fails to satisfy Positive Involvement.
- c) Fails to satisfy the Condorcet Criterion.
- d) Fails to correspond with a positional method, because fails to satisfy Young's Consistency property.

##### Proof:

a) Let  $f_{(I, R)} = f_{(I_{IED}, R_{FPM})}$ , and  $(X, p)$  a situation in which does not exist a  $R_{FPM}$ -perfect candidate. The correspondence chooses as winners those candidates who become perfects in the situation  $(X, p')$ , where  $p'$  has been obtained from profile  $p$  when making  $i(R_{FPM}, X)$  direct elementary interchanges, so that  $i(R_{FPM}, x) \leq i(R_{FPM}, y), \forall y \in X$ .

Because,  $i(R_{FPM}, x) = \min_{y \in X} i(R_{FPM}, y), \forall y \in X$ , if we improve the candidate  $x$  position by means of an elementary interchange, then if we denote by  $i'(R_{FPM}, x)$  the number of elementary interchanges needed to convert  $x$  in a  $R_{FPM}$ -Perfect candidate in the new situation,  $i'(R_{FPM}, x) \leq i(R_{FPM}, x)$  and on the other hand,  $i'(R_{FPM}, y) \geq i(R_{FPM}, y) \forall y \in X/x$ . Therefore,  $i'(R_{FPM}, x) \leq \min_{y \in X} i(R_{FPM}, y), \forall y \in X \leq i(R_{FPM}, y) \forall y \in X/x \leq i'(R_{FPM}, y) \forall y \in X/x$ , with which  $x$  continues being chosen when improving their position in the new profile, so that  $f_{(I_{IED}, R_{FPM})}$  satisfies the Monotonicity property.

Additionally, if a candidate  $z$  is not a winner in the situation  $(X, p)$  then  $i(R_{FPM}, z) > i(R_{FPM}, x) = \min_{y \in X} i(R_{FPM}, y), \forall y \in X$ , and when we improve the candidate  $x$  position by means of an elementary interchange, we obtain that  $i'(R_{FPM}, z) \geq i(R_{FPM}, z) > i(R_{FPM}, x) \geq i'(R_{FPM}, x)$ . This implies that if  $z \notin f_{(I_{IED}, R_{FPM})}$  then  $z \notin f(X, p')$ , so that  $f_{(I_{IED}, R_{FPM})}$  satisfies the Semi-Strict Monotonicity property.

It fails to satisfy strict monotonicity because in the situation described in (c) the DM winners are  $x_1$  and  $x_3$  (both of them need only one elementary interchange to become  $R_{FPM}$ ) and improving  $x_1$  in the last winner does not lead to the election of  $x_1$  as the only winner, because  $x_1$  and  $x_3$  remain as winners (both of them need again only one elementary interchange to become  $R_{FPM}$ )

b) Let  $X = \{x_1, x_2, x_3, x_4, x_5, x_6, x_7\}, m = 4$  and  $p$  the following profile [ $x_2x_7x_1x_4x_3x_5x_6$  (1 voter),  $x_2x_1x_4x_3x_5x_6x_7$  (1 voter),  $x_7x_5x_1x_4x_3x_6x_2$  (1 voter) and  $x_6x_3x_5x_4x_1x_7x_2$  (1 voter)].

The winner of DM method is  $x_1$ , because he only needs 5 elementary interchanges and  $x_2$  y  $x_7$  need 6 elementary interchanges. But if in the preference profile  $p$  we add a new voter with preferences  $v: x_1x_2x_6x_3x_4x_7x_5$ , the new winner is  $x_2$  (only needs 1 elementary interchange in the new voter preferences, and the winner in the profile  $p$ , candidate  $x_1$ , needs 3 elementary interchanges in the profile  $p+v$ ). DM method fails to satisfy Positive Involvement.

c) Let  $X = \{x_1, x_2, x_3, x_4\}, m = 5$  and  $p$  the following profile [ $x_3x_1x_2x_4$  (1 voter),  $x_1x_2x_3x_4$  (1 voter),  $x_1x_3x_4x_2$  (1 voter),  $x_4x_3x_2x_1$  (1 voter) and  $x_3x_2x_4x_1$  (1 voter)].

The Comparison and Rank Matrices are:

TABLE I COMPARISON MATRIX				
	$x_1$	$x_2$	$x_3$	$x_4$

$x_1$	3	2	3
$x_2$	2	1	3
$x_3$	3	4	4
$x_4$	2	2	1

TABLE II  
RANK MATRIX

	$1^o$	$2^o$	$3^o$	$4^o$
$x_1$	2	1	0	2
$x_2$	0	2	2	1
$x_3$	2	2	1	0
$x_4$	1	0	2	2

The  $x_3$  candidate is the Condorcet winner, and nevertheless the DM winners are  $x_1$  and  $x_3$ , both of them need only one elementary interchange to become  $R_{FPM}$ -Perfects candidates.

So, we have demonstrated that this correspondence fails to fulfil the Condorcet Criterion.

d) Let  $X = \{x_1, x_2, x_3, x_4\}$ ,  $m = 4$  and  $p$  the following profile  $[x_3x_2x_1x_4$  (1 voter),  $x_1x_2x_3x_4$  (1 voter),  $x_1x_2x_4x_3$  (1 voter),  $x_3x_2x_4x_1$  (1 voter)]

The winning candidates are  $x_1$  and  $x_3$ , both of them need an only elementary interchange to get a first positions majority.

Let us add to the previous profile new  $m' = 4$  voters with the following profile  $p'$ :  $[x_3x_2x_1x_4$  (1 voter),  $x_1x_2x_4x_3$  (2 voters),  $x_3x_2x_4x_1$  (1 voter)].

In this new profile  $x_1$  is the one and only winner because it is the candidate who needs the minimum number of elementary interchanges (two) to become winner.

If we add the two profiles in  $p'' = p + p'$ , the winners are  $\{x_1, x_3\}$ , so we demonstrate that the External Consistency property is failed to fulfil, and so DM is not a positional voting correspondence. ■

#### B. Natural Lexicographic Refinement of Minmax (NLRM).

As we will prove in Proposition 5, the NLRM correspondence satisfies the Condorcet Criterion and also all Monotonicity and Positive Involvement properties, but not the Optimist Participation properties.

#### Proposition 5

The voting correspondence NLRM:

- a) Satisfies Strict Monotonicity.
- b) Satisfies Strict Positive Involvement.

- c) Satisfies the Condorcet Criterion.
- d) Fails to satisfy Optimist Participation.

#### Proof:

a) Let  $x$  be a winner for the situation  $(X, p)$ . This means that, for every other candidate  $y$ ,  $q(x, p) \geq_{LEX} q(y, p)$ , where  $q(z, p)$  is the vector formed by the nondiagonal row entries of  $z$  in a nondecreasing order. If we make an elementary interchange favourable to  $x$ , in the new situation  $(X, p'_x)$  the vector  $q(x, p'_x)$  is the result of adding 1 to a component of  $q(x, p)$ . It is obvious that  $x$  will be the only winner for the new situation  $(X, p'_x)$ .

b) Let  $x$  be a winner for  $(X, p)$ , so  $q(x, p) \geq_{LEX} q(y, p)$  for every other candidate  $y$ . If we add a new voter  $v$  for whom  $x$  is the favourite, in the new situation

$(X, p+v)$  the vector  $q(x, p+v)$  is the result of adding 1 to every component of  $q(x, p)$ . It is obvious that  $x$  will be the only winner for the new situation  $(X, p+v)$ .

c) When there is a Condorcet winner, Minmax chooses only this candidate, and, since NLRM is a refinement of Minmax, it has also to choose only this candidate.

d) NLRM does not satisfy Optimist Participation because it satisfies the Condorcet Criterion and, as proved in Jimeno et al [6], no Condorcet voting correspondence satisfies Optimist Participation. ■

TABLE III  
METHODS AND PROPERTIES

	Plurality	DM	Minmax	NLRM
<i>Strict Monotonicity</i>	0	0	0	1
<i>Semi-Strict Monotonicity</i>	1	1	1	1
<i>Monotonicity</i>	1	1	1	1
<i>Strict Positive Involvement</i>	1	0	0	1
<i>Semi-Strict Positive Involvement</i>	1	0	1	1
<i>Positive Involvement</i>	1	0	1	1
<i>Strict Optimist Participation</i>	0	0	0	0
<i>Semi-Strict Optimist Participation</i>	1	0	0	0
<i>Optimist Participation</i>	1	0	0	0

Table III resumes the fulfilment of the monotonicity and participation properties by Plurality and Minmax known methods, and the two new ones defined in this paper.

## V. FINAL REMARKS AND CONCLUSIONS

In this paper we have described two unifying frameworks, one located in the context of the preference profiles and based on the idea of distance, through improvement actions, to a perfect candidate, and the other located in the context of the comparison matrices and based on the idea of distance to the comparison matrices corresponding to unanimous profiles. Between the many possibilities that each of these schemes offered to us, we have chosen a voting method from each of them and we have analyzed them from the point of view of their theoretical properties, especially those of monotonicity and participation.

The first of these voting methods, that we have called DM, has certain similarities with positional voting methods. Unfortunately, and as it is indicated in Proposition 4 and Table III, it does not have better properties than those already known for these methods, and it even gets worse in terms of properties of participation as opposed to some of the most outstanding as Borda and Plurality.

The second one, called NLRM, is very similar to the well-known Minmax method (of which it is a refinement). Indeed, it is because of this fact, that some of their good properties are derived. In fact, their monotonicity and participation properties are even better (at least from the point of view of its resolutivity) than those of Minmax, as shown in Proposition 5 and Table III.

Although we are not aware of any practical application of this or other related methods, like Minmax, we consider that it is possible to take advantage of this method in some contexts, especially those with a small number of candidates or alternatives where the voters are very familiar with them.

However, our primary objective has not been only to obtain new methods with good properties (objective that still remains open), but also to analyze the unifying schemes and to illustrate its possible uses. It is left for future investigations the search of new concepts of perfection and improvement, new distances and pseudodistances, and the exhaustive analysis of those ones already defined.

## ACKNOWLEDGEMENTS

This research has been supported by the

Research Projects SEC 2001-1186 and SEJ 2004-07875 of Spanish Ministerio de Ciencia y Tecnología.

## REFERENCES

- [1] K. J. Arrow and H. Raynaud, *Social choice and multicriterion decisionmaking*, MIT Press, Cambridge, 1986.
- [2] J. P. Barthélémy and B. Monjardet, "The median procedure in cluster analysis and social choice theory", in *Mathematical Social Sciences*, vol. 1, pp. 235-267, 1981.
- [3] D. Bouyssou, Th. Marchant and P. Perny, "Théorie du choix social et aide multicritère à la décision", working paper, 2003.
- [4] D. E. Campbell and S. Nitzan, "Social Compromise and Social Metrics", *Social Choice and Welfare*, vol. 3, pp 1-16, 1986.
- [5] J.L. Jimeno, "Propiedades de Participación en los Métodos de Agregación de Preferencias", Unpublished Doctoral Thesis, Universidad de Alcalá, Spain, 2003.
- [6] J.L. Jimeno, J. Pérez and E. García, "Some results concerning No Show Paradoxes", communication to the XXVIII Simposio de Análisis Económico, Sevilla (Spain), 2003.
- [7] E. Lerer and S. Nitzan, "Some General Results on the Metric Rationalization for Social Decision Rules", *Journal of Economic Theory*, vol. 37, pp 191-201, 1985.
- [8] T. Marchant, "Towards a theory of mcdm; stepping away from social choice theory", *Mathematical Social Sciences* 45, 343-363, 2003.
- [9] J. F. Marcotorchino and P. Michaud, *Optimization en analyse ordinaire des données*, Masson, Paris, 1979.
- [10] H. Moulin, "Condorcet's Principle Implies the No Show Paradox". *Journal of Economic Theory*, vol. 45, pp 53-64, 1988.
- [11] J. Pérez, "Propiedades de consistencia en los métodos de la Decisión Multicriterio Discreta", Unpublished Doctoral Thesis, Universidad de Alcalá, Spain, 1991.
- [12] J. Pérez, "Theoretical elements of comparison among ordinal discrete multicriteria methods", *Journal of Multi-Criteria Decision Analysis* 3, 157-176, 1994.
- [13] J. Pérez, "The strong No Show Paradoxes are a common flaw in Condorcet voting correspondences", *Social Choice and Welfare* vol. 18, pp 601-616, 2001.
- [14] J. Pérez and S. Barba-Romero, "Three practical criteria of comparison among ordinal preference aggregating rules", *European Journal of Operational Research* 85, 473-487, 1994.
- [15] J.-Ch. Pomerol and S. Barba-Romero, *Multicriterion Decision in Management, Principles and Practice*, Kluwer, Dordrecht, 2000.
- [16] H.P. Young, "An axiomatization of Borda's rule", *Journal of Economic Theory* vol. 9, pp 43-52, 1974.

# Real time management of a metro rail terminus

Marta Flamini\* and Dario Pacciarelli\*

\*Dipartimento di Informatica e Automazione, Università Roma Tre

Via Della Vasca Navale, 79, I-00146 Roma Italy

Email: flamini@dia.uniroma3.it, pacciarelli@dia.uniroma3.it

**Abstract**—This paper addresses a scheduling problem arising in the real time management of a metro rail terminus. It mainly consists in routing incoming trains through the station and scheduling their departures with the objective of optimizing punctuality and regularity of train service. This task is presently carried out by a human operator, called local area manager. The purpose of this work is to develop an automated train traffic control system, able to directly implement most traffic control actions, without the authorization of the local area manager. To this aim, a detailed optimization model is necessary, in order to guarantee that a solution, which is feasible for the optimization model, is always also physically feasible. The scheduling problem is modeled as a bicriteria job shop scheduling problem with blocking constraints, in which the two objective functions, in lexicographical order, are the minimization of tardiness/earliness and the headway optimization. The problem is solved in two steps. In the first step a fast heuristic builds a feasible solution by considering the first objective function. In the second step the regularity is optimized under the constraint that the first objective function does not deteriorate. Computational results carried out on a real case shows that the system is able to manage the terminus very efficiently and effectively.

**Keywords**—Train scheduling, real time, job shop scheduling, blocking, bicriteria optimization.

## I. INTRODUCTION

RAILWAY traffic optimization is experiencing an increasing interest both among researchers and practitioners. Solving problems of practical interest in this field requires using detailed models, able to represent real and different railway traffic situations, and developing efficient algorithms to be used as decision support system in traffic control operation. Railway scheduling problems have been studied by using different techniques, including linear programming, integer or non-linear programming, graph theory and dynamic programming. Among the published results, we cite the papers by Dorfman and Medanic [5], Adenso-Diaz *et al.* [1], Cai *et al.* [3], Higgins *et al.* [6], Şahin [10] and the survey paper of Cordeau *et al.* [4].

This paper deals with the real time management of a metro rail terminus. The management and control of rail operations is usually based on off-line generated timetables for every train, and consists in operating in real time with strict adherence to these timetables. When dealing in particular with a metro rail terminus, it mainly consists in routing incoming trains through the station and scheduling their departures with strict adherence to the off-line timetable. However, when incoming trains are heavily delayed with respect to the off-line timetable, it is necessary to reschedule their departure times in order to provide service continuity and punctuality as much as possible. To a large extent, this task is carried out by human operators all over the world. A local area manager is in charge of setting routes and scheduling train departures with the objective of pursuing punctuality and regularity of the train service as much as possible. Computer support, when available, consists in most cases of a control panel describing the current situation of the network. On the other hand, there are several attempts to develop computerized decision support systems allowing a more efficient and easier management process [1], [5], [10].

Railway traffic control is particularly important in the management of metropolitan rail networks, where the problem is made more difficult because of the smaller area available and heavy traffic conditions. In this paper we report on the implementation of scheduling algorithms for a real time Train Management System (TMS), able to route and schedule train movements through an underground line terminus. We report in particular on the results of a research project on the management of rail traffic at an underground metro rail terminus, in Italy. The scheduling algorithm developed within the project produces a plan of movements for all trains circulating in the terminus, with the objective of optimizing punctuality and regularity of the train service. One aim of the project is to move a step further in the direction of automating the train traffic control process, by enabling the TMS to implement most traffic control actions, without the authorization of the local area manager. To this aim, detailed optimization models are necessary, in order

to guarantee that a solution, which is feasible for the optimization model, is always also physically feasible.

Since the punctuality is more important than the regularity of train service, in this paper the two objective functions are considered in lexicographical order. An alternative model for the problem would be to consider the two objective functions jointly. This would make possible comparing different non-dominated solutions and then choosing a better compromise solution. However, punctuality is considered more important than regularity by railway practitioners, since respecting the off-line timetable would imply automatically respecting the regularity of train service. Hence, in this paper, a plan of movements is developed by first considering the punctuality function only. Then, the plan is improved by optimizing the regularity of train service, without affecting the punctuality objective function. More precisely, the overall decision problem is approached by dividing it into a routing/sequencing problem and a scheduling problem. The routing/sequencing problem consists in assigning a path to each train in the station and in solving conflicts among trains by sequencing their movements with the objective of optimizing train punctuality. We model this problem by means of an alternative graph [7], adapting it to deal with some specific constraints arising in the rail terminus. The problem is solved with a fast heuristic, able to find a feasible solution within the strict time limits necessary for real time purposes. The scheduling problem consists then in defining an exact timing for train movements, without changing the routing and the sequencing, with the objective of improving the regularity of the service in terms of headway, and without degrading train punctuality. We show that this second problem can be solved at optimality with a polynomial time algorithm.

## II. PROBLEM DESCRIPTION: ROUTING/SEQUENCING

The problem consists in defining a schedule in real time for all the trains circulating in the terminus, plus a given number of incoming trains. In particular, for each train it is necessary to define a routing in the terminus, and a departure time from the terminus in a given time horizon. The routing problem consists in assigning a path to each train from its current position, or from the station entrance, to the terminus exit point.

Figure 1 describes a typical example of an Italian metro rail terminus. The terminus is divided into “blocks” of different length, a block being a track segment between two signals. Within the station a signal may turn into two colors, say red or green. A red signal means that the subsequent block is not available, e.g. occupied by another train. A green signal means that

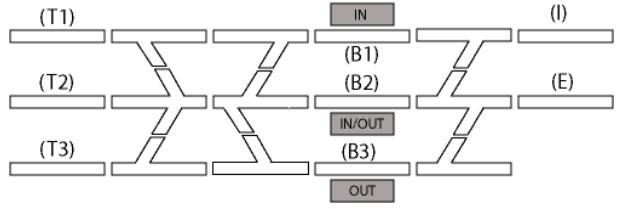


Fig. 1. The metro rail terminus

the subsequent block section is empty and available. A train is allowed to enter a block only if the signal is green. Hence, each block can host at most one train at a time. Besides this constraint, there are special sequence of blocks, called “routes” that are considered as single resources of the network. More precisely, a route  $R_i$  is a sequence of consecutive blocks, and railway safety rules impose that when a train traverses any block of route  $R_i$ , no other train is allowed to enter any route  $R_j$  having at least one block in common with route  $R_i$ . Route  $R_j$  is called “incompatible” with  $R_i$ . The combinatorial structure of the train scheduling problem is therefore similar to that of blocking job shop scheduling problem, a block corresponding to a blocking machine, and a train corresponding to a job (see, e.g., Mascis and Pacciarelli [7]).

In order to formulate the problem we distinguish two sets of resources: a first set is composed by the following blocks in the terminus:

- the platforms, arrival (B1), exit (B3), both arrival and exit (B2), where passengers can get in or out of the train,
- the final tracks (T1, T2, T3), railroad ending blocks, beyond the platforms,
- one arrival point to the terminus (I),
- one exit point from the terminus (U).

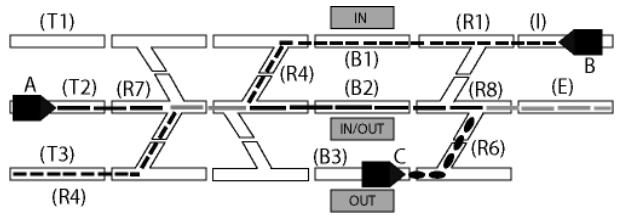


Fig. 2. Incompatibilities between different routes

The second set is the set of available routes: each route goes from one block to another of the previously defined set, and the two blocks do not belong to the route. In the terminus of figure 1 the possible routes are: R1(from block I to B1, or I-B1), R2(B1-T1), R3(B1-T2), R4(B1-T3), R5(T3-B3), R6(B3-E), R7(T2-B2), R8(B2-

E), R9(T3-B2), R10(T1-B2), R11(I-B2), R12(T1-B3), R13(T2-B3). A path for a train is therefore a set of consecutive routes going from its current position to the terminus exit point. A path for a train arriving at the terminus must therefore depart from block I. Then, for example, the train can move through route R1 (from I to B1) and stop in platform B1, where passengers can leave the train. Then the train can move through a different route, say R2, to the end block T1 and go back to the exit platform B2, board the passengers, and finally reach the terminus exit point E: in this case the train covers the sequence of routes R1, R2, R10, R8. When a path is assigned to a train, the scheduling phase consists in assigning to each train a set of consecutive time windows, associated with the traversing of each route of the train path. A schedule is feasible if incompatible time windows do not overlap. Incompatibility constraints can be expressed as follows: a train can enter a route  $R$  if and only if: (i) it has completed traversing the previous route in its path, (ii) no other train is traversing a route incompatible with  $R$ . Preemption is not allowed when traversing a route. Introducing suitable precedence constraints can solve each conflict deriving from the traversing of two incompatible routes.

In the example of Figure 2, to train  $A$  are assigned the routes  $R7$ , and  $R8$ , to train  $B$  the routes  $R1$ , and  $R4$ , and finally to train  $C$  is assigned the route  $R6$ . In this case we have the following pairs of incompatible routes:  $(R7(A), R4(B))$  and  $(R8(A), R6(C))$ . Solving the two conflicts corresponds therefore to define a precedence between the traversing of route  $R7$  for  $A$  and the traversing of route  $R4$  for  $B$ , and a precedence between the traversing of route  $R8$  for  $A$  and the traversing of route  $R6$  for  $C$ .

#### A. Objective functions

We consider two different objective functions in lexicographical order: the first one is the minimization of the sum of total tardiness plus total earliness for all trains with respect to the off-line timetable. In particular, the off-line timetable specifies a departure time for each train from each station. Let  $O^i$  be the off-line departure time for train  $i$ . A punctuality time window  $[O^i, O^i + m_p]$  is associated to train  $i$ , where  $m_p$  is called the *punctuality margin* for the train. The train is then considered tardy [early] if it leaves the station after  $O^i + m_p$  [before  $O^i$ ]. If train  $i$  leaves the station at time  $t_i$ , its contribution to the objective function is then the quantity:

$$D_i = \begin{cases} 0 & \text{if } O^i \leq t_i \leq O^i + m_p \\ t_i - (O^i + m_p) & \text{if } t_i \geq (O^i + m_p) \\ (O^i) - t_i & \text{if } t_i \leq (O^i) \end{cases} \quad (1)$$

The first objective function is then  $\sum_{i=1}^k D_i = \sum_{i=1}^k \max\{0; t_i - (O^i + m_p); (O^i) - t_i\}$ , where  $k$  is the number of trains to schedule.

The second objective function we consider is the minimization of the difference between the off-line headway and the actual headway for all pairs of consecutive trains leaving the station. The purpose of this objective function is the improvement of the service regularity in terms of headway. In other words, provided that a train can leave the station within the punctuality time window, the exact departure time is computed by optimizing the headway.

### III. SEQUENCING MODELS

The sequencing problem described in this paper has been modeled as a blocking job shop problem by using the alternative graph model of Mascis and Pacciarelli [7], a generalization of the disjunctive graph of Roy and Sussman [9]. With this model a set of jobs (trains) compete for the usage of a set of resources (routes). Each job has an ordered list of resources to request, which is represented by a chain. A node in the chain is called an *operation*, and is associated with the usage of a particular resource for a job. An arc  $(i, j)$ , between two consecutive nodes  $i$  and  $j$  in a chain, is called *fixed arc*, and it represents a precedence constraint, the arc weight  $p_{ij}$  indicating the processing time of operation  $i$ . The job cannot start operation  $j$  before it completes processing  $i$ .

Since incompatible resources cannot be used at the same time, whenever two jobs require two incompatible resources, there is a potential conflict. In this case, a processing order must be defined between the incompatible operations, and we model it by introducing in the graph a suitable pair of *alternative arcs*. Each alternative arc models a possible precedence between two operations.

The scheduling problem can be therefore formulated as a particular *disjunctive program*, i.e. a linear program with logical conditions involving operation “or” ( $\vee$ , disjunction), as in Balas (1979). By denoting with  $S, n_1, \dots, n_n, F$  the set of all operations to be scheduled, i.e. all the nodes of the chains associated to the trains, the variables of the problem are the starting times  $t_1, \dots, t_n, t_F$  of operations  $n_1, \dots, n_n, F$ , respectively, while operation  $S$  is a dummy operation associated to time zero. Also  $F$  is a dummy operation, associated with the completion of all other operations.

In this paper, we define  $n_{o_k}$  the operation associated with the departure time of train  $k$  from the station, and we call  $O^k$  its *due date*, equal to the off-line departure time of train  $k = 1, \dots, N$ . We also call  $F$  the set of all fixed precedence relations,  $A$  the set of all pairs of alternative precedence relations,  $N$  the number of trains to be scheduled, and  $\{n_{o_1}, n_{o_2}, \dots, n_{o_N}\}$  the nodes associated to the departure time of some train from the station. Hence, the problem consists in assigning values to  $t_1, \dots, t_n$  such that all fixed precedence relations, and exactly one for each pair of the alternative precedence relations, are satisfied.

### Problem 3.1:

$$\begin{aligned} & \min \sum_{h=1, \dots, N} \max\{0; t_{o_h} - O^h - m_p; O^h - t_{o_h}\} \\ & \text{s.t.} \\ & \left\{ \begin{array}{l} t_j - t_i \geq p_{ij} \quad (i, j) \in F \\ (t_j - t_i \geq a_{ij}) \vee (t_k - t_h \geq a_{hk}) \quad ((i, j), (h, k)) \in A \end{array} \right. \end{aligned}$$

A feasible solution to problem 3.1 consists in solving each possible conflict among trains. In terms of alternative graph formulation it corresponds to selecting an arc for each alternative pair in such a way that the resulting graph has no cycles. In fact, a cycle represents an operation preceding itself, which is not feasible.

The graph composed by all fixed arcs plus the selected alternative arcs is called a *precedence graph*. Given an acyclic precedence graph, the length  $l(S, i)$  of the longest path from the dummy node  $S$  to any other node  $n_i$  gives then the earliest starting time  $t_i = l(S, i)$  of operation  $n_i$  in the feasible solution, i.e. the time at which the train associated to the node enters the associated resource in the terminus. We call this quantity the *head*  $t_i = l(S, i)$  of node  $i$ .

We next describe the construction of the alternative graph implemented in our algorithm. In the routing phase, a sequence of resources/routes is assigned to each job/train. In terms of alternative graph, the traversing of a single route is represented by three nodes: the first node and the third node correspond to the train waiting on the first and on the last blocks of the route, respectively. These two blocks are associated to platforms, arrival/exit points or final tracks of the station, where a train can stop for a while, waiting for the release of the next resource. The second node corresponds to traversing the route, which is possible only if all incompatible routes are empty, as well as the final block of the route. Hence, the train never stops when traversing a route. The weight on the outgoing arcs from the first and the third nodes represents the minimum stopping time for the train on the two blocks, while the weight on the outgoing arc from the second node is the route running time.

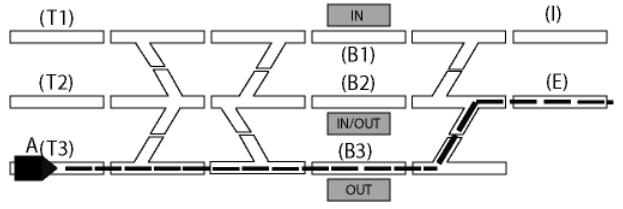


Fig. 3. A path

For example, in Figure 4 train  $A$  must traverse the two consecutive routes R5 and R6. The corresponding alternative graph is shown in Figure 4.

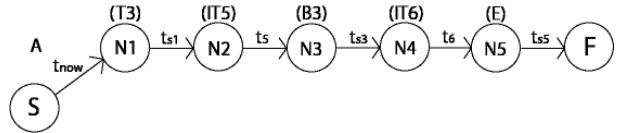


Fig. 4. The chain associated to the path of Figure 3

Here, node  $S$  represents time zero, arc  $(S, N1)$  is weighted with the current time  $t_{now}$ , and node  $F$  is a dummy node representing the end of the chain. Nodes  $N1, N3$  and  $N5$  are the associated to the train stopping at blocks  $T3, B3$  and  $E$ , respectively.  $t5$  and  $t6$  are the traversing times of routes  $R5$  and  $R6$ , respectively. Finally,  $ts1, ts3$ , and  $ts5$  represent the stopping times at the final track  $T3$ , the platform  $B3$ , and the exit node  $E$ , respectively. The departure time of train  $A$  from the station is the time at which the train leaves platform  $B3$ , i.e. the starting time of operation  $n_{o_A} = N4$ , when the train enters the route to the exit.

As already observed, conflicts among trains derive from incompatibility between routes, i.e. between trains covering the same route and between trains covering routes with at least one block in common. In Figure 1 the pairs of different incompatible routes are:

- (R1, R2), (R1, R3), (R1, R4), (R4, R5), (R3, R7), (R4, R9), (R2, R10), (R7, R8), (R9, R8), (R10, R8). For all these pairs the first block of the first route coincides with the last block of the second route.
- (R1, R11), (R2, R3), (R2, R4), (R3, R4), (R5, R9). These are all the pairs of routes starting with the same block.
- (R9, R10), (R9, R11), (R10, R11), (R6, R8). These are all the pairs of routes ending with the same block.
- (R3, R10), (R3, R9). These are all the pairs of crossing routes, i.e. having one intermediate block in common.

Since in our model we distinguish the occupancy of the first, the last blocks of each route and the traversing

of the other intermediate blocks, we also have to introduce the incompatibility between traversing of routes and block occupancy: a train cannot cover a route if there is an other train in a block incompatible with the route. The resulting incompatible pairs in Figure 1 are:

- (R1, B1), (R2, T2), (R3, T2), (R4, T3), (R5, B3), (R7, B2), (R9, B2), (R10, B2).

An instance of the problem at the current time  $t_{now}$  is given by the terminus state, i.e. which routes are available, and the information about the current position and the off-line timetable of all circulating trains. In Figure 2 an instance of the problem with three trains  $A$ ,  $B$  and  $C$  in the terminus is given, while Figure 5 shows its alternative graph formulation, once the paths for the three trains are defined. In this instance, train  $A$  has to leave the station from platform  $B2$  covering routes  $R7$  and  $R8$ ; train  $B$  has to enter the station and stop in the final track  $T3$ , covering routes  $R1$  and  $R4$ ; train  $C$  has to leave the station through route  $R6$ . Figure 2 represents the paths assigned to  $A$ ,  $B$ , and  $C$ , and, in gray, the tracks they have in common. Hence, the incompatible route pairs are :  $(R7(A), R4(B))$  and  $(R6(C), R8(A))$ . In the alternative graph these incompatibilities are modeled like in Figure 5.

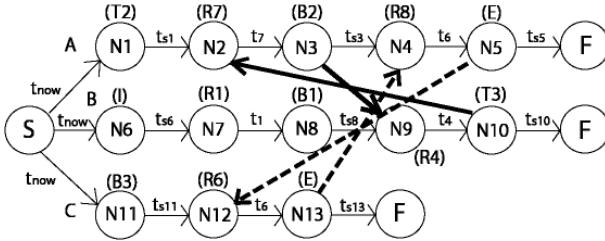


Fig. 5. The alternative graph for an example with three trains

Incompatibility  $(R7(A), R4(B))$  is modeled with the pair of alternative arcs  $(N3, N9), (N10, N2)$ : arc  $(N3, N9)$  corresponds to giving precedence to  $A$  over  $B$ , that is train  $B$  cannot enter route  $R4$  until after  $A$  leaves route  $R7$ . Conversely, with arc  $(N10, N2)$  train  $A$  cannot enter route  $R7$  before train  $B$  leaves route  $R4$  (enters the final track  $T3$ ). A similar discussion holds for incompatible pair  $(R6(C), R8(A))$ , for which the alternative arcs are drawn with broken lines in Figure 5.

When the alternative graph is formulated as in problem (3.1), and given the precedence graph associated to any feasible solution, we have that the longest path in the graph from node  $S$  to any node  $i$  equals the starting time of operation  $i$  in a feasible schedule, i.e. the time at which the train associated to the node enters the

associated resource in the terminus. We call this quantity the *head*  $t_i$  of node  $i$ .

#### IV. ROUTING AND SEQUENCING PHASE

We next describe the routing algorithm. The main problem to face when assigning paths to incoming trains is that a train entering the station from platforms  $B1$  or  $B2$  may affect the departure of earlier trains, due to the incompatibility between routes. The basic idea of the routing algorithm is to build a first solution in a greedy fashion, by assigning to each train a sequence of routes in such a way that the train can reach the exit with the smallest possible delay, and without delaying previously scheduled trains. In other words, in this phase we associate a route *and* a temporary schedule to each train.

Given the current status of the terminus, and the set of trains to manage (either within the station or incoming), the following quantities are computed:

$w_i^P$ , waiting time at platform  $P = E, F, I$  for train  $i$ : it is the minimum waiting time at platform  $P$  for train  $i$ . It may depend on the train status, e.g. if it is late or not, or during peak/smooth hours. It is the *remaining* waiting time if the train is waiting at the platform at  $t_{now}$ . In particular,  $w_i^E$  is the waiting time at the *exit* platform, necessary for letting passengers to enter the train,  $w_i^I$  is the waiting time at the *input* platform, necessary for letting passengers to leave the train, and  $w_i^F$  is the waiting time at a *final track*, necessary for letting the driver to rest before a new trip, or at least to walk from one end to the other of the train and change train direction.

$t_i^P$ , traversing time to a relevant point  $P = E, F, I$  for train  $i$ : it is the minimum traversing time for train  $i$  from its current position to  $P$  considering the station empty, i.e. without conflicts, plus the waiting time at intermediate platforms, if any. In particular,  $t_i^E$  is the time to the *exit* point,  $t_i^F$  is the time to a specific *final track*, and  $t_i^I$  is the time to a given *input* platform.

$d_i^P = \max\{t_{now} + t_i^P; O^i + t_i^P - t_i^E\}$ , due date at point  $P = E, F$  for train  $i$ : this quantity is computed in real time for each train, and it is the time at which the train should reach that point of the terminus. In particular,  $d_i^E = \max\{t_{now} + t_i^E; O^i\}$  is the due date at the *exit* point,  $d_i^F = \max\{t_{now} + t_i^F; O^i + t_i^F - t_i^E\}$  is the due date at the closest *final track*. Clearly, if train  $i$  is waiting in a final track, or between a final track and the exit, the latter quantity is not computed.

In particular:

- if train  $i$  is traversing an exit route, it is considered not controlled anymore by the TMS, simply its exit time is estimated as the first time at which the route will be available for future trains;
- if train  $i$  is waiting in a final track, or between a final track and the exit, a due date is associated to it,  $d_i^E$ , at which the train should reach the exit point of the terminus;
- if train  $i$  still has to reach the station, or it is in the station between the input point and a final track, two due dates are associated to it,  $d_i^E$ , at which the train should reach the exit point of the terminus, and  $d_i^F < d_i^E$ , at which the train should reach a final track of the terminus.

All the due dates are then ordered for increasing value, and a partial routing is built for the trains by considering one train at a time. In particular, when considering a  $d_i^F$  value, all possible paths for train  $i$  from its current position to the terminus exit point are considered, and the arrival times  $a_i^{T1}, a_i^{T2}, a_i^{T3}$  of train  $i$  at the three possible final tracks is computed, taking into account the incompatibilities with previously scheduled trains. Then, the route minimizing the quantity  $|d_i^F - a_i^{Tj}|$ , for  $j = 1, 2, 3$ , is assigned to train  $i$ , and the corresponding chain of operations is added to the alternative graph, together with the precedence constraints necessary to solve possible conflicts. Similarly, when considering a  $d_i^E$  value, all possible paths for train  $i$  from its current position to the terminus exit point are considered, and the departure times  $a_i^{B2}, a_i^{B3}$  of train  $i$  from the two possible platforms  $B2$  and  $B3$  is computed, taking into account the incompatibilities with previously scheduled trains. Then, the route minimizing the quantity  $|d_i^E - a_i^{Bj}|$ , for  $j = 2, 3$ , is assigned to train  $i$ , and the corresponding chain of operations is added to the alternative graph, together with the precedence constraints necessary to solve possible conflicts.

In this way, when all the due dates have been considered, the resulting alternative graph provides a conflict free plan of operations for all train movements. The head  $t_i$  of each node  $i$  is the earliest starting time of the associated operation, i.e. the earliest time at which the train can enter the associated resource.

In a last post-processing step, the departure times of the trains are adjusted in order to improve their punctuality. In this phase only the nodes associated to the departure time of each train are considered. To this aim, let us assume that, after the sequencing phase, the trains are sequenced in the order  $1, 2, \dots, N$ , and let  $o_j$  be the node associated to the departure time of train

$j$  from the station. Also, let  $l(h, k)$  be the length of the longest path from  $o_h$  to  $o_k$  in the precedence graph, with  $k > h$ , and let  $\tau_j \geq t_{o_j}$  be the new value of the departure time for train  $j$  after the post-processing.

If train  $i$  is late, i.e. if  $t_{o_i} \geq O^i$ , we fix  $\tau_i = t_{o_i}$ . In fact, fixing  $\tau_i < t_{o_i}$  would violate some precedence constraint, while  $\tau_i > t_{o_i}$  would not improve the objective function for train  $i$ , and neither for the other trains.

If train  $i$  is early, i.e. if  $t_{o_i} < O^i$ , postponing its departure time until after time  $O^i$  would improve its punctuality. On the other hand, delaying the departure of train  $i$  might cause a delay in the departure time of some other later train. In such cases we delay train  $i$  only if there is an overall improvement in the objective function. It is quite simple to compute the best departure time  $\tau_i$  for train  $i$  on the precedence graph as follows:

$$\tau_i = \min\{O^i; \min_{k=i, \dots, N} \{\max\{t_{o_k}; O^k + m_p\} - l(i, k)\}\} \quad (2)$$

In fact, if train  $i$  leaves the station  $\epsilon$  time units after  $\tau_i$  there will be at least one later train  $k$  enforced to leave the station with an additional delay  $\epsilon$ , while delaying the train from  $t_{o_i}$  to  $\tau_i$  does improve the objective function for train  $i$  without worsening the punctuality for all other trains.

At the end of the post-processing phase we have an *intermediate* schedule addressing the punctuality function only. In particular, all departure times  $\tau_i < t_{o_i}$  or  $\tau_i \geq t_{o_i} + m_p$  cannot be anticipated without violating some precedence constraint and/or worsening the punctuality function. However, the departure time of all trains that are scheduled to depart in the time window  $O^i \leq \tau_i < O^i + m_p$  can be delayed without affecting the punctuality, i.e. the earliness/tardiness objective function. More precisely, the departure time of train  $i$  can vary in the time window  $[\tau_i; \tau_i^{max}]$ , with:

$$\tau_i^{max} = \min\{O^i + m_p; \min_{k=i, \dots, N} \{\max\{\tau_k; O^k + m_p\} - l(i, k)\}\} \quad (3)$$

Note that  $\tau_i^{max} \geq \tau_i$ . In the scheduling phase we use these margins in order to improve the headway objective function.

## V. SCHEDULING PHASE

Given the alternative graph resulting from routing/sequencing phase, in the scheduling phase the departure times of the trains are adjusted in order to improve the regularity of the train service. The purpose of this phase is to optimize the headway without affecting the punctuality and without changing the routing and the

sequencing determined in the previous phase. Hence, the scheduling problem consists in determining new departure times  $h_1, h_2, \dots, h_N$  for the trains such that the train punctuality does not vary, i.e.  $\tau_i \leq h_i \leq \tau_i^{max}$ , and the regularity is optimized, i.e. the quantity  $\sum_{i=1,\dots,N} |(h_i - h_{i-1}) - (O^i - O^{i-1})|$  is minimized. By definition, we assume  $h_0$  and  $O^0$  equal to the real departure time and the off-line departure time, respectively, of the last train which left the station before  $t_{now}$ .

### Problem 5.1:

$$\begin{aligned} & \min \sum_{i=1,\dots,N} |(h_i - h_{i-1}) - (O^i - O^{i-1})| \\ & \text{s.t.} \\ & \quad \tau_i \leq h_i \leq \tau_i^{max} \quad i = 1, \dots, N \end{aligned}$$

In what follows, we show that this problem can be solved at optimality with a simple algorithm, which iteratively updates the values  $h_i$ , initially set to  $h_i = \tau_i$ .

**Definition 5.2:** A group is a maximal set of consecutive trains such that  $h_i - h_{i-1} \leq O^i - O^{i-1}$ .

The algorithm works with groups of trains, starting from train 1. If  $\tau_1 - \tau_0 > O^1 - O^0$ , there is clearly no advantage in delaying train 1, since it would worsen the headway of the first interval by an amount larger or equal to the improvement in the headway of the second interval. If  $\tau_1 - \tau_0 < O^1 - O^0$ , it might be useful delaying train 1 of a quantity  $\delta_1 > 0$ . In order to compute the maximum value of  $\delta_1$ , consider the group train 1 belongs to, composed by the first  $k$  trains. In order to have an improvement  $\delta_1$  in the objective function, it is necessary to delay all the first  $k$  trains by an amount  $\delta_1$ , otherwise improving the headway of the first train will worsen the headway of a later train, i.e. it must hold:

$$\delta_1 \leq \tau_i^{max} - \tau_i \quad \text{for } i = 1, \dots, k. \quad (4)$$

Consider now train  $k + 1$ . By definition of group,  $h_{k+1} - h_k > O^{k+1} - O^k$ . Hence, delaying the first  $k$  trains by an amount  $\delta_1 \leq h_{k+1} - h_k - (O^{k+1} - O^k)$  would improve both the headway of train 1 and train  $k + 1$ . On the other hand, delaying the first  $k$  trains more than  $h_{k+1} - h_k - (O^{k+1} - O^k)$  would improve the headway of train 1 and worsen that of train  $k + 1$ , unless we also delay train  $k + 2$ . Hence, we first set  $h_i = \tau_i$ , for  $i = 1, \dots, N$  and compute the value:

$$\begin{aligned} & \max \delta_1 \\ & \text{s.t.} \\ & \left\{ \begin{array}{l} \delta_1 \leq O^1 - O^0 - (h_1 - h_0) \\ \delta_1 \leq \tau_i^{max} - h_i \quad i = 1, \dots, k. \\ \delta_1 \leq h_{k+1} - h_k - (O^{k+1} - O^k) \end{array} \right. \end{aligned}$$

Then, we set  $h_i = h_i + \delta_1$  for  $i = 1, \dots, k$ . If  $\delta_1 = O^1 - O^0 - (h_1 - h_0)$ , the headway of train 1 is now the best possible in terms of headway, and the whole procedure can be repeated by substituting train 1 with train 2. If  $\delta_1 = h_{k+1} - h_k - (O^{k+1} - O^k)$ , train 1 belongs now to a new larger group of trains, which includes train  $k + 1$ . Hence, we can re-apply the whole procedure computing a new additional delay for the trains belonging to this group. If  $\delta_1 = \tau_j^{max} - h_j$ , for some  $j \in \{1, \dots, k\}$ , a further delay for the trains from 1 to  $j$  would not improve the overall regularity and/or would worsen the punctuality function. Hence, the whole procedure can be repeated by substituting train 1 with train  $j + 1$ .

Figure 6 shows the overall scheduling algorithm. We assume that the last train left the station at time  $h_0 = 0$ , and that there exists a dummy train  $n + 1$  such that  $h_{n+1} = +\infty$ . Hence, train  $n$  is always the last train of a block, and it can be delayed up to  $\tau_n^{max}$ .

### Procedure Scheduling

```

begin
  set int = 1;  $h_0 = 0$ ;  $h_{n+1} = +\infty$ ;  $h_i = \tau_i$  for  $i = 1, \dots, n$ 
  while (int ≤ n) do
    begin
      while ( $h_{int} - h_{int-1} \geq O^{int} - O^{int-1}$ ) do
        int = int + 1
        let  $k + 1$  be the first interval  $k + 1 > int$ :
         $h_{k+1} - h_k > O^{k+1} - O^k$ 
        compute max  $\delta_{int}$  such that:
        
$$\left\{ \begin{array}{l} \delta_{int} \leq O^{int} - O^0 - (h_{int} - h_0) \\ \delta_{int} \leq \tau_i^{max} - h_i \quad i = int, \dots, k. \\ \delta_{int} \leq h_{k+1} - h_k - (O^{k+1} - O^k) \end{array} \right.$$

        set  $h_i = h_i + \delta_{int}$  for  $i = int, \dots, k$ 
        if  $\delta_{int} = O^{int} - O^0 - (h_{int} - h_0)$  set int = int + 1
        if  $\delta_{int} = \tau_j^{max} - h_j$  for some  $j \in \{int, \dots, k\}$ 
          set int = j + 1
    end
end

```

Fig. 6. The procedure for headway optimization

**Theorem 5.3:** The algorithm in Figure 6 optimizes traffic headway.

**Proof.** Let us consider the output solution  $h_1, \dots, h_N$  of procedure scheduling, and the departure time  $y_1, \dots, y_N$  of an optimal solution. In order to prove the theorem, it is sufficient to prove that, if there exists an optimal solution such that  $y_i = h_i$  for  $i = 0, \dots, k - 1$ , then there exists an optimal solution such that  $y_k = h_k$ , for

$k = 1, \dots, N$ .

Let us consider first the case  $k = 1$ . If  $h_1 \neq y_1$ , only three cases are possible:

- 1)  $h_1 - h_0 = O^1 - O^0$ . In this case, fixing  $y_1 = h_1$  and letting unchanged  $y_i$ ,  $i = 2, \dots, N$  will decrease the cost for the first interval by the amount  $|y_1 - h_1|$  and cannot increase the cost of the second interval more than  $|y_1 - h_1|$ . Hence, there is an optimal solution  $y$  such that  $y_1 = h_1$ .
- 2)  $h_1 - h_0 < O^1 - O^0$ . In procedure scheduling this case may only occur if there is a train  $j \geq 1$  such that  $h_j = \tau_j^{max}$ , and  $h_i - h_{i-1} < O^i - O^{i-1}$  for any  $i = 1, \dots, j$ . In this case, the contribution of the first  $j$  intervals to the objective function is  $\sum_{i=1, \dots, j} O^i - O^{i-1} - (h_i - h_{i-1}) = O^j - O^0 - \tau_j^{max} + h_0$ . Since  $y_j \leq \tau_j^{max}$ , it must be  $\sum_{i=1, \dots, j} O^i - O^{i-1} - (y_i - y_{i-1}) \geq O^j - O^0 - \tau_j^{max} + h_0$ . Hence, fixing  $y_i = h_i$  for  $i = 1, \dots, j$  and letting unchanged  $y_i$ ,  $i = j+1, \dots, N$  will decrease the cost for the first  $j$  intervals at least by the amount  $\tau_j^{max} - y_j$  and cannot increase the cost of the  $(j+1)$ -th interval more than  $\tau_j^{max} - y_j$ . Hence, also in this case there is an optimal solution  $y$  such that  $y_1 = h_1$ .
- 3)  $h_1 - h_0 > O^1 - O^0$ . In procedure scheduling this case may only occur if  $h_1 = \tau_1$ , and therefore  $y_1 > h_1$ . In this case, anticipating  $y_1$  to the value  $h_1$  and letting unchanged  $y_i$ ,  $i = 2, \dots, N$  will decrease the cost for the first interval by the amount  $|y_1 - h_1|$  and cannot increase the cost of the second interval more than  $|y_1 - h_1|$ . Hence, also in this case there is an optimal solution  $y$  such that  $y_1 = h_1$ .

Consider now the case  $k = 2, \dots, N$ , and assume that there exists an optimal solution such that  $y_i = h_i$  for  $i = 1, \dots, k-1$ . Also in this case if  $h_k \neq y_k$ , only three cases are possible:

- 1)  $h_k - h_{k-1} = O^k - O^{k-1}$ .
- 2)  $h_k - h_{k-1} < O^k - O^{k-1}$ .
- 3)  $h_k - h_{k-1} > O^k - O^{k-1}$ .

By applying the same arguments as in the case  $k = 1$ , it follows that in all three cases there is an optimal solution in which  $y_k = h_k$ , thus completing the proof.

## VI. COMPUTATIONAL RESULTS

This section deals with the performance of the routing/sequencing algorithm with respect to the punctuality objective function, and with the performance of the scheduling algorithm with respect to the regularity objective function. We report in particular on our experience with a practical case study, concerning the real

time management of rail traffic at an Italian metro rail terminus (see Figure 1).

All the computational experiments are carried out by using a detailed simulator of the terminus, equivalent to the safety system used in the terminus for monitoring train traffic. Hence, even if the experiments are based on randomly generated data, the results are quite reliable. The algorithm is implemented in C++ language, and executed on a Personal Computer equipped with an Intel Pentium III – 1 GHz processor.

An instance of the problem is given by several fixed data and some variable data. Fixed data include the infrastructure, described in Section II-A, the train characteristics, and the off-line timetable. Variable data include the sequence of trains entering the station, which is randomly generated. For each instance we define a punctuality margin  $m_p$  and an input delay for each incoming train, chosen as a random variable with uniform distribution in the  $[0, m_d]$  interval, where  $m_d$  is called the maximum input delay. Note that, due to the random input delays, the sequence of trains entering the station can be different from the off-line planned sequence. In such cases, which also occur in practice due to trains coming from different origins, the sequencing algorithm must re-sequence the trains in the off-line order.

We tested our algorithm on 160 instances by varying the number of trains, and parameters  $m_p$  and  $m_d$ . Three different sets of problem instances were generated.

The first set of instances aims at determining the computation times of the algorithm when varying the number of trains to schedule. In fact, the real time requirement imposes strict time limits for computing a new plan of operations. The set consists of 60 instances obtained by fixing the punctuality margin value  $m_p = 120$  sec and the max delay value  $m_d = 480$  sec, and by varying the number of trains  $n$  in the set  $n = \{5, 10, 15, 20, 25, 30\}$ . For each value of  $n$  we have considered 10 instances.

Figure 7 shows the computation times of the sequencing phase and of the scheduling phase of the algorithm, expressed in seconds. The computation time of both phases is almost constant for different instances with the same number of trains, and does not depend significantly on other parameters, such as the margins  $m_p$ ,  $m_d$  and by the off-line timetable. In particular, the algorithm is able to generate a plan of operations in less than 5 seconds for up to 25 trains, thus being suitable for real time purposes. More in details, for the sequencing phase time increases almost linearly with the number of trains, while for the scheduling phase the computation time is not very sensible to the number of trains, and up to 30 trains it is always smaller than 1 second.

The remaining sets of experiments aim at evaluating

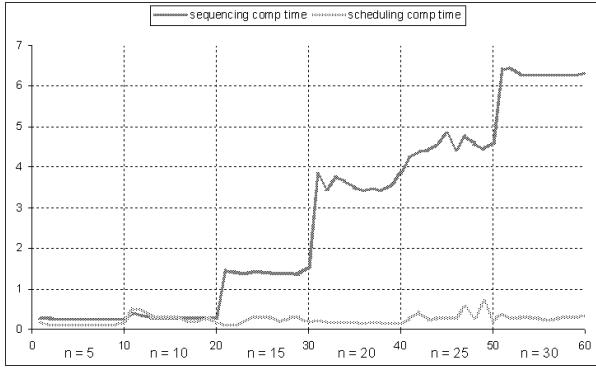


Fig. 7. Computation times (seconds) for  $m_d = 480$  seconds and  $m_p = 120$  seconds

the performance of the two phases of the algorithm, for varying  $m_p$  and  $m_d$ . All the experiments are executed with reference to the behavior of the terminus during the critical time interval of transition from smooth to peak traffic, between 7:30 and 9:15 a.m., corresponding to 20 incoming trains. The slack included in the off-line timetable, which can be used to recover input delays, ranges from 300 seconds in the smooth time period to 60 seconds in the peak hour. The computation times for the following two sets of experiments never exceed four seconds for the sequencing phase and one second for the scheduling phase.

The second set of instances consists of 50 instances obtained by fixing the punctuality margin value  $m_p = 120$  seconds, and by generating 10 random instances for each value of  $m_d$  in the set  $m_d = \{180, 240, 300, 360, 480\}$  seconds.

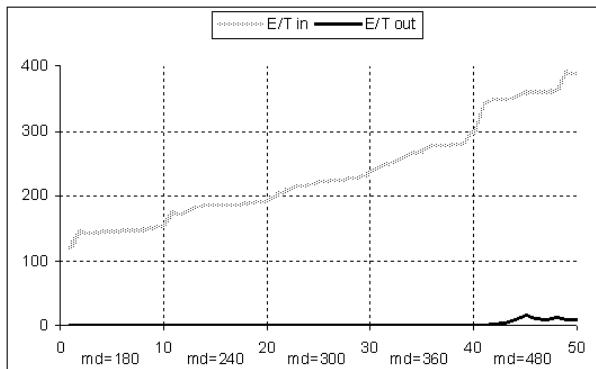


Fig. 8. Earliness/Tardiness for  $m_p = 120$  seconds

Figures 8 and 9 show the objective function values for the 50 instances. In particular, the instances are numbered from 1 to 50 and ordered for increasing total input delay for each value of  $m_d$ . For each instance, figure 8 shows the average input earliness/tardiness value

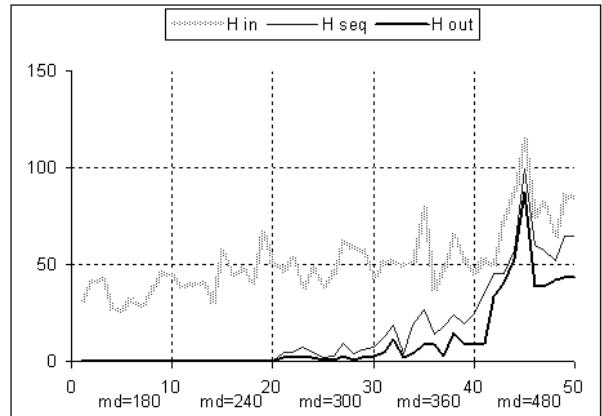


Fig. 9. Headway for  $m_p = 120$  seconds

of incoming trains (referred to as "E/T in") and the average output earliness/tardiness value "E/T out" of outcoming trains, as planned by the routing/sequencing algorithm. Figure 9 shows, for each instance, the average input headway value "H in" of incoming trains , the average output headway value "H seq" of outcoming trains computed by the routing/sequencing algorithm , and the average output headway value of outcoming trains "H out" computed by the scheduling algorithm.

This set of instances show that the routing/sequencing algorithm is able to exploit quite effectively the slack included in the off-line timetable in order to recover train punctuality. However, since the problem is bi-objective in nature, pursuing the punctuality objective function is not sufficient in order to provide a good level of service. In fact, the output headway after the routing/sequencing phase is satisfactory for  $m_d \leq 300$ , which is the usual behavior of the line, while for larger input delays becomes less regular. The scheduling procedure is therefore necessary in order to improve it. After the scheduling phase the traffic is quite acceptable for  $m_d$  growing up to 360 seconds.

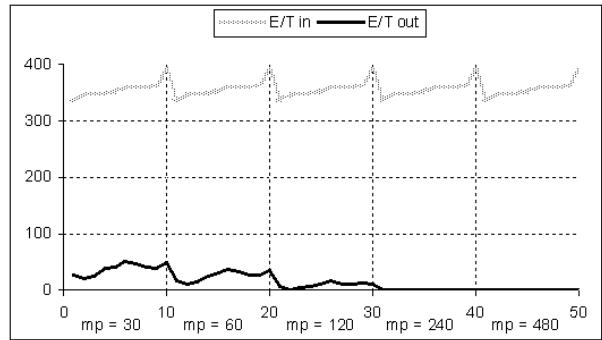


Fig. 10. Earliness/Tardiness for  $m_d = 480$  seconds

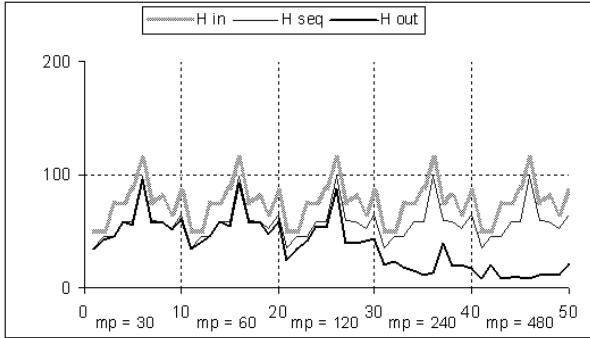


Fig. 11. Headway for  $m_d = 480$  seconds

The third set of instances aims at evaluating the algorithm performance when varying the punctuality margin  $m_p$ . We generated 10 random instances obtained by fixing  $m_d = 480$  seconds. From each of the 10 instances we then generated 5 instances by varying the punctuality margin in the set  $m_p = \{10, 30, 60, 120, 240\}$  seconds, thus obtaining 50 instances grouped in 5 groups of 10. The results are shown in Figures 10 and 11.

As expected, the larger flexibility allowed by larger punctuality margins allows for a better regulation of rail traffic, both in terms of punctuality and regularity. The TMS is able to deal with high values of the input delay, recovering most of it, although the scheduling algorithm is less performing, at least for small values of the punctuality margins. When  $m_p \geq m_d/2$ , the headway improves significantly after the scheduling phase. This behavior is due to the fact that, when a train is scheduled to depart later than its off-line departure time plus the punctuality margin, the scheduling algorithm cannot change its departure time, thus having little means to improve the regularity of rail traffic.

## VII. CONCLUSION AND FUTURE RESEARCH

This paper addresses the real time scheduling of train service in a Metro rail terminus. Since the problem is bi-objective in nature, the train schedules are generated in two steps. The first step addresses the problem of routing and sequencing trains through the station with the objective of optimizing the punctuality, which it is solved with a fast heuristic. The second step addresses the problem of scheduling train departures with the objective of optimizing the regularity of train service, under the constraint that the first objective function does not deteriorate. This second problem is solved at optimality. The computational results, carried out on the basis of a real case, are quite promising. Unfortunately, a comparison with the performance of human local area managers was not possible, due to the real time nature

of the problem. However, our results show that the scheduling algorithm is able to generate solutions that are feasible in practice. On the performance side, we observe that the optimization of punctuality is not always sufficient in order to guarantee a satisfactory level of train service. On the other hand, after optimizing the regularity, the solutions obtained are of good quality. This demonstrates the ability of the two optimization algorithms to automate train traffic control operations.

Future research should address the development of exact algorithms for the routing/sequencing phase and the comparison of optimal solutions with the solutions produced by our algorithm, in order to certify the quality of the solutions obtained. A further interesting research direction would be to study the problem as a bicriteria problem, i.e. considering the two objective functions simultaneously. Clearly, the automated system has to generate a single plan of operations. However, the availability of several Pareto optimal solutions would make possible a better comparison than the lexicographical order between the two objective functions, and consequently would make possible to generate a better compromise solution.

## REFERENCES

- [1] Adenso-Díaz B., Oliva González M., González-Torre P., 1999. On-line timetable re-scheduling in regional train services, *Transportation Research, Part B*, **33**, 387–398.
- [2] Balas, E. 1979. Disjunctive programming, *Annals of Discrete Mathematics*, **5**, 3–51.
- [3] Cai X., Goh C.J., Mees A.I., 1998. Greedy heuristics for rapid scheduling of trains on a single track, *IIE Transaction*, **30**, 481–493.
- [4] Cordeau J.F., Toth P., Vigo D., 1998. A Survey of Optimization Models for Train Routing and Scheduling, *Transportation Science*, **32** (4), 380–420.
- [5] Dorfman M.J., Medanic J., 2004. Scheduling trains on a railway network using a discrete event model of railway traffic, *Transportation Research, Part B*, **38**, 81–98.
- [6] Higgins A., Kozan E., Ferreira L., 1997. Modelling the Number and Location of Sidings on a Single Line Railway, *Computers and Operations Research*, **3**, 209–220.
- [7] Mascis A., Pacciarelli D., 2002. Job shop scheduling with blocking and no-wait constraints, *European Journal of Operational Research*, **143** (3), 498–517.
- [8] Pacciarelli D., Pranzo M., Mascis A., 2004. *Scheduling Models for Short-term Railway Traffic Optimisation*. Technical Report DIA-94-2004, Dipartimento di Informatica e Automazione, Università Roma Tre.
- [9] Roy B., Sussman B., 1964. *Les problèmes d'ordonnancement avec contraintes disjonctives*. Note DS No. 9bis, SEMA, Paris.
- [10] Şahin İ., 1999. Railway traffic control and train scheduling based on inter-train conflict management, *Transportation Research, Part B*, **33**, 511–534.

# Optimization models for the delay management problem in public transportation

Géraldine Heilporn, Luigi De Giovanni<sup>§</sup> and Martine Labbé

Université Libre de Bruxelles, Computer Science Department

Bd. du Triomphe CP 210/01, 1050 - Bruxelles, Belgium

{gheilpor, ldegiova, mlabbe}@ulb.ac.be

**Abstract**— Passengers travelling in public transportation networks often have to use different lines to cover the trip from their origin to the desired destination. As a consequence, the reliability of connections between vehicles is a key issue for the attractiveness of the intermodal transportation network and it is strongly affected by some unpredictable events like breakdowns or vehicle delays. In such cases, a decision is required to determine if the connected vehicles should wait for the delayed ones or keep their schedule. The Delay Management Problem (DMP) consists of defining the wait/depart policy which minimizes the total delay on the network. In this work, we present two equivalent Mixed Integer Linear Programming models for the DMP, able to reduce the number of variables with respect to the formulations proposed by literature. The two models are solved by a Branch and Cut procedure and by a Constraint Generation approach respectively, and preliminary computational results are presented.

**Keywords**— Delay Management, Mixed Integer Linear Programming, Constraint Generation.

## I. INTRODUCTION

THE attractiveness of the public transportation network is strongly related to the reliability of intermodal connections. But connections imply passenger transfers from one vehicle to another and can generate important waiting times, in particular when low frequency lines are taken into account. Missing a connection, because of a delayed incoming vehicle, implies waiting for the next one of the same line, thus remarkably increasing the total travel time.

We define an intermodal public transportation network as a set of train, metro, tramway and/or bus lines, the vehicles moving between different stations. Suppose that a vehicle is delayed. The users of this vehicle who want to transfer to another vehicle at a station could miss their connection. In fact, either the other vehicle waits for the

delayed one and the transfer is allowed, or it does not wait for the delayed vehicle. If a vehicle waits for the delayed one, the users travelling on it suffer a delay. Also, a delay is caused for passengers wishing to get on this vehicle later on, and possibly for subsequent other vehicles which will have to wait because of this delay. On the other hand, if a vehicle departs on time, only the passengers on the delayed vehicle suffer a delay, but it might be very high, as the passengers should wait for the next vehicle of the missed line.

To avoid passengers missing their transfers, one could force all departing vehicles to wait until the delayed one has arrived. But, in this case, the delay spreads out through the network, thus affecting many customers. On the other hand, if all vehicles depart on time, the number of affected passengers is minimized but they will suffer greater delays as they miss their connections. As a consequence, the best decision is generally to force only a subset of the vehicles to wait for the delayed ones. The delay management problem (DMP) consists in determining how the other vehicles of the network should react (to wait or not to wait) in order to minimize the sum of delays of all the passengers at their destinations.

Let  $S$  be the station set,  $V$  be the vehicle set, and suppose vehicle  $i$  arrives at station  $k$  with a delay  $D$ . Further, let  $A$  be the set of effectively used paths by the passengers, each path being defined as a sequence of direct rides between pairs of stations. We will consider a time horizon  $T$  which is the scheduled time interval between the stops of two vehicles of the same line at a given station. We consider that  $T$  is identical for all vehicles of the network. Finally, we suppose  $D < T$ .

We represent the change from vehicle  $i \in V$  to vehicle  $j \in V$  at station  $l \in S$  by the triplet  $(i, j, k)$  and we call it a connection. Suppose that a delay  $D$  occurs at a station  $k$  for the vehicle  $i$ . For the vehicles  $j \in V$  for which a connection  $(i, j, l)$  is possible,  $l \in S$  being any station after station  $k$ , there are two possibilities: either vehicle  $j$  waits for the delayed vehicle  $i$ , allowing

<sup>§</sup> The research is part of the project “Analysis and Optimization of Intermodal Public Transportation Networks in the Brussels Capital Region” funded by the Brussels-Capital Region in the context of the Prospective Research for Brussels program.

travellers on vehicle  $i$  to change to vehicle  $j$  at station  $l$ , or  $j$  leaves on time. In the latter case, the users of vehicle  $i$  needing to change to vehicle  $j$  at station  $l$  have to wait for the next vehicle of the same line (as  $j$ ). If vehicle  $j$  waits for vehicle  $i$ , we say the connection is maintained; otherwise it is suppressed. Of course, if vehicle  $j \in V$  waits for vehicle  $i$  at station  $l \in S$ , then it becomes a delayed vehicle too. Thus we also have to decide what the vehicles  $j' \in V$  should do, where the vehicles  $j'$  are those for which a connection  $(j, j', l')$  is possible,  $l' \in S$  being any station after  $l$  on vehicle  $j$ 's trip, and so on.

We also consider the possibility of reducing the delays: a slack time is defined for each stop of a vehicle at a station, for each direct ride of a vehicle from a station to the next one, and for each change from a vehicle to another at a station. For example, if we consider that at least one minute is necessary to change from vehicle  $i$  to vehicle  $j$  at station  $k$ , and that the scheduled departure time of vehicle  $j$  is three minutes after the scheduled arrival time of vehicle  $i$  to station  $k$ , the slack time is equal to two minutes.

In this paper, we consider the delay management problem from the passenger's point of view, so that we want to minimize the sum of passenger delays at their destinations.

This problem has been the object of a few papers, mainly by Schoebel. In [11], she presents three equivalent mixed integer models with one criteria. In the first two ones (which are also presented in [10]), she considers exactly the same problem as the one we discuss here. In the third model, she uses Event-Activity Networks [2], i.e. she considers a set of events and a set of activities that link these events. An event is the arrival or the departure of a given vehicle to or from a given station. An activity corresponds to either a direct ride of one vehicle from a given station to the next one, or a stop of a vehicle at a given station, or a change from one vehicle to another one at a given station. She minimizes the sum of delays of passengers at each activity. Furthermore, Schoebel presents in [11] a fourth mixed integer model, which generalizes the previous ones. Indeed, she considers there could be more than one initial delays, each of them causing some delays on a part of the network.

Schoebel and al. also proposes (see [6], [11]) some bicriteria models, minimizing the sum of delays of all vehicles at all stations as well as the number of suppressed connections. Finally, Schoebel presents in [11] a very general bicriteria model, whose goal is to minimize, (i) the number of users who cannot change from one vehicle to another (because of a suppressed connection), and, (ii)

the sum of delays that all the other users experience at their destinations. This model generalises all the previous ones. Scholl [12] and Kliewer [8] have also considered this problem, but they minimize the total waiting time of the users. Scholl's model does not include slack times, thus the initial delay cannot be reduced. Ginkel [5], [6] has presented the same bicriteria model as Schoebel, and has proposed to solve it by Event-Activity Networks [2].

A problem closely related to the DMP is presented in [1], where the impact of delays on the vehicle schedules (rather than the passenger comfort) is analyzed. A Integer Programming Model is presented where the decision variables establish if a delayed vehicle has to perform the following scheduled task or skip it. The slacks times are taken into account, while the changing activities and the missed connections are neglected. A heuristic procedure is proposed, based on backtracking for exploring the solutions space, that reduces the search by means of the elimination of certain branches which are not likely to generate good solutions. The evaluation of the quality of each obtained solution is made on the basis of the priority of each service, the passengers transported and the delays that these passengers have to suffer. The best results are offered to the traffic controller so that, using what-if tools, he or she may choose the alternative that he considers the most adequate from among these.

The subject of the DMP was brought up by two large traffic associations serving the states Rheinland-Pfalz and the Saarland (both in Germany). Public transportation companies are interested in analyzing the consequences of delays or changes in the schedule. On a regional train line in Rheinland-Pfalz, the 40km long *Lautertalbahn* leading from Kaiserslautern to Lauterecken, *Deutsche Bahn* installed an automatic system informing the bus drivers waiting at the stations about the exact arrival times of the incoming trains. Based on this information, the DMP determine whether the drivers should wait for a delayed train or depart on time, see Schoebel [11].

In Brussels (Belgium), the DMP concerning the connections metro-tram, metro-bus and tram-bus during the off-peak hours is an important issue for the public transport company STIB because it is considered as an important factor influencing the quality of service.

In this paper, we propose two new compact models for the DMP. In Section 2, we present a new graph interpretation, from which we derive a first model for the DMP containing three types of variables. We show that this model can be seen as a simplification of Schoebel's linear mixed integer model presented in [10]. In the third section, we further reduce the number of variables,

and obtain a second equivalent model. The first and the second models are solved by a standard MILP solver and by a constraint generation approach respectively. The procedures and some preliminary results are presented in the fourth section. Finally, the fifth section is devoted to some concluding remarks and suggestions for further research on DMP.

## II. A NEW MODEL FOR THE DMP

In this section, we present a new model for the DMP, exploiting a simple network graph representation based on the Event-activity-networks concept (see [2]).

Let  $R_i \subseteq S \times S$  be the set of vehicle  $i$ 's rides from a station to the next one, and  $S_i \subseteq S$  the set of stations where vehicle  $i$  stops. Furthermore, let  $C_a \subseteq V \times V \times S$  be the connection set on path  $a$ , with  $C = \cup_{a \in A} C_a$ .

We define the "Arr-Arr graph" as a directed graph  $G = (N, E)$ , where each node corresponds to an arrival of a vehicle at a station. The arcs correspond either to a vehicle direct ride, or to a connection ride, from a station to the next one. Thus we have:

- $N = \{(i, k)_{arr} : i \in V, k \in S_i\} \subseteq V \times S$ ,
- $E = \{((i, k)_{arr}, (i, l)_{arr}) : (k, l) \in R_i\}$  (direct ride from  $k$  to  $l$ )
- $\cup \{((i, k)_{arr}, (j, l)_{arr}) : (i, j, k) \in C\}$  (connection ride from  $k$  to  $l$ ).

We also associate weights  $slack_{uv}$  to the arcs  $(u, v) \in E$ . They represent the times that can be saved on those arcs, and we associate variables  $d_u$  to the nodes  $u \in N$  which correspond to the arrival delays at these nodes.

We illustrate this definition of the Arr-Arr graph with the example network depicted in Figure 1. We have two vehicles  $i, j$  and five stations  $h, k, l, m, n$ . So the nodes of the Arr-Arr graph for this network are  $(i, k)_{arr}$ ,  $(j, n)_{arr}$ ,  $(i, l)_{arr}$  and  $(j, k)_{arr}$ . The vehicle direct rides are  $((i, k)_{arr}, (i, l)_{arr})$  and  $((j, k)_{arr}, (j, n)_{arr})$ , and the connection rides are  $((i, k)_{arr}, (j, n)_{arr})$  and  $((j, k)_{arr}, (i, l)_{arr})$ . The related Arr-Arr graph is represented in Figure 2.

Note that this graph representation can be seen as a simplification of the graph representation used in [5], [10].

Let  $arcC$  be the connection arcs set, in other words the set of arcs of type  $((i, k)_{arr}, (j, l)_{arr})$ ,  $arcC_a$  being the set of connection arcs on path  $a$ . We also call  $arcDA$  the non connection arcs set, i.e. the set of arcs of type  $((i, k)_{arr}, (i, l)_{arr})$ . Furthermore, let  $p_a$  be the number of passengers on  $a$ , and  $v_a$  the last node on path  $a$ . Finally, we define a boolean variable  $z_a$  that says if all connections on path  $a$  are maintained or not:

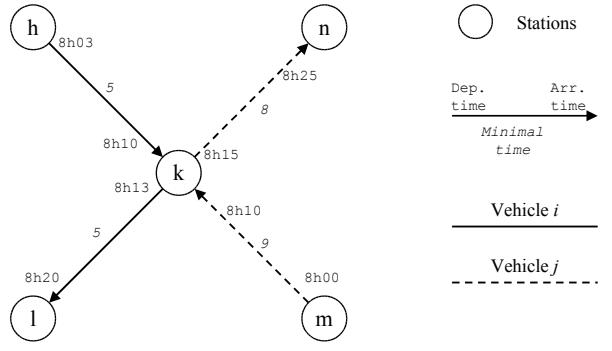


Fig. 1. An example of public transportation network.

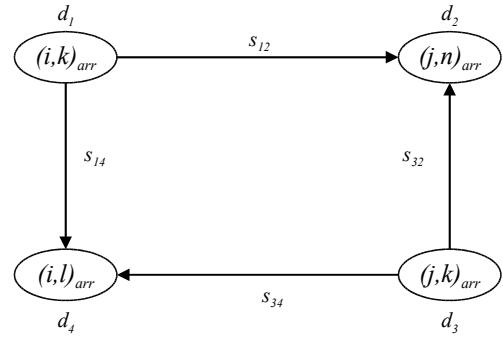


Fig. 2. Arr-Arr graph of the network of Figure 1.

$$z_a = \begin{cases} 1 & \text{if all connections on path } a \text{ are maintained,} \\ 0 & \text{otherwise,} \end{cases}$$

and a variable  $u_a$  which represents the delay at the last node of path  $a$  if  $z_a = 1$  and is 0 otherwise:

$$u_a = \begin{cases} d_{v_a} & \text{if all connections on path } a \text{ are maintained,} \\ 0 & \text{otherwise.} \end{cases}$$

Suppose that there is an initial delay at node 1. The DMP can be formulated as follows:

$$\begin{aligned} (\text{MILP1}) \quad & \min \sum_{a \in A} p_a [(1 - z_a)T + u_a] \\ \text{s.t.:} \quad & \begin{aligned} d_1 &= D & (1) \\ d_i - d_j &\leq slack_{ij} \quad \forall (i, j) \in arcDA & (2) \\ d_i - d_j + M_{ij}z_a &\leq slack_{ij} + M_{ij} & \forall a \in A, (i, j) \in arcC_a & (3) \\ Tz_a - u_a + d_{v_a} &\leq T \quad \forall a \in A & (4) \\ d_i &\geq 0 \quad \forall i \in N & (5) \\ u_a &\geq 0 \quad \forall a \in A & (6) \\ z_a &\in \{0, 1\} \quad \forall a \in A, & (7) \end{aligned} \end{aligned}$$

where  $M_{ij} = \text{slack}_{ij} + T$ .

The objective is to minimize the total sum of passenger delays, once they have arrived at their destinations. Note that a delay of  $T$  is considered for passengers missing at least one connection on their paths (they have to wait for the next vehicle). Constraint (1) gives the initial delay. Constraint (2) says that the delay at the end of a direct ride must be greater or equal to the delay at the beginning of this ride, minus perhaps the time that can be saved thanks to the slack on this ride. Constraint (3) says that, if all connections are maintained on path  $a$ , we must have the same constraint on the delays for connection rides as the one for the direct rides. If at least one connection is not maintained on path  $a$ , this constraint is redundant (thanks to constant  $M_{ij}$  definition and the assumption  $D < T$ ). Constraint (4) says that if all connections on path  $a$  are maintained,  $u_a$  is greater or equal to the delay at the last node of path  $a$  (equal thanks to the objective function). Constraint (5) says that the delay at a node is always greater or equal to zero.

In [10], Schoebel presents the following linear mixed integer problem for the DMP:

$$\begin{aligned} (\text{MILP}) \quad & \min \sum_{a \in A} p_a[(1 - z_a)T + u_a] \\ \text{s.t.:} \end{aligned}$$

$$d_{arr,1}^1 = D \tag{8}$$

$$d_{arr,i}^k \leq d_{dep,i}^k + \text{slack}_i^k \quad \forall i \in V, k \in S_i \tag{9}$$

$$\begin{aligned} d_{dep,i}^k \leq d_{arr,i}^l + \text{slack}_i^{kl} \\ \forall i \in V, k, l \in S_i : (k, l) \in R_i \end{aligned} \tag{10}$$

$$\begin{aligned} M_{ij}^k(z_a - 1) \leq d_{dep,j}^k + \text{slack}_{ij}^k - d_{arr,i}^k \\ \forall a \in A, (i, j, k) \in C_a \end{aligned} \tag{11}$$

$$u_a \geq d_{arr,i_a}^{k_a} - T(1 - z_a) \quad \forall a \in A \tag{12}$$

$$d_{arr,i}^k \geq 0 \quad \forall i \in V, k \in S_i \tag{13}$$

$$d_{dep,i}^k \leq T \quad \forall i \in V, k \in S_i \tag{14}$$

$$u_a \geq 0 \quad \forall a \in A \tag{15}$$

$$z_a \in \{0, 1\} \quad \forall a \in A \tag{16}$$

where  $M_{ij}^k = \text{slack}_{ij}^k + T$ .

$d_{dep,i}^k$  is the departure delay of vehicle  $i$  from station  $k$ , and  $d_{arr,i}^k$  is the arrival delay of vehicle  $i$  to station  $k$ .  $i_a$  is the last vehicle on path  $a$ , and  $k_a$  is the last station on path  $a$ . Furthermore, the slack times are  $\text{slack}_i^k$  for the stop time of vehicle  $i$  at station  $k$ ,  $\text{slack}_i^{kl}$  for the trip of vehicle  $i$  from station  $k$  to the next one  $l$ , and  $\text{slack}_{ij}^k$  for the change from vehicle  $i$  to vehicle  $j$  at station  $k$ .

We show that models (MILP) and (MILP1) are equivalent, i.e. they are both valid formulations of the DMP

and the optimal values of their LP relaxations are equal. Indeed, let FS be the set of all feasible solutions of (MILP), and FS1 be the set of all feasible solutions of (MILP1). Then FS1 is the projection of FS on the subspace of variables  $z_a, u_a, d_{arr,i}^k : i \in V, k \in R_i, a \in A$ .

**PROPOSITION 1** *Model (MILP1) is equivalent to model (MILP).*

*Proof:* First, let us show that constraints (14) are not necessary. Indeed, if we suppress them,  $d_{dep,i}^k$  is no more bounded. But the  $d_{dep,i}^k$  are used to determine the values of the  $d_{arr,i}^k$  variables, and thus the values of the  $d_{arr,i_a}^{k_a}$  variables. As those last variables occur in the objective function we want to minimize, we ever choose the  $d_{dep,i}^k$  values as small as possible. Since  $d_{arr,1}^1 = D < T$ , the constraints (9), (10) and (11) from (MILP) and the above argumentation show that we always choose the  $d_{dep,i}^k$  values such that  $d_{dep,i}^k < T$ .

Second, we show how we can suppress the variables  $d_{dep,i}^k$  by projecting the (MILP) feasible domain on the subspace of the  $z_a, u_a, d_{arr,i}^k : a \in A, i \in V, k \in S_i$  variables. By constraints (9) and (10) of (MILP), we know that:

$$\begin{aligned} d_{arr,i}^k - \text{slack}_i^k \leq d_{dep,i}^k \leq d_{arr,i}^l + \text{slack}_i^{kl} \\ \forall i \in V, k, l \in S_i : (k, l) \in R_i. \end{aligned}$$

Thus we also have:

$$\begin{aligned} d_{arr,i}^k - \text{slack}_i^k \leq d_{arr,i}^l + \text{slack}_i^{kl} \\ \forall i \in V, k, l \in S_i : (k, l) \in R_i. \end{aligned}$$

In the same way, by constraints (10) and (11), we have:

$$\begin{aligned} d_{arr,j}^k - \text{slack}_{ji}^k + M_{ji}^k(z_a - 1) \leq d_{arr,i}^l + \text{slack}_i^{kl} \\ \forall a \in A, (j, i, k) \in C_a, l \in S_i : (k, l) \in R_i. \end{aligned}$$

By applying the Fourier-Motzkin principle (see e.g. [9]), we obtain the following model:

$$(\text{MILP}') \quad \min \sum_{a \in A} p_a[(1 - z_a)T + u_a]$$

s.t.:

$$d_{arr,1}^1 = D \quad (17)$$

$$d_{arr,i}^k - slack_i^k \leq d_{arr,i}^l + slack_i^{kl} \quad \forall i \in V, k, l \in S_i : (k, l) \in R_i \quad (18)$$

$$d_{arr,j}^k - slack_{ji}^k + M_{ji}^k(z_a - 1) \leq d_{arr,i}^l + slack_i^{kl} \quad \forall a \in A, (j, i, k) \in C_a, l \in S_i : (k, l) \in R_i \quad (19)$$

$$u_a \geq d_{arr,i_a}^k - T(1 - z_a) \quad \forall a \in A \quad (20)$$

$$d_{arr,i}^k \geq 0 \quad \forall i \in V, k \in S_i \quad (21)$$

$$u_a \geq 0 \quad \forall a \in A \quad (22)$$

$$z_a \in \{0, 1\} \quad \forall a \in A \quad (23)$$

As (MILP') is the projection of (MILP), their optimal values are identical.

We can easily see that each variable  $d_{arr,i}^k$  corresponds to a variable  $d_n : n \in N$  (see above). Moreover, let the  $slack_{ij} : (i, j) \in E$  be equal to  $slack_i^k + slack_j^{kl}$  for the  $(i, j)$  in  $arcDA$ , and to  $slack_{ij}^k + slack_j^{kl}$  for the  $(i, j)$  in  $arcC$ . If we change those notations in the (MILP'), we obtain model (MILP1).

■

As (MILP1) does not consider departure delay variables, it is useful to show that no vehicle is allowed to leave before the scheduled departure time, which was ensured in (MILP) by constraints (13).

**PROPOSITION 2** *There exists an optimal solution of (MILP1) in which the vehicles never leave a station before the scheduled departure time.*

*Proof:* By contradiction, we suppose that, in an optimal solution, a vehicle has to leave a station before the scheduled depart time. Even if the slack times are zero, this means that the vehicle arrives at the next station before the scheduled arrival time, and that we will have a variable  $d_i < 0$ . But, this is in contradiction with constraint (5) from (MILP1).

■

### III. REDUCING THE NUMBER OF VARIABLES

Here we propose another model which involves variables  $z_a$  and  $u_a$  only. Consider the following model:

$$(MILP2) \quad \min \sum_{a \in A} p_a [(1 - z_a)T + u_a]$$

s.t.:

$$\begin{aligned} Tz_{a'} - u_{a'} + \sum_{(i,j) \in arcC_a} M_{ij}z_{a''(i,j)} \\ \leq T - D + \sum_{(i,j) \in E_a} slack_{ij} + \sum_{(i,j) \in arcC_a} M_{ij}, \\ \forall a, a', a''(\bar{i}, \bar{j}) \in A \text{ so that the first node on path} \\ a \text{ is } 1, v_{a'} = v_a \text{ and } (\bar{i}, \bar{j}) \in arcC_a \cap arcC_{a''} \quad (24) \\ u_a \geq 0 \quad \forall a \in A \quad (25) \\ z_a \in \{0, 1\} \quad \forall a \in A, \end{aligned}$$

where  $E_a$  is the set of arcs on path  $a$ .

We show that models (MILP1) and (MILP2) are equivalent. Indeed, we suppress the arrival delay variables from (MILP1) by projecting the polyhedron associated to the LP-relaxation of (MILP1) on the space of the other variables  $z_a, u_a : a \in A$  of the problem. We then obtain (MILP2).

**PROPOSITION 3** *Let FS1 be the (MILP1) feasible solutions set, and FS2 the (MILP2) feasible solutions set. Then FS2 is the FS1 projection on the subspace of the  $z_a, u_a : a \in A$  variables, that is to say:  $FS2 = \{(z_a, u_a) \text{ s.t. there exists } d_i \text{ with } (z_a, u_a, d_i) \in FS1\}$ .*

*Proof:* Starting from the (MILP1), we apply Fourier-Motzkin elimination to  $d_i : i \in N$  variables. In other words we project the (MILP1) feasible domain on the subspace of the  $z_a, u_a : a \in A$  variables.

For each arc  $(i, j)$  from graph  $G = (N, E)$ , we define:

$$l_{ij} = \begin{cases} \min_{a \in A: (i,j) \in arcC_a} (slack_{ij} + M_{ij}(1 - z_a)), \\ \quad \quad \quad \text{if } (i, j) \in arcC \\ slack_{ij}, \quad \quad \quad \text{if } (i, j) \in arcDA \end{cases}$$

These weights on the arcs allow us to evaluate the delay at each node. Indeed, if  $(i, j) \in arcDA$ , we have, thanks to constraint (2):

$$d_j \geq d_i - slack_{ij}.$$

■ In the same way, if  $(i, j) \in arcC$ , constraint (3) says :

$$d_j \geq d_i - slack_{ij} - M_{ij}(1 - z_a) \quad \forall a \in A : (i, j) \in arcC_a,$$

thus we have :

$$d_j \geq d_i - \min_{a \in A: (i,j) \in arcC_a} (slack_{ij} - M_{ij}(1 - z_a)).$$

Consequently, for each arc  $(i, j)$  of the graph, we have:

$$d_j \geq d_i - l_{ij}.$$

Let us now concentrate on a path  $a$  whose first node is 1, and suppose its nodes are called  $1, 2, \dots, v_a$ . We know, from (MILP1), that  $d_1 = D$  et  $d_2 \geq d_1 - l_{12}$ . If we put together these two inequalities, we obtain:

$$d_2 \geq D - l_{12}.$$

As we also know that  $d_3 \geq d_2 - l_{23}$ , we have:

$$d_3 \geq D - l_{12} - l_{23}.$$

If we continue like this up to  $d_{v_a}$ , and if we call  $E_a$  the set of arcs on path  $a$ , we obtain:

$$d_{v_a} \geq D - \sum_{(i,j) \in E_a} l_{ij}.$$

It is clear that these arguments are valid for all paths  $a$  whose first node is 1. In addition, it is not necessary to consider other paths. Indeed, suppose we have a path  $a'$  such that  $v_{a'} = v_a$  and whose first node is not 1. Two situations can happen:

- if there is no intersection between this path  $a'$  and paths whose first node is 1, then the delay at each node of path  $a'$  is zero;
- if the path  $a'$  has at least one common part with paths whose first node is 1, then the  $a'$  nodes which are before the first intersection with paths whose first node is 1 do not have any delay, and the delays at the other  $a'$  nodes are the same as those on paths whose first node is 1. Thus we can write:

$$d_{v_{a'}} = d_{v_a} \geq D - \sum_{(i,j) \in E_a} l_{ij}.$$

Thus the following inequality :

$$d_{v_a} \geq D - \sum_{(i,j) \in E_a} l_{ij} \quad \forall a \in A \text{ whose first node is 1} \quad (27)$$

can replace constraints (2) and (3).

Furthermore, if we put together inequality (27) and constraint (4), we have:

$$D - \sum_{(i,j) \in E_a} l_{ij} \leq d_{v_a} \leq T(1 - z_{a'}) + u_{a'}$$

for each  $a, a' \in A$  so that  $a$  first node is 1 and  $v_{a'} = v_a$ .

By applying Fourier-Motzkin elimination, the following inequality replaces constraints (27) and (4):

$$D - \sum_{(i,j) \in E_a} l_{ij} \leq T(1 - z_{a'}) + u_{a'} \quad \forall a, a' \in A \text{ so that } a \text{ first node is 1 and } v_{a'} = v_a. \quad (28)$$

Summarizing, we obtain the following model:

$$\min \sum_{a \in A} w_a [(1 - z_a)T + u_a]$$

s.t. :

$$D - \sum_{(i,j) \in E_a} l_{ij} \leq T(1 - z_{a'}) + u_{a'}$$

$$\forall a, a' \in A \text{ so that } a \text{ first node is 1 and } v_{a'} = v_a. \quad (29)$$

$$u_a \geq 0 \quad \forall a \in A \quad (30)$$

$$z_a \in \{0, 1\} \quad \forall a \in A. \quad (31)$$

Let us now replace the terms  $l_{ij}$  using their definition. First, note that:

$$\begin{aligned} & \min_{a \in A: (i,j) \in arcC_a} (slack_{ij} + M_{ij}(1 - z_a)) \\ &= slack_{ij} + M_{ij} - \max_{a \in A: (i,j) \in arcC_a} M_{ij} z_a, \end{aligned}$$

and let us define  $\tilde{z}_{i,j} = \max_{a \in A: (i,j) \in arcC_a} z_a$ . Thus we can write the model above as:

$$\min \sum_{a \in A} w_a [(1 - z_a)T + u_a]$$

s.t.:

$$\begin{aligned} & Tz_{a'} - u_{a'} + \sum_{(i,j) \in arcC_a} M_{ij} \tilde{z}_{i,j} \\ & \leq T - D + \sum_{(i,j) \in E_a} slack_{ij} + \sum_{(i,j) \in arcC_a} M_{ij} \end{aligned}$$

$$\forall a, a' \in A \text{ so that } a \text{ first node is 1 and } v_{a'} = v_a. \quad (32)$$

$$u_a \geq 0 \quad \forall a \in A \quad (33)$$

$$z_a \in \{0, 1\} \quad \forall a \in A. \quad (34)$$

Unfortunately, because of the terms  $\tilde{z}_{i,j}$ , this model is not linear. However, if we look at an inequality of type (32), we see that if this inequality is true with the corresponding  $\tilde{z}_{i,j}$  for each  $(i, j) \in arcC_a$ , it is also true with, for each  $(i, j) \in arcC_a$ , the  $z_a : (i, j) \in arcC_a$  that are by definition smaller or equal to  $\tilde{z}_{i,j}$ .

Consequently we can transform the model above to obtain (MILP2). ■

**COROLLARY 1** *The (MILP1) and (MILP2) optimal values are identical.*

It is interesting to note that model (MILP2) has the following interpretation. Focussing on path  $a'$  and on variable  $u_{a'}$ , constraint (24) can be written as:

$$\begin{aligned} u_{a'} &\geq T(z_{a'} - 1) + D - \sum_{(i,j) \in E_a} slack_{ij} \\ &+ \sum_{(i,j) \in arcC_a} M_{ij} (\tilde{z}_{i,j} - 1) \end{aligned}$$

for all paths  $a$  s.t. the first node of  $a$  is 1 and  $v_{a'} = v_a$ .

If  $z_{a'} = 0$ , the constraint is redundant, and the contribution of path  $a'$  in the objective function is given by  $T$ .

If  $z_{a'} = 1$ , we have:

$$u_{a'} \geq D - \sum_{(i,j) \in E_a} \text{slack}_{ij} + \sum_{(i,j) \in \text{arc}C_a} M_{ij}(\tilde{z}_{i,j} - 1)$$

for all paths  $a$  s.t. the first node of  $a$  is 1 and  $v_{a'} = v_a$ .

We consider a “primary network” given by the path  $a$  and we refer the delay on  $a'$  to this primary network. Two situations can happen:

- If all connections on the primary network must be maintained, the delay for  $a'$  passengers is the same as that for  $a$  passengers (we recall that, by definition of the Arr-Arr graph, having the same destination node implies arriving at the same station with the same vehicle). Indeed, in this case  $\sum_{(i,j) \in \text{arc}C_a} M_{ij}(\tilde{z}_{i,j} - 1) = 0$  causing  $u_{a'} \geq D - \sum_{(i,j) \in E_a} \text{slack}_{ij}$ , which is the delay at path  $a$  destination.
- On the contrary, if at least one connection on the primary network is not maintained, then path  $a$  does not have any influence on the delay on  $a'$ . Hence, the constraint is redundant.

#### IV. APPLICATION

Model (MILP1) has been implemented using AMPL (see [4]) and solved using Cplex 8.1 (see [7]).

Note that model (MILP) has not been implemented in order to compare results with that of (MILP1) or (MILP2). Indeed, the formulation of (MILP) is almost the same as the one for (MILP1), except that it has a larger number of variables (corresponding to the departure delay variables) and some additional constraints (corresponding to the redundant constraint we have suppressed). Since both formulations provide the same LP-relaxation optimal value, it seems obvious that results obtained from (MILP1) are better in terms of computational times than those we would have obtained from (MILP). Further, this has been confirmed through preliminary computational experiments.

As model (MILP2) contains an exponential number of constraints, a direct implementation is not appropriate and a constraint generation approach is proposed. We start solving the (MILP2) with only constraints (25) and (26). Generally, the solution obtained will not be feasible, as some constraints (24) might be violated. We then select a subset of the violated constraints and we add them to the model. The constraints adding procedure is

repeated, until a solution is found which does not violate any constraints: this solution is also optimal for (MILP2).

For the sake of the constraint generation procedure, it is sufficient to generate inequalities (32) instead of inequalities (24). From (32), given the optimal solution  $z^*, u^*$  of the current model, a violated inequality exists if

$$\begin{aligned} & D - \sum_{(i,j) \notin \text{arc}C_a} \text{slack}_{ij} \\ & - \sum_{(i,j) \in \text{arc}C_a} [\text{slack}_{ij} + M_{ij}(1 - \tilde{z}_{i,j}^*)] \\ & > T(1 - z_{a'}^*) + u_{a'}^* \end{aligned}$$

for some  $a, a' \in A$  so that  $a$  first node is 1 and  $v_{a'} = v_a$ . By definition of  $l_{ij}$ , this corresponds to

$$D - \sum_{(i,j) \in E_a} l_{ij} > T(1 - z_{a'}^*) + u_{a'}^* \quad (35)$$

The term  $d_{v_a} = D - \sum_{(i,j) \in E_a} l_{ij}$  represents the delay at the last node of path  $a$ , for the given  $z^*$  and can be evaluated using a breath-first visit of the Arr-Arr graph (delay propagation on acyclic network).

The separation procedure computes the node delays and then check for a path satisfying condition (35), as from the following procedure.

- 1) For each  $(i, j) \in \text{arc}C$ , determine the variable  $\tilde{z}_{i,j}^*$ .
- 2) For each node of the Arr-Arr graph  $v \in V$ , compute the delay  $d_v^*$  according to the values  $u^*$  and  $\tilde{z}^*$  (breath-first visit of the Arr-Arr graph).
- 3) For each path  $a \in A$  whose first node is 1
- 4) For each path  $a'$  whose last node is  $v_{a'} = v_a$
- 5) If  $d_{v_a}^* > T(1 - z_{a'}^*) + u_{a'}^*$  then inequality (32) is violated for  $a, a'$ .
- 6) Go to the next path  $a'$ .
- 7) Go to the next path  $a$ .

Step 1 runs in  $O(|\text{arc}C||A|)$  time: for each connection, all the paths involving the connection itself are considered. Step 2 consists of a breath first visit of the Arr-Arr graph, whose complexity is  $O(|E|)$ . Steps from 3 to 7 run in  $O(|A|^2)$ . In terms of the DMP input, i.e. the vehicle set  $V$ , the station set  $S$  and the passenger path set  $A$ , we can state  $|\text{arc}C| = O(|V|^2|S|)$  and  $|E| = O(|S|^2 + |V|^2|S|)$ . It follows that the overall complexity of the separation procedure is  $O(|V|^2|S||A| + |S|^2 + |A|^2)$  and then polynomial.

At each iteration of the constraint adding procedure, all the violated inequalities generated by step 5 are added (computational results put in evidence that this is better than adding a subset of violated constraints).

TABLE I  
INSTANCE SUMMARY.

$ N $	$ arcDA $	$ arcC $	$ A $	Num.
16 - 20	11 - 16	3 - 11	62 - 132	28
25 - 40	20 - 33	3 - 14	99 - 549	88
59 - 101	55 - 93	10 - 22	1821 - 6366	12

Note that the constraint generation approach can be applied to both (MILP2) and its linear relaxation. In the former case, we obtain the optimal solution to the DMP, in the latter case the procedure should be integrated in a Branch and Bound framework.

The AMPL implementation of (MILP1) and the constraint generation approach for (MILP2) have been tested on the instances described in Table I, derived as subnetworks of the intermodal public transportation network around Brussels in Belgium. For each class, the minimum and the maximum cardinality of the set describing the DMP are given, together with the number of instances, in the last column. The slack times have been randomly generated as integer number between zero and three minutes. Initial delays  $D$  of 12, 20, 25 and 28 minutes have been considered for each instance and a time horizon  $T$  of 30 minutes. The number of passengers on each path has been obtained as follows: first we have randomly generated the origin/destination matrix; after that, we have used a logit function (see e.g. [3]) to distribute any entry of the matrix among the paths having the same origin and the same destination, this function taking in account the number of connection arcs and the total length of each path.

The results obtained on a 3 GHz Pentium IV processor are summarized in Table II. The first two columns give the instance size and the initial delay. The third and the fourth columns give the time spend to solve the linear relaxation of (MILP1) and (MILP2). Column 5 shows the percentage of time spent by the separation procedure. Columns 6 and 8 give the same information with respect to the original (MILP1) and (MILP2), including the integrality constraints. Columns from 9 to 11 give some statistics on the solution values: column 9 gives the percentage of maintained connections, column 10 refers to the percentage of instances containing suppressed connections and column 11 reports the percentage of suppressed connection within those instances. The last column show the integrality gap.

Both Cplex directly applied to (MILP1) and the constraint generation procedure applied to (MILP2) are able to solve the instances in a very reasonable amount of time: less than one second, both for the linear relaxation and the integer program. Indeed, Cplex opens a very

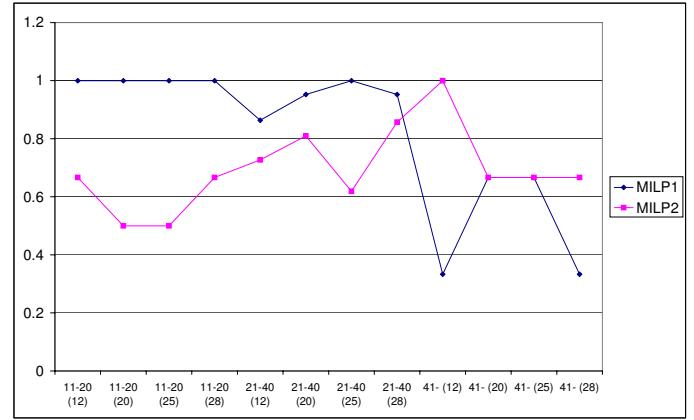


Fig. 3. Performance evaluation: linear relaxation.

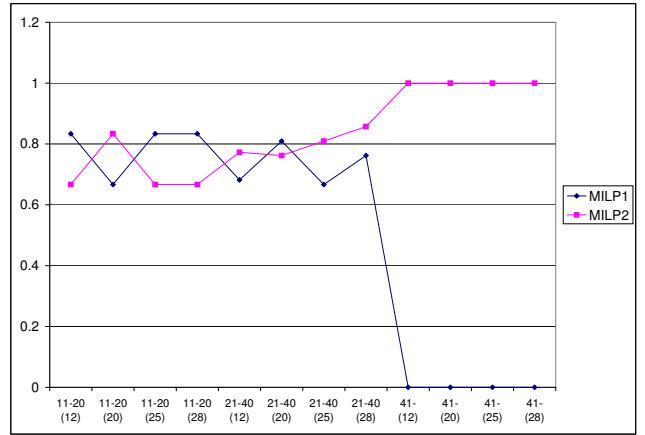


Fig. 4. Performance evaluation: integer program.

small number of branch and cut nodes when applied to (MILP1) and often closes the gap at the root node, using the built-in cuts. Also, six constraint generation iterations are sufficient in average to solve the linear relaxation of (MILP2), and four to solve the related integer program (we recall that in this case the Cplex branch and cut procedure is applied at each iteration).

The comparison between the performance of the two proposed models and related methods is highlighted in Figures 3 and 4: they show, for different instance classes, the frequency one method has been able to find the optimal solution faster (or within the same computational time) than the others. We can observe that MILP1 performs better for small instances, while the constraint generation approach tends to be better with instances of increasing size, despite of the fact that the the majority of the time is spent by the separation procedure. Actually, just a few iterations provide the optimal (relaxed or integer) solution: a very small subset of constraints (24)

TABLE II  
COMPUTATIONAL RESULTS.

Inst.	D	LR			IP						gap
		MILP1	MILP2	% sep.	MILP1	MILP2	% sep.	% z=1	p(z=0)	% z=0	
16-20	12	0.00	0.04	9.1%	0.01	0.01	0.0%	98.33%	16.67%	10.00%	11.67%
16-21	20	0.00	0.05	6.7%	0.01	0.03	11.1%	96.17%	33.33%	11.50%	6.55%
16-22	25	0.00	0.05	12.9%	'	0.04	0.0%	96.17%	33.33%	11.50%	4.82%
16-23	28	0.00	0.05	13.8%	0.01	0.03	5.0%	96.17%	33.33%	11.50%	3.51%
25-40	12	0.00	1.07	1.4%	0.03	0.64	1.6%	99.64%	4.55%	8.00%	3.16%
25-41	20	0.00	0.68	1.6%	0.02	0.81	0.9%	99.24%	4.76%	16.00%	1.16%
25-42	25	0.00	0.83	1.4%	0.03	0.72	1.0%	98.52%	4.76%	31.00%	0.85%
25-43	28	0.00	0.72	1.4%	0.03	0.70	1.2%	98.52%	4.76%	31.00%	0.81%
55-101	12	0.15	0.14	76.2%	0.45	0.14	76.7%	100.00%	0.00%	-	0.00%
55-102	20	0.15	0.15	68.9%	0.46	0.16	75.5%	100.00%	0.00%	-	0.00%
55-103	25	0.15	0.15	72.7%	0.41	0.14	69.8%	100.00%	0.00%	-	0.00%
55-104	28	0.15	0.14	72.1%	0.39	0.14	76.2%	100.00%	0.00%	-	0.00%

is added and considerably smaller problems have to be solved.

Concerning the value of the optimal solutions, we observe that, in most cases, all the connections are maintained. Moreover, in the instances where some connections are suppressed, it only concerns a small number of paths (from 10 to 30%, depending on the amount of the initial delay).

## V. CONCLUSIONS

This paper has presented two new MILP formulations for the Delay Management Problem. The first formulation (MILP1) is based on a new graph interpretation of the DMP, which allows us to reduce the number of variables and constraints with respect to equivalent models presented by literature. The second equivalent model (MILP2) further reduces the number of variables, by projecting out all the variables explicitly related to node delays. The cost to pay is the increasing number of constraints, which are exponentially many. The computational results presented in this paper show that the trade off between the number of variables and the number of constraints tends to privilege (MILP2) and the related constraint generation procedure when facing instance of greater size, similar to real public transportation networks. This suggests to further improve the solution approach based on (MILP2), in particular by enhancing the separation procedure (which takes most of the computational time and is presently based on path enumeration) and by integrating it in a Branch and Cut context.

Also note that both models (MILP1) and (MILP2), together with the related solution approaches, can be easily extended to the case of multiple initial delays. As the related model size increases, this also suggests to

adopt (and improve) constraint generation based solution methods.

From the public transportation network point of view, the results show that there is room for improving the connection protocols currently adopted by the carriers. In fact, the decisions about the suppression of a connection is mostly based on a threshold delay of the incoming vehicles. Results show that, at least from the passengers point of view, this is not always the best decision: most of connection should be maintained even under large vehicle delays, as this leads to smallest total passenger delays, thus considerably increasing the attractiveness of public transportation networks.

## REFERENCES

- [1] B. ADENZO-DIAZ, M. OLIVA-GONZALEZ, and P. GONZALEZ-TORRE, "On-line timetable re-scheduling in regional train services," *Transportation Research*, vol. 33B, pp. 387–398, 1999.
- [2] S. ELMAGHRABY, *Activity Networks*. Wiley Interscience Publication, 1977.
- [3] A. FOTHERINGHAM and M. O'KELLY, *Spatial Interaction Models: Formulations and Applications*. Kluwer Academic Publishers, 1989.
- [4] R. FOURER, D. GAY, and B. KERNIGHAN, "A modeling language for mathematical programming," *Management Science*, vol. 36, pp. 519–554, 1990.
- [5] A. GINKEL, "Event-activity networks in delay management," Master's thesis, Universität Kaiserslautern, 2001.
- [6] A. GINKEL and A. SCHÖBEL, "The bicriterial delay management problem," Universität Kaiserslautern, Tech. Rep., 2002.
- [7] ILOG, "<http://www.ilog.com/products/cplex>."
- [8] N. KLIEWER, "Mathematische Optimierung zur Unterstützung kundenorientierter Disposition im Schienengüterverkehr," Master's thesis, Universität Paderborn, 2000.
- [9] R. MARTIN, *Large scale linear and integer optimization*. Kluwer Academic Publishers, 1999, p. 39 à 46.
- [10] A. SCHÖBEL, "A model for the delay management problem based on mixed-integer-programming," *Electronic Notes in Theoretical Computer Science*, vol. 50, no. 1, 2001.
- [11] ———, "Customer-Oriented Optimization in Public Transportation," 2003, Habilitation thesis.

- [12] S. SCHOLL, "Anschluss sicherung bei Verspätungen im ÖPNV,"  
Master's thesis, Universität Kaiserslautern, 2001.

# A Metaheuristic Approach for the Vertex Coloring Problem

Enrico Malaguti\*, Michele Monaci<sup>†</sup> and Paolo Toth\*

\*Dipartimento di Elettronica, Informatica e Sistemistica, University of Bologna

Viale Risorgimento, 2 - 40136 - Bologna (Italy)

Emails: emalaguti@deis.unibo.it, ptot@deis.unibo.it

<sup>†</sup>Dipartimento di Ingegneria dell'Informazione, University of Padova

Via Gradenigo, 6/A - 35131 - Padova (Italy)

Email: monaci@dei.unipd.it

**Abstract**— Given an undirected graph, the Vertex Coloring Problem (VCP) requires to assign a color to each vertex in such a way that colors on adjacent vertices are different and the number of colors used is minimized. In this paper we propose a preliminary version of the metaheuristic Algorithm MMT for VCP. The proposed approach performs two phases: the first applies in sequence fast greedy heuristics and a Tabu Search procedure, while the second one is a post-optimization phase based on the Set Covering formulation of the problem. Computational results on the DIMACS set of instances show that the algorithm is able to produce high quality solutions in a reasonable amount of time.

**Keywords**— Vertex Coloring, Set Covering, Metaheuristic, Tabu Search.

## I. INTRODUCTION

**G**IVEN an undirected graph  $G = (V, E)$ , the Vertex Coloring Problem (VCP) requires to assign a color to each vertex in such a way that colors on adjacent vertices are different and the number of colors used is minimized.

Vertex Coloring is a well known NP-hard problem (see Garey and Johnson [20]) with real world applications in many engineering fields, including scheduling [25], timetabling [12], register allocation [10], frequency assignment [19] and communication networks [33]. This suggests that effective algorithms would be of great importance. Despite its relevance, few exact algorithms for VCP have been proposed, and are able to solve consistently only small instances, with up to 100 vertices for random graphs [13], [22], [30], [31]. On the other hand, several heuristic and metaheuristic algorithms have been proposed which are able to deal with graphs of hundreds or thousands of vertices. We review below, after some useful definitions, the most important classes of known heuristics and metaheuristics proposed for VCP.

Let  $n$  and  $m$  be the cardinalities of vertex set  $V$  and edge set  $E$ , respectively; let  $\delta(v)$  be the degree of a given vertex  $v$ . A subset of  $V$  is called an independent set if no two adjacent vertices belong to it. A clique of a graph  $G$  is a complete subgraph of  $G$ . A  $k$  coloring of  $G$  is a partition of  $V$  into  $k$  independent sets. An optimal coloring of  $G$  is a  $k$  coloring with the smallest possible value of  $k$  (the *chromatic number*  $\chi(G)$  of  $G$ ). The *chromatic degree* of a vertex is the number of different colors of its adjacent vertices.

The first approaches to VCP are based on greedy constructive algorithms. These algorithms sequentially color the vertices of the graph following some rule for choosing the next vertex to color and the color to use. They are generally very fast but produce poor results, which can be very sensitive to some input parameter, like the ordering of the vertices. Beyond the simple greedy sequential algorithm SEQ, the best known techniques are the *maximum saturation degree* DSATUR and the *Recursive Largest First* RLF procedures proposed by Brèlaz [5] and by Leighton [25], respectively (see section II-B for a short description of these algorithms). Culberson and Luo [11] proposed the *iterated greedy algorithm* IG which can be combined with various techniques. In [4] Bollobàs and Thomason proposed the algorithm MAXIS that recursively selects the maximum independent set from the set of uncolored vertices.

Many effective metaheuristic algorithms have been proposed for VCP. They are mainly based on simulated annealing (Johnson, Aragon, McGeoch and Schevon [22] compared different neighborhoods and presented extensive computational results on random graphs; Morgenstern [29] proposed a very effective neighborhood search) or Tabu Search (Hertz and De Werra [21]; Dorne and Hao [15]; Caramia and Dell'Olmo [7] proposed a local search with priorities rules, inspired from Tabu Search techniques). Funabiki and Higashino [17]

proposed one of the most effective algorithms for the problem, which combines a Tabu Search technique with different heuristic procedures, color fixing and solution recombination in the attempt to expand a feasible partial coloring to a complete coloring. Hybrid algorithms integrating local search and diversification via crossover operators were proposed (Fleurent and Ferland [16]; Galinier and Hao [18] proposed to combine an effective crossover operator with Tabu Search), showing that diversification is able to improve the performance of local search.

As a general observation, two main strategies can be identified in the literature, which correspond to different formulations of the problem. The first strategy tackles the problem in the most natural way, trying to assign a color to each vertex. This leads to fast greedy algorithms but seems to produce poor results. The second strategy tackles the problem of feasibly coloring the graph by partitioning the vertex set into independent sets. Algorithms based on this strategy build different color classes by identifying different independent sets in the graph, and try to cover all the vertices by using the minimum number of independent sets, i.e. by solving the Set Partitioning Problem (or the Set Covering Problem) associated with the identified independent sets.

#### A. Set Covering Formulation

Several combinatorial optimization problems can be formulated as large-size Set Covering (or Set Partitioning) problems; this happens for all those problems in which one is required to partition a given set of items into subsets having special features and minimizing the sum of the cost associated with the subsets. This can be done not only for VCP (see Merhotra and Trick [27]) but, for instance, for Bin Packing Problems [28], Vehicle Routing Problems [24], Crew Scheduling Problems [3], [6], [26], [32] as well.

We derive a Set Partitioning Formulation for VCP and show how it can be easily transformed into a Set Covering Formulation. Let  $\mathcal{S}$  be the family of all the Independent Sets of  $G$ . Each independent set (column)  $s \in \mathcal{S}$  has associated a binary variable  $x_s$  having value 1 iff all the vertices of  $s$  receive the same color. VCP can be formulated as the following Set Partitioning Problem:

$$\min \sum_{s \in \mathcal{S}} x_s \quad (1)$$

$$\sum_{s: i \in s} x_s = 1 \quad \forall i \in V \quad (2)$$

$$x_s \in \{0, 1\} \quad (s \in \mathcal{S}) \quad (3)$$

Objective function (1) asks to minimize the total number of independent sets (and hence of colors) used. Constraints (2) state that for every vertex  $i$  in the graph, there must exist exactly one selected independent set which contains the vertex. Constraint (3) impose variables  $x_s$  to be binary. We can now replace constraints (2) with (4) and obtain a Set Covering Formulation for the problem.

$$\sum_{s: i \in s} x_s \geq 1 \quad \forall i \in V \quad (4)$$

Indeed, if a solution selects more than one independent set which contains the same vertex, a feasible solution of the same value can be obtained removing the vertex from all except one of these independent sets. In others words if a vertex is assigned more than one color, a feasible solution of the same value can be obtained using any one of these colors for the vertex. The set covering formulation allows us to consider a smaller (but still exponential in the worst case) number of variables, since we can define  $\mathcal{S}$  as the family of all *maximal* independent sets in the graph  $G$ . The advantage of the Set Covering (or Set Partitioning) formulation, w.r.t different proposed formulations, is that it avoids symmetries in the solution and its continuous relaxation leads to tighter lower bounds. The main drawback is that the number of maximal independent sets (i.e. the number of columns) can grow exponentially with the cardinality of vertex set  $V$ .

## II. THE HEURISTIC ALGORITHM MMT

This paper proposes a preliminary version of the MMT algorithm, which performs 2 phases after an initialization step. During the initialization some fast lower bounding procedures are applied to derive a lower bound (LB) for the problem; in the first phase we do not use an explicit algorithm to generate columns, but we apply in sequence some fast greedy heuristics from the literature, possibly considering different parameter sets. Indeed, each feasible independent set in any heuristic solution of the original problem corresponds to a column of  $\mathcal{S}$ . It is known that the solution found by a greedy algorithms for VCP depends on the order in which vertices are given in input. Since the column generation phase is aimed at generating a large set of different columns, in our approach greedy procedure are applied several times, in an iterative way, perturbing the order of vertices so that different columns are generated. After that, an effective Tabu Search algorithm, based on the concept of partitioning the vertex set into independent sets, is executed. This algorithm works in decision version (i.e.,

given as input the number  $k$  of colors to use, it looks for a  $k$  coloring in the graph  $G$ ), trying to improve on the best valued solution found by the greedy procedures previously executed. Sometimes the Tabu Search algorithm is able to find a provably optimal solution; in any case, this algorithm often improves the best incumbent solution. During phase 1 (*Column Generation*), a very large number of independent sets (*columns*) is produced. When optimality of the incumbent solution is not proved, such columns are stored in the family  $\mathcal{S}'$ , which represents a subfamily of the family  $\mathcal{S}$  of all the maximal independent sets of the graph. The second phase (*Column Optimization*) considers the Set Covering Problem (SCP) associated with the columns in  $\mathcal{S}'$  and heuristically solves it through the Lagrangian heuristic algorithm CFT proposed by Caprara, Fischetti and Toth [6], improving many times the best incumbent solution.

The main drawbacks of this approach are evident:

- a lot of independent sets which are not maximal are generated;
- the same independent set can be generated at different times, thus producing many redundant columns.

The first problem is structural to our approach, since we extract independent sets from feasible colorings. This drawback is solved by applying a greedy procedure to complete independent sets to maximal independent sets. As to the second problem, a hashing technique is used in order to avoid to store identical columns (see Monaci and Toth [28] for more details).

Both phases can be stopped as soon as a solution which is proven to be optimal is found, i.e., if the cost of the best solution found so far is equal to a lower bound for the original problem.

The overall algorithm MMT is structured as follows:

**begin**

*Initialization Step*

1. Compute lower bound  $LB$ ;
2. Set  $\mathcal{S}' = \emptyset$ ;

*Phase 1: Column Generation*

3. Apply greedy heuristics, update  $UB$  and  $\mathcal{S}'$ ;
4. if  $LB = UB$  **stop**;
5.  $k := UB - 1$ ;
6. **while**  $k \geq LB$
7.     apply the Tabu Search Algorithm;
8.     update  $UB$  and  $\mathcal{S}'$ ;
9.     if no feasible solution of value  $k$  has been found **then break**;
10.     $k := k - 1$
11. **endwhile**;

*Phase 2: Column Optimization*

12. apply heuristic algorithm CFT to the Set Covering

instance corresponding to subfamily  $\mathcal{S}'$  with a given time limit (possibly updating  $UB$ );  
**end.**

*A. Initialization Step: Lower Bounding*

As lower bound  $LB$  we use the cardinality of a maximal clique  $K$  of  $G$ .

Although this is the simplest lower bound for the problem, the execution of our algorithm does not depend on  $LB$ . Thus, stronger lower bounds could affect only the stopping criterion of the algorithm. However, the computing times of our algorithm on the test bed of instances considered (see Section IV) are quite small, and would be not reduced by computing better lower bounds (which would be large time consuming, see for instance Caramia and Dell'Olmo [8], [9]).

We compute  $LB$  as the maximum cardinality of the maximal cliques of  $G$  obtained by executing several times (say 10), with different random orderings of the vertices, the following greedy algorithm, which defines a maximal clique  $K$ . Let  $v_i$  be the  $i$ -th vertex of the considered ordering,  $LB$  the incumbent value of the lower bound and  $\eta(v_i)$  the number of vertices in  $K$  that are adjacent to  $v_i$ . While the incumbent clique  $K$  can be expanded (line 2), we insert in  $K$  the first vertex of maximum degree of the considered ordering, if this insertion will improve on the best incumbent  $LB$ :

**begin**

1.  $K = \emptyset$ ;
2. **while** ( $|K| = \max_{i:v_i \in V \setminus K} \eta(v_i)$ )
3.      $j = \min(\arg \max_{i:v_i \in V \setminus K} \text{and } \delta(v_i) \geq LB \eta(v_i))$ ;
4.     **if** no such  $j$  exists **then break**;
4.      $K := K \cup \{v_j\}$
7. **end while**;

**end.**

*B. Phase 1: Heuristic Generation*

It is known that the solution found by a greedy algorithm for VCP depends on the order in which vertices are given in input. In our approach we perform several iterations (say 500) of the greedy procedures SEQ, DSATUR, RLF [22] with different random orderings of the vertices.

SEQ is the simplest greedy algorithm for VCP. Assume that the vertices are labelled  $v_1, \dots, v_n$ . Vertex  $v_1$  is assigned to the first color class, and thereafter, vertex  $v_i$  ( $i = 1, \dots, n$ ) is assigned to the lowest indexed color class that contains no vertices adjacent to  $v_i$ . Generally, algorithm SEQ does not procedure solutions of high quality; however, this algorithm is very fast and

it generates independent sets that are useful in phase 2 of the algorithm.

DSATUR [5], [22] is similar to SEQ, but dynamically chooses the vertex to color next, picking the first vertex that is adjacent to the largest number of distinctly colored vertices (i.e. the vertex with maximum chromatic degree).

The *Recursive Largest First* (RLF) algorithm [22], [25] colors the vertices one class at a time, in the following greedy way. Let  $C$  be the next color class to be constructed,  $V'$  the set of uncolored vertices that can legally be placed in  $C$ , and  $U$  the set (initially empty) of uncolored vertices that cannot legally be placed in  $C$ .

- Choose the first vertex  $v_0 \in V'$  that has the maximum number of adjacent vertices in  $V'$ . Place  $v_0$  in  $C$  and move all the vertices  $u \in V'$  that are adjacent to  $v_0$  from  $V'$  to  $U$ .
- While  $V'$  remains nonempty, do the following: choose the first vertex  $v \in V'$  that has the maximum number of adjacent vertices in  $U$ ; add  $v$  to  $C$  and move all the vertices  $u \in V'$  that are adjacent to  $v$  from  $V'$  to  $U$ .

### C. Phase 1: Tabu Search Algorithm

To find high quality columns and improve on the solutions found by the greedy heuristics, we use a Tabu Search procedure, a metaheuristic technique that showed a very good experimental behavior on hard combinatorial optimization problems.

A local search procedure can be seen as the result of three main components:

- the definition of a solution  $S$ ;
- the solution evaluating function  $f(S)$ ;
- the solution neighborhood  $N(S)$ .

In the simple local search procedures, given a solution  $S$  the algorithm explores its neighborhood  $N(S)$  and moves to the best (according to the evaluating function  $f(S)$ ) improving solution  $S' \in N(S)$ . If a solution  $S$  is the best of its neighborhood, i.e. it is a local optimum, the local search algorithm is not able to move and the search is stopped. In Tabu Search procedures, to avoid local optimum traps, the algorithm moves to the best solution  $S'$  in the neighborhood, even if it is not improving the current solution. To avoid cycling, some attributes of solution  $S'$  are stored in a *Tabu List*; for a specified number of iterations (the so called *Tabu Tenure*) a solution which presents tabu attributes is declared tabu and is not considered, except in the case it would improve the best incumbent solution (*aspiration criterion*). Most of the Tabu Search algorithms proposed so far for VCP move between infeasible solutions, i.e. they partition the

set  $V$  in subsets which are not necessary independent sets, trying to reduce the number of infeasibilities in every subset. Following an idea by Morgenstern [29], we propose a Tabu Search procedure which moves between *partial feasible colorings*, i.e. solutions in which each vertex subset is an independent set but not all vertices are assigned to subsets. In [29] Morgenstern defines the *Impasse Class Neighborhood*, a structure used to improve a partial  $k$  coloring to a complete coloring of the same value. The *Impasse Class* requires a target value  $k$  for the number of colors to be used. A solution  $S$  is a partition of  $V$  in  $k+1$  color classes  $\{V_1, \dots, V_k, V_{k+1}\}$  in which all classes, but possibly the last one, are independent sets. This means that the first  $k$  classes constitute a partial feasible  $k$  coloring, while all vertices that do not fit in the first  $k$  classes are in the last one. Making this last class empty gives a complete feasible  $k$  coloring. To move from a solution  $S$  to a new solution  $S' \in N(S)$  one can randomly choose an uncolored vertex  $v \in V_{k+1}$ , assign  $v$  to a different color class, say  $h$ , and move to class  $k+1$  all vertices  $v'$  in class  $h$  that are adjacent to  $v$ . This assures that color class  $h$  remains feasible. Class  $h$  is chosen by comparing different target classes by mean of the evaluating function  $f(S)$ . Rather than simply minimizing  $|V_{k+1}|$  it seems a better idea to minimize the value:

$$f(S) = \sum_{w \in V_{k+1}} \delta(w) \quad (5)$$

This forces vertices having small degree, which are easier to color, to enter class  $k+1$ . Morgenstern uses this idea, together with a procedure for the recombination of the solutions, to build a simulated annealing algorithm. We use the same idea within a Tabu Search approach. At every iteration we move from a solution  $S$  to the best solution  $S' \in N(S)$  (even if  $f(S) < f(S')$ ). To avoid cycling, we use the following tabu rule: a vertex  $v$  cannot take the same color  $h$  it took at least one of the last  $T$  iterations; for this purpose we store in a tabu list the pair  $(v, h)$ . While pair  $(v, h)$  remains in the tabu list, vertex  $v$  cannot be assigned to color class  $h$ . We also use an *Aspiration Criterion*: a tabu move can be performed if it improves on the best solution encountered so far. A Tabu Search algorithm based on the same neighborhood structure was experimented by Blöchliger and Zufferey [2]: in this work the next vertex to color is not chosen randomly, but selected so that it, entering the best color class, produces the best solution in the neighborhood. This approach explores the all neighborhood reducing at the same time the randomness introduced in the search. Thus, to maintain some randomness, the authors use an evaluating function that simply minimizes  $|V_{k+1}|$ .

Our Tabu Search algorithm takes in input:

- graph  $G(V, E)$ ;
- the target value  $k$  for the coloring;
- a feasible partial  $k$  coloring;
- the maximum number  $L$  of iterations to be performed ;
- the tabu tenure  $T$ .

If the algorithm solves the problem within  $tabuiter$  iterations it gives on output the feasible coloring of value  $k$ , otherwise it gives in output the best scored partial coloring found during the search.

Let  $S$  be the current solution and  $S^*$  the best incumbent solution. The Tabu Search algorithm works as follows:

```

begin
1. initialize a solution  $S = \{V_1, \dots, V_k, V_{k+1}\}$ ;
2.  $S^* := S$ ;
3.  $tabulist := \emptyset$ ;
4. for ( $iterations = 1$  to  $L$ )
5.   randomly select an uncolored vertex  $v \in V_{k+1}$ ;
6.   for each  $j \in \{1, \dots, k\}$  (explore  $N(S)$ )
7.      $V'_j := V_j \setminus \{w \in V_j : (v, w) \in E\} \cup \{v\}$ ;
8.      $V'_{k+1} := V_{k+1} \setminus \{v\} \cup \{w \in V_j : (v, w) \in E\}$ ;
9.      $S_j = S \setminus \{V_j, V_{k+1}\} \cup \{V'_j, V'_{k+1}\}$ 
10.    end for;
11.     $h := \arg \min_{j:(v,j) \notin tabulist \text{ or } f(S_j) < f(S^*)} f(S_j)$ ;
12.    if no such  $h$  exists then
13.       $h := \arg \min_{j \in \{1, \dots, k\}} f(S_j)$ ;
14.     $S := S_h$ ;
15.    insert  $(v, h)$  in  $tabulist$ ,
         $(v, h)$  is tabu for  $T$  iterations;
16.    if  $f(S) < f(S^*)$  then  $S^* := S$ ;
17.    if  $V_{k+1} = \emptyset$  then return  $S^*$ 
18.  end for;
19. return  $S^*$ 
end.
```

At line 10 we try to select the best color class which improves on the best solution so far or does not represent a tabu move. If all moves are tabu, at line 11 we simply select the best color class.

Our Tabu Search algorithm is very simple and requires as parameter to be experimentally tuned only the tabu tenure  $T$ . At the same time it has a good experimental behavior, since it is often able to find good solutions in very short computing times (see section IV-A). All experiments were performed starting the computation with the best solution found by the greedy algorithms previously executed. Preliminary computational experiments (refer to the full paper for a deeper analysis) show that the algorithm generally needs a small number of iterations

to solve the problem, and when this does not occur, seldom the algorithm is able to solve the problem even if a bigger number of iterations is allowed. In particular it seems that the Tabu Search is not able to move from a region of the solution space to explore the whole solution space. This behavior can be explained by the aggressive strategy adopted, which should make the Tabu Search much useful if initialized with a good quality solution or combined with a suitable diversification strategy.

### III. PHASE 2: COLUMN OPTIMIZATION

If the incumbent solution found in phase 1 is not proved to be optimal, phase 2 is executed in order to improve the value of the solution. This phase is based on the Integer Linear Programming formulation of VCP presented in Section I-A, which is solved through the heuristic algorithm CFT [6]. This iterative algorithm can handle very large Set Covering instances, producing good (possibly optimal) solutions within a reasonable amount of computing time. Moreover, algorithm CFT computes an “internal” lower bound (not valid for VCP) on the value of the optimal solution of the corresponding Set Covering instance and its execution can be stopped as soon as this lower bound equals the value of the best incumbent solution for VCP. Of course optimality for SCP does not imply optimality for the original problem, because we do not enumerate all the independent sets of  $G$ .

Generally the global number of independent sets (columns) generated in phase 1 is very large and could ask for excessive memory requirements and computing time to perform the hashing. Hence we decided to consider as candidate for insertion in  $\mathcal{S}'$  only the independent sets generated by the initial greedy algorithms and those corresponding to the feasible solutions found by the Tabu search algorithm.

Computational experiments showed that this choice, that privileges independent sets corresponding to solutions which tend to have high diversity each other (on the complete set of instances, approximately 50% of the independent sets generated are stored in  $\mathcal{S}'$  after the execution of the hashing procedure), did not affect the effectiveness of phase 2 while reducing considerably the computation time and avoiding memory problems.

### IV. COMPUTATIONAL ANALYSIS

The Tabu Search algorithm described in section II-C was coded in ANSI C and compiled with full optimization option; all other procedures, including algorithm CFT [6], were coded in ANSI FORTRAN77 and compiled with full optimization option. The programs

were run on a PIV 2.4MHz with 512MB RAM under Windows XP and tested on the *DIMACS benchmark graph instances* [1], [23]. These instances correspond to different graph types used for evaluating the performance of VCP algorithms. In particular this set of instances contains random graphs (DSJCn.x), geometric random graphs (DSJRn.x and Rn.x[c]), “quasi-random” graphs (flatn\_x\_0), artificial graphs (len\_x and latin\_square\_10), graphs from real life applications (school1 and school1\_nsh). All the computing times reported in this section are expressed in seconds of a PIV 2.4GHz. To allow a meaningful - although approximate - comparison on results obtained with different machines a benchmark program (dfmax), together with a benchmark instance (r500.5), are available. Computing times obtained on different machines can be scaled w.r.t. the performance obtained on this program (our machine spent 7 seconds user time). To perform our computational experiments we selected the subset of DIMACS instances considered by the papers describing the most effective heuristic algorithms for VCP.

#### A. Performance of the Overall MMT Algorithm

In this section we report the experimental results obtained with the preliminary version of the MMT Algorithm. Since our algorithm uses random numbers, we performed 4 runs with 4 different seeds for the random number generator. The corresponding computational results are reported in Table I. Since the overall algorithm works in optimization version, it always gives on output a feasible solution and it does not require a target value  $k$  as input. In all the experiments, the Tabu Search was initialized with the best solution found in phase 1, the tabu tenure for every pair  $(v, h)$  was composed by a constant component of 30 and a random component with uniform distribution between 0 and 10 (an ad hoc tuning of this parameter for each instance did not produce considerable improvements in the best solutions obtained, so we preferred to use a value which gave robust behavior on all the set of instances), the number of Tabu Search iterations was 10000 times  $n$  for each considered graph. The time limit of phase 2 was set equal to 100 seconds. For every instance, we report the best known solution value ever found in the literature (in bold when it corresponds to the optimal value), the lower bound  $LB$  computed during the initialization, the average solution value after phase 1, the best solution value after phase 1, the average solution value after phase 2, the best solution value after phase 2 (these values are reported only if optimality is not proven during phase 1 and phase 2 is executed) and the average total computing time.

When optimality is not proven in phase 1 (i.e. when the solution value after phase 1 is greater than LB) and phase 2 does not improve the incumbent solution, the Tabu Search algorithm has performed one useless iteration up to the iteration limit, spending an important amount of computing time, after the last successful iteration; e.g. for instance DSJC125.1 the final solution is found on average after less than 1 second, but optimality is not proven and 4 seconds are spent trying to improve on this solution. On the contrary, when phase 2 improves the solution or optimality is proven in phase 1 (which happens for all the le450\_5x, le450\_15x instances and for school1), the time of the last improvement corresponds to the total computing time.

The main aspect turning out from Table I is the effectiveness of phase 2, which 8 times is able to improve the solution of phase 1. In particular phase 2 brings the MMT Algorithm to solve for the first time, to proven optimality, instance r1000.5. In synthesis, algorithm MMT, on the complete set of the 42 considered instances, 1 time improves on the best known solution in the literature, 31 times finds the best known solution in the literature and for 10 instances finds a worse solution.

#### B. Comparison with the most effective heuristic algorithms

In Tables II we compare the performance of our algorithm with the heuristic algorithms that, to the best of our knowledge, represent the state of the art for VCP. For every considered instance, we report the value of the best known solution found in the literature (in bold when it is the proven optimal value). All these solutions were found by the algorithms considered in our comparison. We report in Table II the computational results of:

- The *Impasse* algorithm by Morgenstern [29], which is actually composed by tree different algorithms based on the idea of *Impasse Class Neighborhood*. These algorithms work in decision version and require as input the target value  $k$  for the coloring and a couple of other parameters that are tuned for every instance. For each instance considered in [29] we report the smallest value of  $k$  for which no failure occurred over 5 runs and the average running time to solve the instance, scaled w.r.t the benchmark problem.
- The HCA (Hybrid Coloring Algorithm) by Galinier and Hao [18]. HCA requires as input the target value  $k$  for the coloring and a couple of others parameters that are tuned for every instance. For each instance considered in [18] we report the smallest value of  $k$  for which there was at least one

TABLE I  
PERFORMANCE OF THE ALGORITHM MMT.

Instance name	n	m	best ( $\chi$ )	LB	avg k phase 1	best k phase 1	avg k phase 2	best k phase 2	avg total time
DSJC125.1	125	736	5	4	5.00	5	5.00	5	5
DSJC125.5	125	3891	17	9	17.00	17	17.00	17	108
DSJC125.9	125	6961	44	32	44.00	44	44.00	44	110
DSJC250.1	250	3218	8	4	8.00	8	8.00	8	9
DSJC250.5	250	15668	28	10	29.00	29	29.00	29	51
DSJC250.9	250	27897	72	36	72.75	72	72.75	72	96
DSJC500.1	500	12458	12	5	13.00	13	13.00	13	60
DSJC500.5	500	62624	48	11	51.25	51	51.25	51	202
DSJC500.9	500	112437	127	44	130.00	129	128.00	128	661
DSJC1000.1	1000	49629	20	5	21.00	21	21.00	21	150
DSJC1000.5	1000	249826	83	12	92.50	92	92.50	92	1104
DSJC1000.9	1000	449449	224	52	271.75	263	225.00	225	4419
DSJR500.1	500	3555	<b>12</b>	12	<b>12.00</b>	<b>12</b>			0
DSJR500.1C	500	121275	85	69	87.25	85	85.00	85	434
DSJR500.5	500	58862	<b>122</b>	121	124.00	124	<b>122.00</b>	<b>122</b>	416
le450_15a	450	8168	<b>15</b>	15	<b>15.00</b>	<b>15</b>			12
le450_15b	450	8169	<b>15</b>	15	<b>15.00</b>	<b>15</b>			12
le450_15c	450	16680	<b>15</b>	15	<b>15.00</b>	<b>15</b>			22
le450_15d	450	16750	<b>15</b>	15	<b>15.00</b>	<b>15</b>			23
le450_25c	450	17343	26	25	26.00	26	26.00	26	134
le450_25d	450	17425	26	25	26.00	26	26.00	26	134
le450_5a	450	5714	<b>5</b>	5	<b>5.00</b>	<b>5</b>			7
le450_5b	450	5734	<b>5</b>	5	<b>5.00</b>	<b>5</b>			0
le450_5d	450	9757	<b>5</b>	5	<b>5.00</b>	<b>5</b>			0
r125.1	125	209	<b>5</b>	5	<b>5.00</b>	<b>5</b>			0
r125.1c	125	7501	<b>46</b>	44	<b>46.00</b>	<b>46</b>			9
r125.5	125	3838	<b>36</b>	36	<b>36.00</b>	<b>36</b>			1
r250.1	250	867	<b>8</b>	8	<b>8.00</b>	<b>8</b>			0
r250.1c	250	30227	<b>64</b>	59	<b>64.00</b>	<b>64</b>			53
r250.5	250	14849	<b>65</b>	65	66.00	66	<b>65.00</b>	<b>65</b>	50
r1000.1	1000	14378	<b>20</b>	20	<b>20.00</b>	<b>20</b>			16
r1000.1c	1000	485090	98	73	98.25	98	98.00	98	1979
r1000.5	1000	238267	237	234	238.0	238	235.00	<b>234</b>	2298
school1	385	19095	<b>14</b>	14	14.00	<b>14</b>			0
school1_nsh	352	14612	<b>14</b>	14	14.00	<b>14</b>			0
latin_square_10	900	307350	99	90	108.75	108	101.75	101	1435
flat300_20_0	300	21375	<b>20</b>	10	<b>20.00</b>	<b>20</b>			40
flat300_26_0	300	21633	<b>26</b>	10	<b>26.00</b>	<b>26</b>			42
flat300_28_0	300	21695	<b>28</b>	10	31.75	31	31.75	31	70
flat1000_50_0	1000	245000	<b>50</b>	13	<b>50.00</b>	<b>50</b>	<b>50.00</b>	<b>50</b>	1024
flat1000_60_0	1000	245830	<b>60</b>	12	<b>60.00</b>	<b>60</b>	<b>60.00</b>	<b>60</b>	1107
flat1000_76_0	1000	246708	83	12	91.25	91	91.25	91	1086

successful run over 10 (or 5) runs (succ.), and an approximate average running time for the successful runs (we divided the reported time, obtained on an UltraSPARC-IIi 333MHz with 128MB RAM, for which the authors do not report the performance on the benchmark problem, by 6, following the performance ratio reported by Dongarra [14] for machines similar to those used in [18] and in our experiments). We report also the best solution values found during the complete set of computational experiments performed on the algorithm (for which the computing times and the number of successful

runs are not given).

- The PC (Partialcol) and RPC (React-Partialcol) algorithms by Blöchliger and Zufferey [2] are Tabu Search algorithms, based on the idea of *Impasse Class Neighborhood* [29], which implement a dynamic and reactive tabu tenure, respectively, and require as input parameter only the target value  $k$  for the coloring. For each instance considered in [2] we report the smallest value of  $k$  for which there was at least one successful run over 10 runs (succ.) of each of the two algorithms, and an approximate average running time for the successful runs of the

best algorithm (considering as best the algorithm which finds the best solution value, breaking ties by considering first the number of successful runs and then the computing time). The computational experiments were carried out on different Linux systems mostly running on a PIV 2GHz with 512MB RAM, whose performance is similar (and directly comparable) to that of the machine used in our experiments.

- The MIPS-CLR (MInimal-state Processing Search algorithm for the graph CoLoRing problem) by Funabiki and Higashino [17]. This algorithm works in optimization version but requires as input the target value  $k_{init}$  for the coloring, together with some other parameters whose values are tuned for hard instances. If the algorithm is not able to solve the problem with  $k_{init}$  colors, it dynamically modifies this target value. We report the results obtained giving the target value  $k_{init}$  as external input, the best value ( $bestk$ ) found over the 5 runs, the average solution value ( $avgk$ ) and the average running time approximately scaled w.r.t the benchmark problem (we run the benchmark problem on a machine similar to the one used in [17], which spent 17 seconds user time to solve the benchmark problem).
- The MMT algorithm, whose performance is summarized reporting the best value ( $bestk$ ) over 4 runs, the average value of  $k$  ( $avgk$ ) and the average computing time up to the time limit or to proven optimality.

To compare with a single index the performance of the different algorithms considered in this paper, we compute the average ratio between the solution value and the best known solution value from the literature  $k/best$ . This ratio always refers to the best results reported, for the corresponding instance, in the associated paper (i.e.  $k$  for Impasse [29], HCA [18] and PC-RPC [2],  $bestk$  for MIPS-CLR [17]) and the best solution value found by the Tabu Search algorithm, and algorithm MMT. Since Morgenstern [29], Galinier and Hao [18] and Blöchliger and Zufferey [2] did not consider the entire set of instances, in Table III we compare our results with those of the other algorithms on the common subset of instances. Tables II and III confirm the effectiveness of this preliminary version of algorithm MMT which produces, within acceptable computing time, solutions slightly worse (in particular for random instances) than those of the most effective algorithms for VCP. However, it has to be pointed that algorithm MMT does not require a fine tuning of many parameters, as is the case for most of the algorithms for VCP in the literature.

## V. CONCLUSIONS

In this paper we presented the two phases metaheuristic algorithm MMT for the Vertex Coloring Problem. The first phase of MMT applies in sequence fast greedy heuristics and a Tabu Search procedure; the second phase is a post optimization phase based on the Set Covering formulation of the problem.

Extensive computational experiments performed on 42 hard instances from the well known DIMACS benchmark graph instances show the effectiveness of the approach and the capacity of phase 2 of improving the solution of phase 1 in 8 instances. In particular phase 2 brings the MMT Algorithm to solve for the first time, to proven optimality, instance r1000.5. The best known solution values are still found in 31 of the remaining 41 instances.

The main deficiency of our approach is that, when optimality is not proven, the Tabu Search algorithm continues the computation up to the iteration limit, even if the problem is yet solved or the search region has been deeply explored. In particular it seems that the Tabu Search is not able to move from a region of the solution space to explore the whole solution space. Thus, future work will deal with the development of new effective procedures to diversify the search performed by the Tabu Search.

## ACKNOWLEDGEMENTS

The authors would like to thank Luca Agostini for his help in programming.

## REFERENCES

- [1] Ftp://dimacs.rutgers.edu/pub/challenge/graph/.
- [2] I. Blöchliger and N. Zufferey, "A reactive tabu search using partial solutions for the graph coloring problem," Ecole Polytechnique Fédérale de Lausanne (EPFL), Recherche Opérationnelle Sud-Est (ROSE), CH-1015 Lausanne, Switzerland, Tech. Rep. 04/03, 2004.
- [3] L. Bodin, B. Golden, A. Assad, and M. Ball, "Routing and scheduling of vehicles and crews: the state of the art," *Computers and Operations Research*, vol. 10, pp. 63–211, 1983.
- [4] B. Bollobás and A. Thomason, "Random graphs of small order," *Annals of Discrete Mathematics*, vol. 28, pp. 47–97, 1985.
- [5] D. Brélaz, "New methods to color the vertices of a graph," *Communications of the ACM*, vol. 22, no. 4, pp. 251–256, 1979.
- [6] A. Caprara, M. Fischetti, and P. Toth, "A heuristic method for the set covering problem," *Operations Research*, vol. 47, pp. 730–743, 1999.
- [7] M. Caramia and P. Dell'Olmo, "A fast and simple local search for graph coloring," *Proc. of the 3d Workshop on Algorithm Engineering WAE'99, Lecture Notes in Computer Science*, vol. 1668, pp. 319–313, 1999.
- [8] ———, "Constraint propagation in graph coloring," *Journal of Heuristics*, vol. 8, pp. 83–107, 2002.
- [9] ———, "Bounding vertex coloring by truncated multistage branch and bound," *Networks*, pp. 231–242, 2004.

TABLE II  
PERFORMANCE OF THE MOST EFFECTIVE HEURISTICS.

Instance name	best	Impasse [29]		HCA [18]		PC-RPC [2]			MIPS-CLR [17]			MMT								
		k	time	succ.	k	time	succ	k	time	avg	k	best	k	time	avg	k	best	k	time	
DSJC125.1	5									5.0	5	0	5.00	5	5					
DSJC125.5	17	17	1							17.0	17	1	17.00	17	108					
DSJC125.9	44									44.0	44	0	44.00	44	110					
DSJC250.1	8									8.0	8	5	8.00	8	9					
DSJC250.5	28	28	22	9/10	28	13				28.4	28	14	29.00	29	51					
DSJC250.9	72									72.4	72	31	72.75	72	96					
DSJC500.1	12						10/10	12	120	12.4	12	84	13.00	13	60					
DSJC500.5	48	49	660	5/10	48	268	1/10	49	720	49.4	49	349	51.25	51	202					
DSJC500.9	127						2/10	127	1560	127.8	127	480	128.00	128	661					
DSJC1000.1	20			???	20	?	1/10	20	2640	21.0	21	90	21.00	21	150					
DSJC1000.5	83	89	1148	???	83	2258	2/10	88	14400	89.0	88	4658	92.50	92	1104					
DSJC1000.9	224			???	224	?	4/10	226	18000	229.6	228	1565	225.00	225	4419					
DSJR500.1	<b>12</b>	<b>12</b>	0							<b>12.0</b>	<b>12</b>	0	<b>12.00</b>	<b>12</b>	0					
DSJR500.1C	85	85	5							85.0	85	6	85.00	85	434					
DSJR500.5	<b>122</b>	123	14							123.4	<b>122</b>	276	<b>122.00</b>	<b>122</b>	416					
le450_15a	<b>15</b>	<b>15</b>	0							<b>15.0</b>	<b>15</b>	1	<b>15.00</b>	<b>15</b>	12					
le450_15b	<b>15</b>	<b>15</b>	0							<b>15.0</b>	<b>15</b>	1	<b>15.00</b>	<b>15</b>	12					
le450_15c	<b>15</b>	<b>15</b>	5	6/10	<b>15</b>	8	10/10	<b>15</b>	2	15.2	<b>15</b>	11	<b>15.00</b>	<b>15</b>	22					
le450_15d	<b>15</b>	<b>15</b>	3				10/10	<b>15</b>	8	<b>15.0</b>	<b>15</b>	5	<b>15.00</b>	<b>15</b>	23					
le450_25c	26			10/10	26	55	10/10	27	1	26.0	26	7	26.00	26	134					
le450_25d	26						10/10	27	1	26.4	26	1	26.00	26	134					
le450_5a	<b>5</b>									<b>5.0</b>	<b>5</b>	1	<b>5.00</b>	<b>5</b>	7					
le450_5b	<b>5</b>									<b>5.0</b>	<b>5</b>	2	<b>5.00</b>	<b>5</b>	0					
le450_5d	<b>5</b>									<b>5.0</b>	<b>5</b>	3	<b>5.00</b>	<b>5</b>	0					
r125.1	<b>5</b>	<b>5</b>	0							<b>5.0</b>	<b>5</b>	0	<b>5.00</b>	<b>5</b>	0					
r125.1c	<b>46</b>	<b>46</b>	0							<b>46.0</b>	<b>46</b>	0	<b>46.00</b>	<b>46</b>	9					
r125.5	<b>36</b>	<b>36</b>	0							<b>36.0</b>	<b>36</b>	0	<b>36.00</b>	<b>36</b>	1					
r250.1	<b>8</b>	<b>8</b>	0							<b>8.0</b>	<b>8</b>	0	<b>8.00</b>	<b>8</b>	0					
r250.1c	<b>64</b>	<b>64</b>	0							<b>64.0</b>	<b>64</b>	2	<b>64.00</b>	<b>64</b>	53					
r250.5	<b>65</b>	<b>65</b>	7							65.8	<b>65</b>	16	65.00	<b>65</b>	50					
r1000.1	<b>20</b>	<b>20</b>	1							<b>20.0</b>	<b>20</b>	0	<b>20.00</b>	<b>20</b>	16					
r1000.1c	98	98	46							98.8	98	557	98.00	98	1979					
r1000.5	237	241	77							238.6	237	1345	<b>234.00</b>	235	2298					
school1	<b>14</b>									<b>14.0</b>	<b>14</b>	0	<b>14.00</b>	<b>14</b>	0					
school1_nsh	<b>14</b>									<b>14.0</b>	<b>14</b>	1	<b>14.00</b>	<b>14</b>	0					
latin_square_10	99									100.2	99	938	101.75	101	1444					
flat300_20_0	<b>20</b>	<b>20</b>	0				10/10	<b>20</b>	0	<b>20.0</b>	<b>20</b>	2	<b>20.00</b>	<b>20</b>	40					
flat300_26_0	<b>26</b>	<b>26</b>	1				10/10	<b>26</b>	0	<b>26.0</b>	<b>26</b>	1	<b>26.00</b>	<b>26</b>	42					
flat300_28_0	<b>28</b>	31	156	6/10	31	20	3/10	<b>28</b>	420	31.0	31	133	31.75	31	73					
flat1000_50_0	<b>50</b>	<b>50</b>	0				10/10	<b>50</b>	18	<b>50.0</b>	<b>50</b>	14	<b>50.00</b>	<b>50</b>	1024					
flat1000_60_0	<b>60</b>	<b>60</b>	0				10/10	<b>60</b>	90	<b>60.0</b>	<b>60</b>	59	<b>60.00</b>	<b>60</b>	1107					
flat1000_76_0	83	89	897	4/5	83	1471	5/10	87	18000	87.8	87	2499	94.5	91	1086					

- [10] F. Chow and J. Hennessy, “The priority-based coloring approach to register allocation,” *ACM Transactions on Programming Languages and Systems*, vol. 12, no. 4, pp. 501–536, 1990.
- [11] J. Culberson and F. Luo, “Exploring the k-colorable landscape with iterated greedy,” in *Cliques, Coloring, and Satisfiability: 2nd DIMACS Implementation Challenge, 1993*, ser. DIMACS Series in Discrete Mathematics and Theoretical Computer Science, D. Johnson and M. Trick, Eds. American Mathematical Society, 1996.
- [12] D. de Werra, “An introduction to timetabling,” *European Journal of Operational Research*, vol. 19, pp. 151–162, 1985.
- [13] I. Diaz and P. Zabala, “A branch and cut algorithm for graph coloring,” in *Proceedings of the Computational Symposium on Graph Coloring and its Generalization*, Ithaca, New York, 2002.
- [14] J. Dongarra, “Performance of various computers using standard linear equations software, (linpack benchmark report),” University of Tennessee, Computer Science Department, Technical Report CS-89-85, 2005.
- [15] R. Dorne and J. Hao, “Tabu search for graph coloring, t-coloring and set t-colorings,” in *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization*, S. Voss, S. Martello, I. Osman, and C. R. (eds.), Eds. Boston: Kluwer Academic Publishers, 1998.
- [16] C. Fleurent and J. Ferland, “Object-oriented implementation of heuristic search methods for graph coloring,” in *Cliques, Coloring, and Satisfiability: 2nd DIMACS Implementation Challenge, 1993*, ser. DIMACS Series in Discrete Mathematics and Theoretical Computer Science, D. Johnson and M. Trick, Eds. American Mathematical Society, 1996.
- [17] N. Funabiki and T. Higashino, “A minimal-state processing search algorithm for graph coloring problems,” *IEICE Trans.*

TABLE III  
AVERAGE GAP ON THE COMMON SUBSET OF INSTANCES.

	Instance set from [29]	Instance set from [18]	Instance set from [2]	Full instance set [17]
Impasse				
avg gap		1.0114		
HCA				
avg gap		1.0163		
PC-RPC				
avg gap			1.0086	
<i>MIPS_CLR</i>				
avg gap				1.0072
MMT Algorithm				
avg gap	1.0139	1.0476	1.0325	1.01341

- Fundamentals*, vol. E83-A, no. 7, pp. 1420–1430, 1990.
- [18] P. Galinier and J. Hao, “Hybrid evolutionary algorithms for graph coloring,” *Journal of Combinatorial Optimization*, vol. 3, no. 4, pp. 379–397, 1999.
- [19] A. Gamst, “Some lower bounds for a class of frequency assignment problems,” *IEEE Transactions on Vehicular Technology*, vol. 35, no. 1, pp. 8–14, 1986.
- [20] M. Garey and D. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, ser. DIMACS Series in Discrete Mathematics and Theoretical Computer Science. New York: Freedman, 1979.
- [21] A. Hertz and D. de Werra, “Using tabu search techniques for graph coloring,” *Computing*, vol. 39, pp. 345–351, 1987.
- [22] D. Johnson, C. Aragon, L. McGeoch, and C. Schevon, “Optimization by simulated annealing: an experimental evaluation; part 2, graph coloring and number partitioning,” *Operations Research*, vol. 39, no. 3, pp. 378–406, 1991.
- [23] D. Johnson and M. T. (eds.), *Cliques, Coloring, and Satisfiability: 2nd DIMACS Implementation Challange, 1993*, ser. DIMACS Series in Discrete Mathematics and Theoretical Computer Science. American Mathematical Society, 1996.
- [24] J. P. Kelly and J. Xu, “A set-partitioning-based heuristic for the vehicle routing problem,” *INFORMS Journal on Computing*, vol. 11, no. 2, pp. 161–172, 1999.
- [25] F. Leighton, “A graph coloring algorithm for large scheduling problems,” *Journal of Research of the National Bureau of Standards*, vol. 84, no. 6, pp. 489–503, 1979.
- [26] R. Marsten and F. Shepardson, “Exact solution of crew scheduling problems using the set partitioning model: recent successful applications,” *Networks*, vol. 112, pp. 167–177, 1981.
- [27] A. Mehrotra and M. Trick, “A column generation approach for graph coloring,” *INFORMS Journal on Computing*, vol. 8, pp. 344–354, 1996.
- [28] M. Monaci and P. Toth, “A set-covering based heuristic approach for bin-packing problems,” *INFORMS Journal on Computing*, (to appear).
- [29] C. Morgenstern, “Distributed coloration neighborhood search,” in *Cliques, Coloring, and Satisfiability: 2nd DIMACS Implementation Challange, 1993*, ser. DIMACS Series in Discrete Mathematics and Theoretical Computer Science, D. Johnson and M. Trick, Eds. American Mathematical Society, 1996.
- [30] T. Sager and S. Lin, “A pruning procedure for exact graph coloring,” *ORSA Journal on Computing*, vol. 3, no. 3, pp. 226–230, 1991.
- [31] E. Sewell, “An improved algorithm for exact graph coloring,” in *Cliques, Coloring, and Satisfiability: 2nd DIMACS Implementation Challange, 1993*, ser. DIMACS Series in Discrete Mathematics and Theoretical Computer Science, D. Johnson and M. Trick, Eds. American Mathematical Society, 1996.
- [32] D. Wedelin, “An algorithm for large scale 0-1 integer programming with application to airline crew scheduling,” *Annals of Operations Research*, vol. 57, pp. 283–301, 1995.
- [33] T. Woo, S. Su, and R. N. Wolfe, “Resource allocation in a dynamically partitionable bus network using a graph coloring algorithm,” *IEEE Trans. Commun.*, vol. 39, no. 12, pp. 1794–1801, 2002.

# Investigating inventory control tactics in two node capacitated supply chains

Georgia Skintzi<sup>\*</sup>, Gregory Prastacos<sup>†</sup> and George Ioannou<sup>‡</sup>

Athens University of Economics and Business

Department of Management Science and Technology  
Evelpidon 47A & Lefkados 33, 113 62 Athens, Greece

<sup>\*</sup>Email: [gskintzi@aueb.gr](mailto:gskintzi@aueb.gr)

<sup>†</sup>Email: [gpp@aueb.gr](mailto:gpp@aueb.gr)

<sup>‡</sup>Email: [ioannou@aueb.gr](mailto:ioannou@aueb.gr)

**Abstract**— In this paper we explore two alternative inventory control policies, upstream managed inventory and downstream managed inventory. Recent empirical results indicate that in certain industries has been a significant alteration in the perspective and the way inventory is managed. In contrast with “traditional” supply chains where the downstream node makes stock level decisions and controls inventory there has been a tendency in supply chain management to shift inventory decisions and control to the upstream. We consider a two-node supply chain that consists of one upstream node (manufacturer) and one downstream node (retailer). We examine two cases of inventory control, in the first case the downstream node is responsible for inventory decisions and bears the burden of inventory holding while in the second case the upstream node is responsible for deciding the stock level and hold inventory.

**Keywords** - Supply chain management, Inventory, Vendor managed inventory.

## I. INTRODUCTION

INVENTORY theory has been an influential research topic since the seminal work of [2] that triggered and stimulated a vast literature on dynamic inventory policy [1]. During the last two decades inventory theory and specifically inventory control found a fertile ground to develop in the context of supply chain management. [5] reviews supply chain inventory management models. The authors classified models into six categories according to the aspects they address: multiple retailers with stochastic demand, multiple retailers with deterministic demand, capacity allocation, information and production timing, internal markets, vendor managed inventories. During the last decade technological advancements coupled with the pressure of globalization and fierce competition have forced organizations towards new

business models revealing the benefits of cooperation and information sharing. In this context new inventory models have evolved and put into practice.

In the business world, there has been recorder an increased tendency to transfer inventory decisions upstream in the supply chain in contrast with “traditional” inventory practices where the downstream node makes stock level decisions and controls inventory. In [17] the authors list the new innovative approaches: Click and Mortar (CAM), Drop-Shipping (DS), Vendor Managed Inventory (VMI), Consignment VMI, and Vendor Hub. Empirical research indicates that upstream inventory control has been successfully employed by numerous organizations. CAM has been adopted by pharmaceutical chain CVS, the electronics retailer Circuit City [18] and Virgin’s V Shops [16]. VMI has been successfully implemented in the grocery industry [10], [6] and [22] resulting in most cases in improved performance, cost reduction, and increased service levels. Other VMI success stories include the apparel sector, the household appliances sector, and the electronics industry. In all cases has been recorded a considerable decrease in inventory and increase in service levels.

The large number of applications and empirical research of upstream inventory control indicate the significance of the approach and the need for theoretical background. Although the inventory control literature is vast, only recently has it been extended to incorporate the new trends in practical inventory control. Literature mainly examines

supplier/manufacturer-retailer supply chains [4], [6] and [19] focusing on retailers or manufacturers competition and supplier-manufacturer supply chains [14], [20], and [9], inventory costs and supplier contracts. [11] and [12] have investigated the implications of upstream inventory control on the Bullwhip effect suggesting that VMI may be employed to effectively respond to demand fluctuations and therefore, to the bullwhip effect. Several authors have also investigated the implications of vendor managed inventory on transportation operations [8], [7], [13], and [21].

The problem we address in this chapter emanates from the inventory control literature. We consider a two-node supply chain that consists of one upstream node (manufacturer) and one downstream node (retailer). The manufacturer produces a single product and provides it to the next supply chain node, the retailer. When demand is realized the retailer forwards the products to the market. We examine two cases of inventory control: in the first case the downstream node is responsible for inventory decisions and bears the burden of inventory holding while in the second case the upstream node is responsible for deciding the stock level and hold inventory. We explore the implications in production and inventory levels in both cases and propose specific directions for managing inventories. It should be highlighted that the optimization models were developed in game theoretical settings. The members of the supply chain do not passively react to given parameters but dynamically set their strategy in order to maximize their profits.

The remaining of the paper is organized as follows. In section II the problem is described in detail. In section III the optimization models and the results are presented. In section IV the results are discussed and analyzed. Finally section V provides some concluding remarks and future directions.

## II. PROBLEM DEFINITION: WHO SHOULD CONTROL INVENTORIES?

We consider a typical serial supply chain that consists of one manufacturer (upstream node), one retailer (downstream node) and one warehouse (buffer node). The manufacturer produces a single

product and provides it to the next supply chain node, the retailer. When demand is realized the retailer forwards the products to the market. We consider a time horizon of one period. At the beginning of the period the capacity is decided and the warehouse is build by the node that controls inventory and production is realized by the upstream node. At the end of the period demand is revealed, the downstream node decides the quantity that will be sold and releases the products to the market. During the period, products are stored in the warehouse. According to who has the control of the warehouse, the wholesale price of inventory is decided by the upstream node at the beginning or at the end of the period. We examine both cases of downstream and upstream control. Figure 1 represents the supply chain configurations in case the downstream or the upstream node controls inventory.

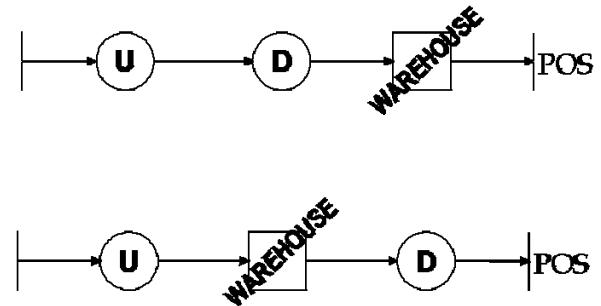


Fig. 1. Supply chain representation in case downstream node controls inventory and in case upstream node controls inventory

### *A. The case of downstream control*

In case the downstream node controls inventory the problem of the downstream node is to determine the capacity of the warehouse at the beginning of the period and the amount of inventory released in the market at the end of the period given the wholesale price of the product, the costs of building and operating the warehouse and the cost of holding the inventory in order to maximize his profits, given inventory constraints (he cannot release to the market quantity greater than the quantity produced and stored). The problem of the upstream node is to determine the wholesale price of products and the level of production at the beginning of the period, given the cost of production and that the

downstream node will act rationally in order to maximize his profits. We have assumed that the capacity of the warehouse and the production size are non-extensible after the completion of the warehouse and the production process respectively. Moreover, for simplicity we have assumed that unsold items have no value and lost sales have no penalty.

Figure 2 represents graphically the decision and action process of the downstream and the upstream node.

### B. The case of upstream control

In case the upstream node controls inventory the problem of the upstream node is to determine the capacity of the warehouse and to produce at the beginning of the period given the costs of building and operating the warehouse, the cost of holding the inventory and the cost of production. At the end of period the manufacturer determines the wholesale price of the product given that the downstream's node optimal policy in order to maximize his profits. The problem of the upstream node is to define the optimal quantity of inventory released in the market at the end of the period given the

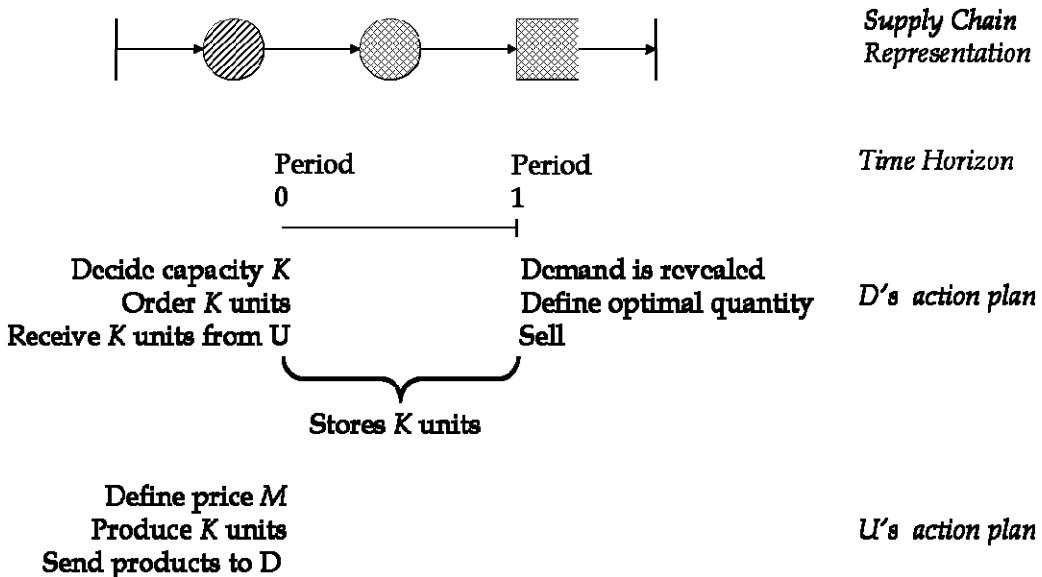


Fig. 2. Downstream's and upstream's action plans in case of downstream control

At the beginning of the period the downstream node (D) determines the size of the warehouse, builds the warehouse, places his orders for quantity equal to the capacity of the warehouse and receives the products. We assume that the production and the transportation of the products are realized instantly. "D" stores the products during the period and at the end when demand is revealed the optimal quantity to be released to the market is determined. The upstream node at the beginning of the period defines the wholesale price, produces, receives orders and provides products to the downstream node.

wholesale price of the product in order to maximize his profits, given inventory constraints (he cannot release to the market quantity greater than the quantity produced and stored). Similarly to the case of downstream control we assume that the capacity of the warehouse and the production size are non-extensible after the completion of the warehouse and the production process respectively. Moreover, for simplicity we have assumed that unsold items have no value and lost sales have no penalty.

Figure 3 below represents graphically the decision and action process of the upstream and the downstream node.

$$P_1 = \alpha - \beta q \quad (1)$$

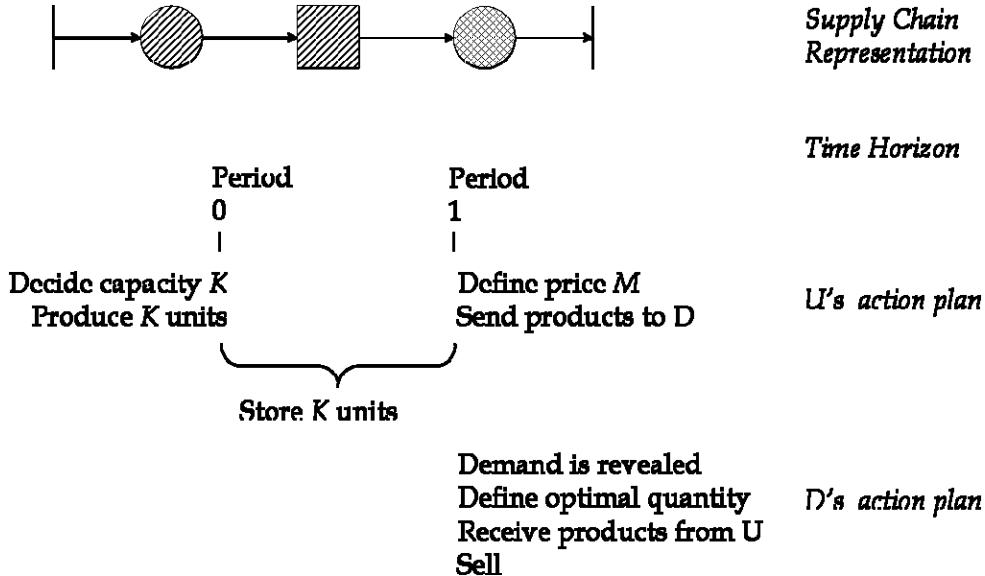


Fig. 3. Downstream' s and upstream' s action plans in case of upstream control

At the beginning of the period the upstream node decides the capacity of the warehouse, builds the warehouse and produces quantity equal to the capacity of the warehouse. We assume that the production is completed instantly. The manufacturer stores the products during the period and at the end defines the wholesale price and fills the downstream node' s orders. We assume that the transportation of the products is realized instantly. The downstream node at the end of the period when demand is revealed determines the optimal quantity of products to be released in the market, receives the products and forwards them to the market.

In both cases the objective of the upstream and the downstream node is to optimize their profits. The retailer optimally solves his problem given the wholesale price and subject to the inventory constraints. The manufacturer maximizes his profits subject to the optimal policy of the other player.

### III. FORMULATING THE MODEL

We assume that the demand in period 1 is linear.  $P_1$  is the price of the product and  $q$  is the quantity of inventory released in the market during period 1. The inverse demand curve for period 1 is:

To introduce uncertainty we will follow the analysis proposed by [3] based on [15] and assume that  $\alpha$  is stochastic and follows a Bernoulli process:

$$\alpha = \begin{cases} a_H, & (\text{high demand}) \text{ with probability } p \\ a_L, & (\text{low demand}) \text{ with probability } (1-p) \end{cases}$$

where  $a_H > a_L$ . In case demand is high then the price of product will be higher than in case of low demand, if the same quantity of inventory is released. We also assume that there exists a traded security that pays out  $a_H$  euros in the high state (in the case of high demand) and  $a_L$  euros in the low state (in the case of low demand). The price of such security at time zero (0) is  $A_0 = a_H e_H + a_L e_L$ , where  $e_H$  is the Arrow Debreu state price corresponding to a €1 payout in the case of high demand and €0 otherwise, and  $e_L$  is the Arrow Debreu state price corresponding to a €1 payout in the case of low demand and €0 otherwise. Moreover, a riskless bond exists that pays out €1 in period 1. The price of such bond at date 0 is  $B_0 = e_H + e_L < 1$ . The existence of such traded securities allows the pricing kernel to be uniquely determined. This assumption allows us to perform the valuations without regard to the specific risk preferences of the manufacturer and even if there is no consensus on the value of probabilities of high and low demand.

Under the risk neutral measure the expected growth of the traded security equals the risk free rate. In this model the latter is determined by the two risk-neutral probabilities  $p_H = e_H/B_0$ ,  $p_L = e_L/B_0$ . Under this measure the mean of the intercept term of the demand curve is:

$$\mu_\alpha = A_0/B_0 \quad (2)$$

and the variance is:

$$\sigma_\alpha^2 = [(e_H e_L)/B_0^2] (a_H - a_L)^2 \quad (3)$$

We also have:

$$\alpha_H = (A_0/B_0) + \sigma_\alpha \sqrt{\rho} \quad (4)$$

$$\alpha_L = (A_0/B_0) - \sigma_\alpha (1/\sqrt{\rho}) \quad (5)$$

where

$$\rho = e_L/e_H \quad (6)$$

#### A. The case of downstream control

At the beginning of the period the capacity of the warehouse and the production size should be determined while the demand is not known. At the end of period 1 demand is revealed and the quantity of inventory released should be determined subject to inventory constraints, that is the downstream node cannot release to the market quantity greater than inventory kept. In period 1 the net cash flow the downstream node receives is given below:

$$DR_1^D = q(a - \beta q) \quad (7)$$

In (7)  $DR_1^D$  is the income the downstream node receives from selling  $q$  units at price  $(a - \beta q)$ .

In period 0 the net present value of the project, considering that during period 0 the downstream node determines the capacity of the warehouse and during period 1 optimally manages the project, is given below:

$$DR_0^D(K) = -(W + iK) - MK - HMK + E[DR_1^D D_1] \quad (8)$$

where  $D_1$  is the state dependant stochastic discount rate (pricing kernel),  $W$  is the fixed cost and  $i$  is the variable cost of building and operating the warehouse.  $K$  is the capacity of the warehouse that equals the amount of inventory held, the inventory holding node will not build and operate a warehouse greater nor smaller than the inventory that will be held.  $M$  is the per unit cost of purchasing the product from the upstream node.  $HMK$  is the holding cost the downstream node bears. More specifically,  $H$  is the cost of holding the inventory expressed as a percentage of the purchasing cost  $M$ .

Therefore  $HM$  is the marginal holding cost. In period 1 the downstream node will choose  $q$  in order to maximize the net cash flow subject to inventory constraints (he cannot sell more than what he has produced).

Given  $e_H$  and  $e_L$ :

$$DR_0^D = -(W + iK) - MK - HMK + e_H [q_H^* (a_H - \beta q_H^*)] + e_L [q_L^* (a_L - \beta q_L^*)] \quad (9)$$

The objective of the manufacturer is to maximize the net present value of the project by identifying the size of inventory held and the capacity of the warehouse and solving the problem of period 1 optimally subject to inventory constraints.

During period 0 the upstream node determines his pricing policy by maximizing his revenues given that during period 1 and 0 the downstream node optimally manages his project. The net present value of the downstream's node project is given below:

$$DR_0^U = K(M - C) \quad (10)$$

where  $C$  is the per unit cost of production. Summarizing, Downstream's node problem in period 1:  $\max_{0 \leq q \leq K} DR_1^D(q)$  subject to inventory constraint. Downstream's node problem in period 0:  $\max_{K \geq 0} DR_0^D(K)$

Upstream's node problem in period 0:  $\max_{M \geq 0} DR_0^U(M)$

#### Proposition 1:

The optimal quantity of inventory released in the market by the downstream node during period 1 is:

$$q^* = \begin{cases} \frac{a}{2\beta}, & \text{if } \frac{a}{2\beta} < K \\ K, & \text{if } \frac{a}{2\beta} \geq K \end{cases}$$

Proof: See Appendix

#### Proposition 2:

The optimal quantity of inventory held during period 0 and the optimal capacity of the warehouse build is:

$$K^* = \begin{cases} \frac{A_0 - i - M(1+H)}{2\beta B_0}, & \text{if } M > \frac{e_H(a_H - a_L) - i}{1+H} \\ \frac{e_H a_H - i - M(1+H)}{2\beta e_H}, & \text{if } M < \frac{e_H(a_H - a_L) - i}{1+H} \end{cases}$$

Proof: See Appendix

*Proposition 3:*

The upstream's node optimal pricing policy is:

$$M^* = \begin{cases} \frac{A_0 - i + C(1+H)}{2(1+H)}, & \text{if } \sigma < \underline{\sigma} \\ \frac{e_H a_H - i + C(1+H)}{2\beta e_H}, & \text{if } \sigma > \underline{\sigma} \end{cases}$$

The downstream's node optimal response is:

$$K^* = \begin{cases} \frac{A_0 - i - C(1+H)}{2\beta B_0}, & \sigma < \underline{\sigma} \\ \frac{e_H a_H - i - C(1+H)}{2\beta e_H}, & \sigma > \underline{\sigma} \end{cases}$$

where

$$\underline{\sigma} = \frac{\sqrt{\rho}}{1+\rho+\sqrt{1+\rho}} \left( \frac{C+i}{e_H} + \frac{A_0}{B_0} \sqrt{1+\rho} \right)$$

Proof: See Appendix

Table I summarizes the optimal capacity of the warehouse, the optimal quantity of inventory and the optimal amount of inventory released in the market.

As a result of the above analysis the upstream node will order  $K^*$  units of inventory from the upstream node and in case of low uncertainty will release to the market all inventory while in case of high uncertainty will release all his inventory if demand proves to be high otherwise he will release quantity equals to  $q_L^*$ . It should be highlighted that, as expected, in case of high uncertainty the inventory held is greater than that in the case of low uncertainty. Moreover the upstream node will optimally respond to downstream's node ordering policy by charging  $M^*$  for his product. Notice that

the quantity the downstream node orders decreases as the wholesale price increases.

#### B. The case of upstream control

During period 0, the capacity of the warehouse and the production level (both controlled by the upstream node in this case) should be determined, while demand is not known. At the beginning of period 1 demand is revealed and the quantity of inventory released should be determined subject to inventory constraints, that is the downstream node cannot release to the market quantity greater than the inventory available by the upstream node. In period 1 the net cash flow the downstream node receives is given below:

$$UR_1^D = q(a - \beta q) - Mq \quad (11)$$

$UR_1^D$  is the income the downstream node receives from selling  $q$  units at price  $(a - \beta q)$ .

In period 0 the net present value of the upstream's node project, considering that during period 1 the downstream node determines the inventory released to the market, is given below:

$$UR_0^U(K, M) = -(W + iK) - CK - HCK \quad (12) \\ + E[R_1^D D_1]$$

where  $HCK$  is the holding cost the upstream node bears. More specifically,  $H$  is the cost of holding the inventory expressed as a percentage of the production cost  $C$ . Therefore,  $HC$  is the marginal holding cost. It should be noted that the per unit holding cost is less in case the upstream node controls inventory since inventory cost is expressed as a percentage of the value of inventory which increases as we move down in the supply chain. Hence, in case the downstream node controls inventory the marginal inventory cost is  $HM$  while in case the upstream node controls inventory the marginal inventory cost is  $HC < HM$ .

Given  $e_H$  and  $e_L$  the net present value of the upstream's node project is:

$$UR_0^D = -(W + iK) - MK - HMK \quad (13) \\ + e_H[q_H^*(a_H - \beta q_H^*)] + e_L[q_L^*(a_L - \beta q_L^*)]$$

The objective of the upstream node is to maximize the net present value of the project by identifying the size of inventory held and the price of the product sold to the downstream node given that the

downstream node optimally manages his project during period 1 subject to inventory constraints (he cannot sell more units than those produced by the upstream node). Summarizing, Downstream' s node problem in period 1:  $\underset{0 \leq q \leq K}{\text{Max}} UR_1^D(q)$  subject to inventory constraint. Upstream' s node problem in period 0:  $\underset{K \geq 0}{\text{Max}} UR_0^D(K, M)$

*Proposition 4:*

The optimal quantity of inventory released in the market by the downstream node during period 1 is:

$$q^* = \begin{cases} \frac{a - M}{2\beta}, & \frac{a - M}{2\beta} < K \\ K, & \frac{a - M}{2\beta} \geq K \end{cases}$$

Proof: See Appendix

*Proposition 5:*

The optimal quantity of inventory held during period 0 and the optimal capacity of the warehouse build is:

$$K^* = \begin{cases} \frac{a_L - M}{2\beta}, & M \leq \frac{a_L B_0}{B_0 + e_L} \\ \frac{a_H - M}{2\beta}, & M > \frac{a_L B_0}{B_0 + e_L} \end{cases}$$

Notice that as the price of product increases the optimum capacity of inventory held also increases.

Proof: See Appendix

The upstream' s node optimal pricing policy is:

$$M^* = \begin{cases} \frac{a_L B_0 + i + C(1+H)}{2B_0}, & \sigma \leq \bar{\sigma} \\ \frac{A_0 + i + C(1+H)}{2B_0}, & \sigma > \bar{\sigma} \end{cases}$$

Therefore, the optimal quantity of inventory held could be rewritten as:

$$K^* = \begin{cases} \frac{a_L B_0 - i - C(1+H)}{4\beta B_0}, & \sigma \leq \bar{\sigma} \\ \frac{2a_H B_0 - A_0 - i - C(1+H)}{4\beta B_0}, & \sigma > \bar{\sigma} \end{cases}$$

where

$$\bar{\sigma} = \sqrt{\rho} \left( \frac{A_0}{B_0} + \frac{[C(1+H) + i](B_0 + A_0)}{e_H B_0} \right)$$

Proof: See Appendix

Table II summarizes the optimal capacity of the warehouse, the optimal quantity of inventory and the optimal amount of inventory released in the market.

As a result of the above analysis the upstream node will produce  $K^*$  units and will charge  $M^*$ . It should be stressed that in case of low uncertainty the quantity produced by the upstream node is greater than in the case of high uncertainty. In addition the downstream node will release to the market all inventory in case of low uncertainty while in case of high uncertainty he will release all his inventory if demand proves to be high otherwise he will release quantity equals to  $q_L^*$ . Note that the quantity the downstream node orders and releases to the market decreases as the wholesale price increases.

#### IV. WHO SHOULD CONTROL INVENTORIES?

From the previous analysis certain direction may be derived as to who should control inventory and in what case.

(i) For the case of low demand uncertainty ( $\sigma < \underline{\sigma}$ ) the downstream node will be better off in case of upstream inventory control only if the initial investment in warehousing facilities is greater than the critical value  $\bar{W}'$  ( $W > \bar{W}'$ ). The upstream node will be better off in case he controls inventory if the initial investment in warehousing facilities is lower than the critical value  $\underline{W}'$  ( $W < \underline{W}'$ ). Since  $\underline{W}' < \bar{W}'$  there is no solution that will lead to a situation that both members of the supply chain are better off.

TABLE I  
RESULTS FOR THE CASE OF DOWNSTREAM INVENTORY CONTROL

Standard Deviation	Inventory/Capacity (K)	Inventory Released		Price of Inventory (M)
		High State	Low State	
$\sigma < \underline{\sigma}$	$\frac{A_0 - i - M(1+H)}{2\beta B_0}$	$K^*$	$K^*$	$\frac{A_0 - i + C(1+H)}{2(1+H)}$
$\sigma > \bar{\sigma}$	$\frac{e_H a_H - i - M(1+H)}{2\beta e_H}$	$K^*$	$\frac{a_L}{2\beta}$	$\frac{e_H a_H - i + C(1+H)}{2(1+H)}$

TABLE II  
RESULTS FOR THE CASE OF UPSTREAM INVENTORY CONTROL

Standard Deviation	Inventory/ Capacity (K)	Inventory Released		Price of Inventory (M)
		High State	Low State	
$\sigma_a < \bar{\sigma}$	$\frac{a_L - M}{2\beta}$	$K^*$	$K^*$	$\frac{a_L B_0 + i + C(1+H)}{2B_0}$
$\sigma_a > \bar{\sigma}$	$\frac{a_H - M}{2\beta}$	$K^*$	$\frac{a_L - M}{2\beta}$	$\frac{A_0 + i + C(1+H)}{2B_0}$

On the contrary both players will be worse off if  $\underline{W}' < W < \bar{W}'$ .

Where

$$\bar{W}' = \frac{A_0(2A_0 - a_L B_0) + 3a_L^2 B_0 + 2\Phi e_H(a_H - a_L)}{16\beta B_0}$$

$$\begin{aligned} \underline{W}' = & \frac{\Phi^2(B_0 - 1 - H) + \Phi[A_0(H - 1)]}{8\beta B_0(1 + H)} \\ & + \frac{a_L B_0(1 + H) + A_0 B_0[A_0 - a_L(1 + H)]}{8\beta B_0(1 + H)} \end{aligned}$$

and  $\Phi = [C(1+H)+i]$ .

(ii) For the case of medium demand uncertainty ( $\underline{\sigma} < \sigma < \bar{\sigma}$ ) the downstream node will be better off in case of upstream inventory control only if the initial investment in warehousing facilities is greater than the critical value  $\bar{W}''$  ( $W > \bar{W}''$ ). The

upstream node will be better off in case he controls inventory if the initial investment in warehousing facilities is lower than the critical value  $\underline{W}''$

( $W < \underline{W}''$ ) In this case  $\underline{W}'' > \bar{W}''$ . Both members of the supply chain will be better off in case of upstream inventory control if  $\bar{W}'' < W < \underline{W}''$ .

Where

$$\begin{aligned} \underline{W}'' = & \frac{(1+H)(a_L B_0 - \Phi)^2}{8\beta B_0(1+H)} \\ & - \frac{B_0[e_H a_H(A_0 + 2e_L a_L) + \Phi(\Phi - 2A_0)]}{8\beta B_0(1+H)} \end{aligned}$$

$$\bar{W}'' = \frac{\rho\Phi^2 + 2\Phi(a_H - a_L)(e_H - e_L)}{16\beta B_0}$$

$$\begin{aligned} \text{and } & - \frac{B_0[3a_L^2(B_0 + 2e_L) + e_H a_H(4a_L - a_H)]}{16\beta B_0} \end{aligned}$$

(iii) For the case of high demand uncertainty

$(\sigma > \bar{\sigma})$  the downstream node will be better off in case of upstream inventory control if the initial investment in warehousing facilities is greater than the critical value  $\bar{W}'''$  ( $W > \bar{W}'''$ ).

The upstream node will be better off in case the downstream node controls inventory if the initial investment in warehousing facilities is lower than the critical value  $\underline{W}'''$  ( $W < \underline{W}'''$ ). In this case it is not clear if  $\underline{W}''' < \bar{W}'''$  or  $\underline{W}''' > \bar{W}'''$ . Therefore, for the case  $\underline{W}''' < \bar{W}'''$  there is no situation where both members of the supply chain are better off, while for the case  $\underline{W}''' > \bar{W}'''$  both are better off when  $\bar{W}''' < W < \underline{W}'''$ .

$$\bar{W}''' = \frac{B_0(e_H a_H - \Phi)(A_0 - \Phi + e_L a_L)}{8\beta B_0(1+H)}$$

Where  $+ \frac{(1+H)(8\Phi\beta B_0 - A_0^2 - \Phi^2)}{8\beta B_0(1+H)}$  and

$$\begin{aligned} \bar{W}''' = & \frac{\rho\Phi^2 + B_0(2\Phi + 2a_L^2 B_0 - 2a_H + 3a_H^2 e_L)}{16\beta B_0} \\ & - \frac{4e_H e_L(a_H - a_L) + A_0(1 - 4e_L - 4A_0)}{16\beta B_0} \end{aligned}$$

Notice that the breaking points of standard deviation alter as the inventory control policy changes. In case the downstream node controls inventory

$$\underline{\sigma} = \frac{\sqrt{\rho}}{1+\rho+\sqrt{1+\rho}} \left( \frac{C+i}{e_H} + \frac{A_0}{B_0} \sqrt{1+\rho} \right) \text{ while in}$$

case the upstream node controls inventory

$$\bar{\sigma} = \sqrt{\rho} \left( \frac{A_0}{B_0} + \frac{[C(1+H)+i](B_0+A_0)}{e_H B_0} \right), \text{ while}$$

$\underline{\sigma} < \bar{\sigma}$ . Moreover in case  $\sigma < \underline{\sigma}$  and  $\underline{\sigma} < \sigma < \bar{\sigma}$  the quantity of inventory kept and the capacity of the warehouse is greater if the downstream node controls inventory while if  $\sigma > \bar{\sigma}$  then the quantity of inventory kept and the capacity of the warehouse is greater if the upstream node controls inventory.

## V. CONCLUSIONS

Inventory control has traditionally been the main leverage in the struggle to effectively respond to

demand uncertainty and fluctuations. Technological advancements and the realization that each company is part of one or more supply chains that dictate its performance and effectiveness have resulted to the adoption of cooperative strategies through information sharing and contracting. Empirical research indicates that has been a significant alteration in the perspective and the way inventory is managed, providing evidence that a new inventory control policy has emerged called vendor managed inventory or upstream inventory control. While most research focuses either on case studies and simulation models or on the impact of VMI on transportation operations, bullwhip effect and competition we explored the effects of upstream inventory control in a manufacturer-retailer capacitated supply chain under warehousing considerations. Downstream and upstream inventory control policies are analyzed and rules that identify when each policy should be employed are derived. In addition, the members of the supply chain do not passively determine their optimal policies but with respect to each others strategy in a game theoretical setting. The optimization models and the dynamic approach enlighten the dynamics of upstream inventory control policies, explore the implications in pricing, and in production and inventory levels and propose specific directions for managing inventory successfully.

## REFERENCES

- [1] Arrow, K.J., "The genesis of Optimal inventory policy". *Operations Research*, 2002, vol. 50, no. 1, pp. 1-2.
- [2] Arrow, K.J.; Harris, T. and Marschak, J., "Optimal inventory policy", *Econometrica*, 1951, vol. 19, pp. 250-272.
- [3] Burnetas, A., Ritchken, P., "Option Contracts in Supply Chains", *6th Annual Real Options Conference*. Paphos, Cyprus, July 2002 [Online]. Available: [http://www.realoptions.org/papers2002/burnetas\\_ritchken.pdf](http://www.realoptions.org/papers2002/burnetas_ritchken.pdf).
- [4] Cachon, G.P., "Stock wars: Inventory competition in a two-echelon supply chain with multiple retailers", *Operations Research*, 2001, vol. 49, no. 5, pp. 658-674.
- [5] Cachon, G.P., "Competitive supply chain inventory management" in *Quantitative Models for Supply Chain Management*. International

- Series in Operations Research and Management Science, 17. Eds. Tayur, S., Ganeshan, R. and Magazine. Kluwer Cambridge, 2002.
- [6] Cachon, G.P. and Fisher, M.L., "Campbell Soup's continuous replenishment program: Evaluation and enhanced inventory decision rules", *Production and Operations Management* 1997, 6, 266-276.
- [7] Cetinkaya, S. and Lee, C.Y. "Stock replenishment and shipment scheduling for vendor-managed inventory systems", *Management Science*, 2000, vol. 46, no. 2, pp. 217-232.
- [8] Cheung, K.L. and Lee, H.L. "The inventory benefit of shipment coordination and stock rebalancing a supply chain", *Management Science*, 2002, vol. 48, no. 2, pp. 300-306.
- [9] Choi, K.S.; Dai, J.G. and Song, J.S., "On measuring supplier performance under vendor-managed-inventory programs in capacitated supply chains", *Manufacturing & Service Operations Management*, 2004, vol. 6, no. 1, pp. 53-72.
- [10] Clark, T.H. and Hammond, J.H. "Reengineering channel reordering processes to improve total supply chain performance" *Production and Operations Management*, 1997, vol. 6, no. 3, pp. 248-265.
- [11] Disney, S.M. and Towill, D.R. "The effect of vendor managed inventory (VMI) dynamics on the Bullwhip Effect in supply chains", *International Journal of Production Economics*, 2003(a), vol. 85, pp. 199-215.
- [12] Disney, S.M. and Towill, D.R., "Vendor-managed inventory and bullwhip reduction in a two-level supply chain". *International Journal of Operations & Production Management*, 2003(b), vol. 23, no. 6, pp. 625-651.
- [13] Disney, S.M.; Potter, A.T.; and Gardner, B.M., "The impact of vendor managed inventory of transport operations", *Transport Research Part E*. 2003, vol. 39, pp. 363-380.
- [14] Dong, Y. and Xu, K., "A supply chain model of vendor managed inventory", *Transportation Research Part E*, 2002, vol. 38, pp. 75-95.
- [15] Duffie D., *Dynamic Asset Pricing Theory*. Princeton University Press, 1996.
- [16] Jardine, A. and Anderson, A.C., "Virgin to sell online brands in V Shop", *Marketing*. 2002, vol. 7, pp. 168.
- [17] Lee, C.C. and Chu, W.H., "Who should control inventory in a supply chain?" *European Journal of Operational Research*, 2005, vol. 164, pp. 158-172.
- [18] Lee, H.L. and Whang, S., "Value of postponement from variability reduction through uncertainty resolution and forecast improvement" in *Product Variety Management: Research Advances*. Ho, T. and Tang, C. (Eds.) Kluwer Publishers, Boston., 1998, pp. 66-84.
- [19] Mishra, B.K. and Raghunathan, S., "Retailer- vs. Vendor- managed inventory and brand competition", *Management Science*, 2004, vol. 50,no. 4, pp. 445-457.
- [20] Pohlen, T.L. and Goldsby, T.J., "VMI and SMI programs. How economic value added can help sell the change" *International Journal of Physical Distribution & Logistics Management*, 2003, vol. 33, no. 7, pp. 565-581.
- [21] Rusdiansyah, A. and Tsao, D., "An integrated model of the periodic delivery problems for vending-machine supply chains", *Journal of Food Engineering*. 2005, to be published.
- [22] Tyan, J. and Wee, H.M., "Vendor managed inventory: a survey of the Taiwanese grocery industry" *Journal of Purchasing and Supply Management*, 2005, To be published.

## APPENDIX

PROOF OF PROPOSITION 1

In period 1 the downstream node solves the problem:

$$\max_{q \geq 0} DR_1^D(q) = (a - \beta q)q \quad (1.1)$$

subject to the constraint  $q \leq K$

$$(1.2)$$

In case we did not have a capacity constraint the optimal solution would be:

$$\bar{q}_H^* = \frac{a_H}{2\beta}, \text{ in case of high demand} \quad (1.3)$$

$$\bar{q}_L^* = \frac{a_L}{2\beta}, \text{ in case of low demand} \quad (1.4)$$

Taking into consideration the capacity constraint the optimal solution is:

$$q^* = \begin{cases} \frac{a}{2\beta}, & \text{if } \frac{a}{2\beta} < K \\ K, & \text{if } \frac{a}{2\beta} \geq K \end{cases} \quad (1.5)$$

### PROOF OF PROPOSITION 2

In period 0 the downstream node solves the problem:

$$\begin{aligned} \underset{K \geq 0}{\text{Max}} \quad DR_0^D(K) = & -(W + iK) - MK - HMK + e_H[q_H^*(a_H - \beta q_H^*)] \\ & + e_L[q_L^*(a_L - \beta q_L^*)] \end{aligned} \quad (2.1)$$

subject to the constraints

$$q_H^* \leq K \quad (2.2)$$

$$q_L^* \leq K \quad (2.3)$$

We should consider three cases:

$$0 < K \leq \frac{a_L}{2\beta} \quad (2.4)$$

$$\frac{a_L}{2\beta} < K \leq \frac{a_H}{2\beta} \quad (2.5)$$

$$\frac{a_H}{2\beta} < K \quad (2.6)$$

Case (2.6) is not applicable since it suggests that the capacity of the warehouse the manufacturer will build will be larger than the largest possible optimal quantity produced and stored ( $\bar{q}_H^*$ ). Since case (2.6) is economically irrational, that leaves us with cases (2.4) and (2.5).

Consider case (2.4).

$$DR_0^D(K_1^*) = \underset{0 \leq K \leq \frac{a_L}{2\beta}}{\text{Max}} \left\{ -(W + iK) - MK - HMK + e_H[K(a_H - \beta K)] + e_L[K(a_L - \beta K)] \right\} \text{ The optimal}$$

solution is:

$$K_1^* = \begin{cases} \frac{A_0 - i - M(1+H)}{2\beta B_0}, & \frac{e_H(a_H - a_L) - i}{1+H} \leq M \leq \frac{A_0 - i}{1+H} \\ \frac{a_L}{2\beta}, & M < \frac{e_H(a_H - a_L) - i}{1+H} \end{cases}$$

Consider case (2.5).

$$DR_0^D(K_2^*) = \underset{\frac{a_L}{2\beta} \leq K \leq \frac{a_H}{2\beta}}{\text{Max}} \left\{ -(W + iK) - MK - HMK + e_H[K(a_H - \beta K)] + e_L[\frac{a_L}{2\beta}(a_L - \beta \frac{a_L}{2\beta})] \right\}$$

The optimal solution is:

$$K_2^* = \begin{cases} \frac{e_H a_H - i - M(1+H)}{2\beta e_H}, & 0 \leq M \leq \frac{e_H(a_H - a_L) - i}{1+H} \\ \frac{a_L}{2\beta}, & M \geq \frac{e_H(a_H - a_L) - i}{1+H} \end{cases} \quad (2.7)$$

The result then follows.

### PROOF OF PROPOSITION 3

Given the results of Proposition 2 the upstream's profit as a function of  $M$  is

$$DR_0^U(M) = \begin{cases} K_1^*(M-C), & \text{if } M \geq \frac{e_H(a_H - a_L) - i}{1+H} \\ K_2^*(M-C), & \text{if } M < \frac{e_H(a_H - a_L) - i}{1+H} \end{cases}$$

Therefore, the maximum value of  $DR_0^U(M)$  is given by

$$DR_0^U(M^*) = \max \{ DR_1^{U*}, DR_2^{U*} \}$$

where,

$$DR_1^{U*} = \max \{ K_1^*(M-C) \} \quad (3.1)$$

$$DR_2^{U*} = \max \{ K_2^*(M-C) \} \quad (3.2)$$

From solving (3.1) we have

$$M_1^* = \begin{cases} \frac{A_0 - i + C(1+H)}{2(1+H)}, & \text{if } C \geq C' \\ \frac{e_H(a_H - a_L) - i}{1+H}, & \text{if } C < C' \end{cases} \quad \text{where } C' = \frac{e_H(a_H - a_L) - a_L e_H (1+\rho) - i}{1+H}$$

From solving (3.2) we have

$$M_2^* = \begin{cases} \frac{e_H a_H - i + C(1+H)}{2(1+H)}, & \text{if } C \leq C'' \\ \frac{e_H(a_H - a_L) - i}{1+H}, & \text{if } C > C'' \end{cases} \quad \text{where } C'' = \frac{e_H(a_H - a_L) - a_L e_H - i}{1+H}$$

Since  $C' < C''$  the solution can be summarized as follows

$$M^* = \begin{cases} \frac{e_H a_H - i + C(1+H)}{2(1+H)}, & \text{if } C \leq C' \\ \frac{A_0 - i + C(1+H)}{2(1+H)}, & \text{if } C \geq C'' \\ \frac{e_H a_H - i + C(1+H)}{2(1+H)}, & \text{if } C' < C < C'' \& DR_2^* > DR_1^* \\ \frac{A_0 - i + C(1+H)}{2(1+H)}, & \text{if } C' < C < C'' \& DR_2^* < DR_1^* \end{cases} \quad (3.3)$$

The solution can be simplified, by substituting the values of  $DR_2^*$  and  $DR_1^*$  for the case where, the relationship  $DR_2^* - DR_1^* > 0$  can be written as the following quadratic inequality

$$C^2(1+H)^2 - 2(1+H)[e_H(a_H - a_L) - i]C + i^2 - 2e_H i(a_H - a_L) - e_H^2 e_L a_L - e_H^2 a_H^2 > 0$$

The roots of the inequality are

$$k_1 = e_H(a_H - a_L) - i - e_H a_L \sqrt{1+\rho} \text{ and } k_2 = e_H(a_H - a_L) - i + e_H a_L \sqrt{1+\rho}$$

and the inequality is valid for  $k_1 < K_0 < k_2$ . Based on this we can simplify (3.3) as follows

$$M^* = \begin{cases} \frac{A_0 - i + C(1+H)}{2(1+H)}, & \text{if } C > k_1 \\ \frac{e_H a_H - i + C(1+H)}{1+H}, & \text{if } C < k_1 \end{cases}$$

Further, the conditions  $C > k_1$  and  $C < k_1$  can be reexpressed as an equivalent condition involving the volatility of the demand curve by substituting  $a_H = \frac{A_0}{B_0} + \sigma \sqrt{\rho}$  and  $a_L = \frac{A_0}{B_0} - \frac{\sigma}{\sqrt{\rho}}$ . Therefore, we obtain

$$M^* = \begin{cases} \frac{A_0 - i + C(1+H)}{2(1+H)}, & \sigma < \frac{\sqrt{\rho}}{1+\rho+\sqrt{1+\rho}} \left( \frac{C+i}{e_H} + \frac{A_0}{B_0} \sqrt{1+\rho} \right) \\ \frac{e_H a_H - i + C(1+H)}{2\beta e_H}, & \sigma > \frac{\sqrt{\rho}}{1+\rho+\sqrt{1+\rho}} \left( \frac{C+i}{e_H} + \frac{A_0}{B_0} \sqrt{1+\rho} \right) \end{cases} \quad (3.4)$$

for simplicity  $\underline{\sigma} = \frac{\sqrt{\rho}}{1+\rho+\sqrt{1+\rho}} \left( \frac{C+i}{e_H} + \frac{A_0}{B_0} \sqrt{1+\rho} \right)$

$$\text{By substituting to (2.7) we have } K^* = \begin{cases} \frac{A_0 - i - C(1+H)}{2\beta B_0}, & \sigma < \underline{\sigma} \\ \frac{e_H a_H - i - C(1+H)}{2\beta e_H}, & \sigma > \underline{\sigma} \end{cases}$$

#### PROOF OF PROPOSITION 4

In period 1 the downstream node solves the problem:

$$\underset{q \geq 0}{\text{Max}} DR_1^D(q) = (a - \beta q)q - Mq \quad (4.1)$$

subject to the constraint

$$q \leq K \quad (4.2)$$

In case we did not have a capacity constraint the optimal solution would be:

$$\bar{q}_H^* = \frac{a_H - M}{2\beta}, \text{ in case of high demand} \quad (4.3)$$

$$\bar{q}_L^* = \frac{a_L - M}{2\beta}, \text{ in case of low demand} \quad (4.4)$$

Taking into consideration the capacity constraint the optimal solution is:

$$q^* = \begin{cases} \frac{a-M}{2\beta}, & \text{if } \frac{a-M}{2\beta} < K \\ K, & \text{if } \frac{a-M}{2\beta} \geq K \end{cases} \quad (4.5)$$

#### PROOF OF PROPOSITION 5

In period 0 the upstream node solves the problem:

$$\underset{K \geq 0}{\text{Max}} \ DR_0^D(K) = -(W + iK) - CK - HCK + e_H[q_H^*M] + e_L[q_L^*M] \quad (5.1)$$

subject to the constraints

$$q_H^* \leq K \quad (5.2)$$

$$q_L^* \leq K \quad (5.3)$$

We should consider three cases:

$$0 < K \leq \frac{a_L - M}{2\beta} \quad (5.4)$$

$$\frac{a_L - M}{2\beta} < K \leq \frac{a_H - M}{2\beta} \quad (5.5)$$

$$\frac{a_H - M}{2\beta} < K \quad (5.6)$$

Case (5.6) is not applicable since it suggests that the capacity of the warehouse the manufacturer will build will be larger than the largest possible optimal quantity produced and stored ( $\bar{q}_H^*$ ). Since case (5.6) is economically irrational, that leaves us with cases (5.4) and (5.5).

Consider case (5.4) where

$$DR_0^D(K_1^*, M_1^*) = \underset{0 \leq K \leq \frac{a_L - M}{2\beta}}{\text{Max}} \left\{ -(W + iK) - CK(1+H) + e_H KM + e_L KM \right\} \quad (5.7)$$

since  $DR_0^D(K, M)$  is increasing with respect to  $K$  and will be maximized in  $K_1^* = \frac{a_L - M}{2\beta}$ . Therefore, (5.7)

$$DR_0^D(K_1^*, M_1^*) =$$

$$\text{will be rewritten: } \underset{0 \leq K \leq \frac{a_L - M}{2\beta}}{\text{Max}} \left\{ -(W + i \frac{a_L - M}{2\beta}) - C \frac{a_L - M}{2\beta} (1+H) + e_H \frac{a_L - M}{2\beta} M + e_L \frac{a_L - M}{2\beta} M \right\} \quad (5.8)$$

From solving (5.8) we have  $M_1^* = \frac{i + C(1+H) + a_L B_0}{2B_0}$ .

Consider case (5.5) where

$$DR_0^D(K_2^*, M_2^*) = \underset{\frac{a_L - M}{2\beta} < K \leq \frac{a_H - M}{2\beta}}{\text{Max}} \left\{ -(W + iK) - CK(1+H) + e_H KM + e_L \frac{a_L - M}{2\beta} M \right\} \quad (5.9)$$

$DR_0^D(K, M)$  is increasing with respect to  $K$ . Therefore, (5.9) will be maximized in  $K_2^* = \frac{a_H - M}{2\beta}$  and may be

$$\begin{aligned} & DR_0^D(K_2^*, M_2^*) = \\ & \text{rewritten: } \text{Max} \left\{ -(W + i \frac{a_H - M}{2\beta}) - C \frac{a_H - M}{2\beta} (1 + H) + e_H \frac{a_H - M}{2\beta} M + e_L \frac{a_L - M}{2\beta} M \right\} \quad (5.10) \\ & \text{From solving (5.10) we have } M_2^* = \frac{i + C(1 + H) + A_0}{2B_0} \quad \text{for } \frac{a_L B_0}{B_0 + e_L} < M. \quad \text{Therefore,} \\ & K^* = \begin{cases} \frac{a_L - M}{2\beta}, & M < \frac{a_L B_0}{B_0 + e_L} \\ \frac{a_H - M}{2\beta}, & M > \frac{a_L B_0}{B_0 + e_L} \end{cases} \quad \text{by substituting } M \text{ we have } K^* = \begin{cases} \frac{a_L B_0 - i - C(1 + H)}{4\beta B_0}, & \sigma \leq \bar{\sigma} \\ \frac{2a_H B_0 - A_0 - i - C(1 + H)}{4\beta B_0}, & \sigma > \bar{\sigma} \end{cases} \quad \text{and} \\ & M^* = \begin{cases} \frac{a_L B_0 + i + C(1 + H)}{2B_0}, & \sigma \leq \bar{\sigma} \\ \frac{A_0 + i + C(1 + H)}{2B_0}, & \sigma > \bar{\sigma} \end{cases} \quad \text{where } \bar{\sigma} = \sqrt{\rho} \left( \frac{A_0}{B_0} + \frac{[C(1 + H) + i](B_0 + A_0)}{e_H B_0} \right). \end{aligned}$$



# Ejection Chain Algorithms for the Traveling Salesman Problem

D. Gamboa\*, C. Osterman†, C. Rego† and F. Glover‡

\*Escola Superior de Tecnologia e Gestão de Felgueiras, Instituto Politécnico do Porto, Apt. 205, 4610-156, Felgueiras, Portugal

Email: [dgamboa@estgf.ipp.pt](mailto:dgamboa@estgf.ipp.pt)

†School of Business Administration, University of Mississippi, University, MS 38677, USA

Email: [{costerman,crego}@bus.olemiss.edu](mailto:{costerman,crego}@bus.olemiss.edu)

‡Leeds School of Business, University of Colorado, Boulder, CO 80309, USA

Email: [fred.glover@colorado.edu](mailto:fred.glover@colorado.edu)

**Abstract**— Ejection chain methods lead the state-of-the-art in local search heuristics for the traveling salesman problem (TSP). The most effective local search approaches primarily originate from the Stem-and-Cycle (S&C) ejection chain method and the classical Lin-Kernighan (LK) procedure, which can be viewed as a special case of an ejection chain method. This paper describes major components of the most effective ejection chain algorithms that are critical for success in solving large scale TSPs. A performance assessment of foremost algorithms is reported based upon an experimental analysis carried out on a standard set of symmetric and asymmetric TSP benchmark problems.

**Keywords**—traveling salesman problem, ejection chains, local search.

## I. INTRODUCTION

The Traveling Salesman Problem (TSP) consists in sequentially visiting a set of cities only once and finally returning to the initial city. The goal is to find the tour of minimum total distance. In spite of the simplicity of the problem statement, the TSP is exceedingly challenging and is the most studied problem in combinatorial optimization, having inspired well over a thousand publications.

In graph theory, the problem can be defined on a graph  $G = (V, A)$  with  $n$  vertices (nodes)  $V = \{v_1, \dots, v_n\}$  and a set of arcs  $A = \{(v_i, v_j) | v_i, v_j \in V, i \neq j\}$  with a non-negative cost (or distance) matrix  $C = (c_{ij})$  associated with  $A$ . The problem is considered to be symmetric (STSP) if we have  $c_{ij} = c_{ji}$  for all  $(v_i, v_j) \in A$ , and asymmetric (ATSP) otherwise. Elements of  $A$  are often called edges (rather than arcs) in the asymmetric case. The STSP, where distances satisfy the triangle inequality  $(c_{ij} + c_{jk} \geq c_{ik})$ , is the most studied special case

of the problem. The STSP (ATSP) consists in determining the minimum cost Hamiltonian cycle (circuit) on the problem graph, which is often simply called a *tour*.

The TSP is a classic NP-hard combinatorial problem, and therefore there is no polynomial-time algorithm able to solve all possible instances of the problem. Therefore, heuristic algorithms are used to provide that are high quality but not necessarily optimal. The importance of finding effective and efficient heuristics to solve large-scale TSP problems prompted the “8<sup>th</sup> DIMACS Implementation Challenge”, organized by Johnson, McGeogh, Glover, and Rego [18] and solely dedicated to TSP algorithms.

In this paper we focus on heuristics based on ejection chain methods because they have proven to be highly effective in solving TSP problems.

Since efficiency is also fundamental for high performance algorithms we also emphasize the importance of choosing an appropriate data structure to represent the tour and describe the state-of-the-art data structures used in the implementation of TSP algorithms.

Although there are several individual publications on ejection chain approaches to TSP, with this paper we intend to provide a new survey that summarizes and compares the best of those approaches that fit into the local search category. Other more general survey publications concerning heuristics for TSP, such as Johnson and McGeoch book chapters [15, 17], are not up to date and therefore we also bring in other algorithms to the analysis. Furthermore, we introduce completely new algorithms such as our two implementations for the ATSP. In addition, we also summarize the latest developments in efficiency obtained by the use recently introduced

data structures to represent the tour.

In the following sections we briefly describe the most prominent ejection chain algorithms for the TSP, discuss their salient performance characteristics, and finally present some conclusions.

## II. SYMMETRIC TSP

### A. Ejection chain based algorithms

Subpath ejection chain methods start from an initial tour and iteratively attempt to improve the current solution, generating moves coordinated by a reference structure. The generation of moves throughout the ejection chain process is based on a set of legitimacy restrictions that determine the set of edges allowed to be used in subsequent steps of constructing the ejection chain. Ejection chains are variable depth methods that generate a sequence of interrelated simple moves to create a compound move.

In the graph theory context, a subpath ejection chain of  $L$  levels on graph  $G$  consists of a sequence of simple operations, called ejection moves,  $\langle e_1, \dots, e_m, \dots, e_L \rangle$ , that sequentially transform a subgraph  $G_m$  of  $G$  into another subgraph  $G_{m+1}$  by disconnecting a subpath and reconnecting it with different components. At each level of the chain the subgraph may not represent a feasible solution (usually the reference structure does not correspond to a solution), but it is always possible to obtain a solution to the problem by applying an extra operation called a trial move. Therefore, a neighborhood search ejection chain procedure consists in generating a sequence of moves  $\langle e_1, t_1, \dots, e_m, t_m, \dots, e_L, t_L \rangle$ , where  $\langle e_m, t_m \rangle$  represents the paired ejection and trial moves of level  $m$  of the chain. The new solution is obtained by carrying out the compound move  $\langle e_1, e_2, \dots, e_m, t_m \rangle$ , where the subscript  $m$  identifies the chain level that produced the best trial solution. For an extensive description of ejection chain methods we refer the reader to [24].

In this section we summarize the main components of the most effective local search ejection chain algorithms and analyze their performance. These algorithms are chiefly based on the Stem-and-Cycle (S&C) procedure [11] and the Lin-Kernighan (LK) heuristic [20].

The S&C procedure is a specialized approach that generates dynamic alternating paths. The classical Lin-Kernighan approach, by contrast, generates static alternating paths. Funke, Grünert and Irnich [7] give a theoretical analysis of the differences between the types of paths generated by S&C and LK procedures.

### Johnson and McGeoch Lin-Kernighan (LK-JM)

The Lin-Kernighan neighborhood search is designed as a method to generate  $k$ -opt moves (which consist in deleting  $k$  edges and inserting  $k$  new edges) in a structured manner that provides access to a relevant subset of these moves by an efficient expenditure of computational effort. The approach is based on the fact that any  $k$ -opt move can be constructed as a sequence of 2-opt moves [4], and a restricted subset of those move sequences can be produced in a systematic and economic fashion.

The method starts by generating a low order  $k$ -opt move (with  $k \leq 4$ ) and then creates a Hamiltonian path by deleting an edge adjacent to the last one added. This completes the first level of the LK process. In succeeding levels each move consists of linking a new edge to the unique degree 1 edge that was not adjacent to the last edge added, followed by deleting the sole edge whose removal will generate another Hamiltonian path.

Additional sophistication of the basic method is provided by a backtracking process that allows restarting with an alternative vertex for insertion or deletion of an edge at level  $i$  and proceeding iteratively until reaching level  $L$ .

The Lin-Kernighan algorithm implementation analyzed in this paper is from Johnson and McGeoch [16], featured among the lead papers of the “8<sup>th</sup> DIMACS Implementation Challenge” [18]. The results reported for this implementation use Greedy initial solutions, 20 quadrant-neighbor candidate lists, the “don’t look bits” strategy, and the 2-level tree data structure [6] to represent the tour.

We will indicate the primary algorithms that incorporate one or more of these strategies in their design, including the best algorithms as determined by the 8<sup>th</sup> DIMACS Implementation Challenge. Algorithms that incorporate more innovative structures and that achieve the highest levels of performance are described in greater

detail.

### **Neto's Lin-Kernighan (LK-N)**

This implementation is described in [21]. Its main differences from LK-JM are the incorporation of special cluster compensation routines, the use of a candidate set combining 20 quadrant-neighbors and 20 nearest neighbors, and a bound of 50 moves for the LK searches. It also takes advantage of the “don't look bits” technique and the 2-level tree data structure.

### **Applegate, Bixby, Chvatal, and Cook Lin-Kernighan (LK-ABCC)**

This implementation is part of the Concorde library [1] and is based on [2]. It uses *Q-Boruvka* starting tours, 12 quadrant-neighbors candidate lists, the “don't look bits” technique, and the 2-level tree data structure. LK-ABCC bounds the LK searches by 50 moves, and the backtracking technique is slightly deeper than that of the LK-JM implementation.

### **Applegate, Cook and Rohe Lin-Kernighan (LK-ACR)**

In this case, the implementation is very similar to the preceding LK-ABBC approach, but the backtracking strategy is even deeper and broader. The depth of the LK searches, by contrast, is half that of the LK-ABBC approach (25 moves). This implementation is based on the design reported in [1, 3].

### **Helsgaun's Lin-Kernighan Variant (LK-H)**

This implementation, described in [14], modifies several aspects of the original Lin-Kernighan heuristic. The most notable difference is found in the search strategy. The algorithm uses larger (and more complex) search steps than the original one. Also, sensitivity analysis is used to direct and restrict the search. The algorithm does not employ backtracking, but it uses the “don't look bits” technique and the 2-level tree data structure.

LK-H is based on 5-opt moves restricted by carefully chosen candidate sets. Helsgaun's method for creating candidate sets may be the most valuable contribution of the algorithm. The rule in the original algorithm restricts the inclusion of links in the tour to the five nearest neighbors of a given city. LK-JM includes at least 20 nearest quadrant neighbors. Helsgaun points

out that edges selected simply on the basis of length may not have the highest probability of appearing in an optimal solution. Another problem with the original type of candidate set is that the candidate subgraph need not be connected even when a large fraction of all edges is included. This is the case for geometrical problems in which the point sets exhibit clusters.

Helsgaun therefore develops the concept of an  $\alpha$ -nearness measure that is based on sensitivity analysis of minimum spanning 1-trees. This measure undertakes to better reflect the probability that an edge will appear in an optimal solution. It also handles the connectivity problem, since a minimum spanning tree is (by definition) always connected.

The key idea, in brief, is to assign a value to each edge based on the length of a minimum 1-tree containing it. A candidate set of edges can then be chosen for each city by selecting edges with the lowest values. The effectiveness of  $\alpha$ -nearness in selecting promising edges can be further improved by transforming the graph. For this, a subgradient optimization method is utilized that strives toward obtaining graphs in which minimum 1-trees are close to being tours.

By using the  $\alpha$ -measure, the cardinality of the candidate set may generally be small without reducing the algorithm's ability to find short tours. In fact, Helsgaun claims that for his initial set of test problems, the algorithm was able to find optimal tours using as candidate edges the 5  $\alpha$ -nearest edges incident to each node.

### **Nguyen, Yoshihara, Yamamori and Yasunaga Lin-Kernighan Variant (LK-NYYY)**

A short description of this implementation can be found in [18]. This variant starts with a 5-opt move but uses 3-opt moves in the LK searches as opposite to LK-H that uses 5-opt as a basic move. It also uses “don't look bits”, Greedy starting solutions, and 12 quadrant-neighbor lists, but it uses a data structure with properties similar to segment trees [6]. The results reported from this algorithm were submitted to the DIMACS Challenge after the summary chapter [18] was finished. An extremely significant difference from the Helsgaun variant is that LK-NYYY is able to run instances up to 1,000,000 nodes whereas LK-H only manages instances up to 18,900 nodes and consumes a significant amount of computational time as is evident in Table IV.

### Rego, Glover and Gamboa Stem-and-Cycle (SC-RGG)

The SC-RGG procedure differs from the LK procedure in several key ways. Most notably the LK approach uses a Hamiltonian path as the reference structure to generate moves throughout the neighborhood construction. This structure is very close to being a valid TSP solution (it only requires adding an edge to link the two degree 1 nodes to obtain a tour). As a result, the structure implicitly limits the different types of moves it can generate. More general ejection chain methods allow a diversified set of reference structures which are able to generate moves that the classical TSP neighborhood approaches cannot.

The S&C method is based on the stem-and-cycle reference structure [11]. The implementation reported here was designed by Rego [23] and subsequently enhanced by Gamboa, Rego and Glover [8, 9]. The S&C reference structure is a spanning sub-graph of  $G$  consisting of a path called a stem  $ST = (v_t, \dots, v_r)$  connected to a cycle  $CY = (v_r, v_{s_1}, \dots, v_{s_2}, v_r)$ . A diagram of a Stem-and-Cycle structure is shown in Figure 1. The vertex  $v_r$  in common to the stem and the cycle is called the *root*, and the two vertices of the cycle adjacent to  $v_r$  are called *subroots*. Vertex  $v_t$  is called the *tip* of the stem.

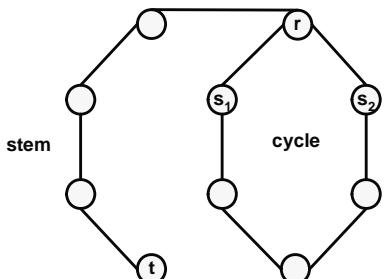


Fig. 1. The S&C reference structure

The method starts by creating the initial S&C reference structure from a TSP tour, by linking two nodes of the tour and removing one of the edges adjacent to one of those nodes. In each step of the ejection chain process, a subpath is ejected in the form of a stem. Each ejection move links the tip node to any other node on the graph, except for the one adjacent to the tip. Two different ejection moves are possible depending where in the graph the node to be linked to  $v_t$  is

placed (in the stem or in the cycle). Trial solutions are obtained by inserting an edge  $(v_t, v_s)$ , where  $v_s$  is one of the subroots, and deleting edge  $(v_r, v_s)$ .

The results reported in this paper improve upon those for the S&C method reported in the DIMACS challenge due to changes outlined in [8]. Here we present results using *Greedy* initial solutions, 12 quadrant-neighbor candidate lists concatenated with a list generated by the construction of Delaunay triangulations, and the 2-level tree data structure.

#### B. Comparative analysis of performance

In this subsection we evaluate the performance of the heuristic algorithms referenced in the previous subsection using the results submitted to the “8th DIMACS Implementation Challenge” [18] and the updated results for SC-RGG for a comparative analysis. We restrict attention to the evaluation of the results reported for the algorithms relevant to this paper’s main focus. For a complete list of algorithm results and other information related to the generation of the testbed instances, the scale factors to compare running times for different computer systems, and other characteristics of the challenge, we refer the reader to the Challenge web site [18].

The complete Challenge testbed consists of 3 sets of instances: uniformly distributed problems (sizes between 1,000 and 10,000,000 nodes), clustered problems (sizes between 1,000 and 316,228 nodes), and those from the TSP Library [25] with at least 1,000 nodes. In the current study we limited the number of problems to instances up to 1,000,000 nodes.

A benchmark code was provided for Challenge participants to run in the same machines as the competing algorithms were run, in order to obtain a more accurate comparison of running times. Since we carried out the tests for the updated version of S&C on the same machine we used to submit the first results to the DIMACS Challenge, we use the same scale factor to normalize the new implementation running times.

Tables I-IV summarize the results of the aforementioned algorithms. The values presented are averages of solution quality and computational times (in seconds), where instances are grouped by size. This grouping is similar to the one used by Johnson and McGeoch [17] to design their

results tables in the book chapter that summarizes the Challenge's submissions. It is important to stress, however, that a number of algorithms and results were submitted or updated after the chapter was published. In the solution quality tables, in addition to reporting average percentage excess over the optimal solution or the Held-and-Karp lower bound (%), we present the number of best solutions (NBS) found by each algorithm, meaning that for the indicated number of instances the associated algorithm obtained the solution of highest quality. The values in bold indicate the best averages.

We separate the basic LK algorithmic variants and the S&C approach from the other two LK variants since the latter are considerably more sophisticated and hence the comparison with the first five would be unfruitful. In other words, we separate the algorithms using 2-opt as a basic move from the ones using 5-opt or 3-opt. Basic LK variants and S&C determine moves by deleting one edge and inserting another one, completing the 2-exchange with the trial move. Helsgaun and NYYY variants search for valid 5-exchange moves and 3-exchange moves. To make this search possible in terms of computational times they use special and very sophisticated candidate lists as previously explained. These

sophistications and move evaluations can be introduced in a S&C variant and only in that case it would be fruitful to compare S&C with Helsgaun and NYYY. We are considering this possibility for further experiments with the S&C method for TSP.

From Tables I and II it is clear that the S&C approach is better than all other implementations for generating high quality solutions. It has, however, longer running times which are due to the lack of sophisticated techniques like the "don't look bits" approach that reduces the size of the neighborhood to be considered at each step of the algorithm without sacrificing tour quality. For the journal version of the paper we intend to report results for a version of the S&C algorithm with this technique incorporated in order to rigorously assess its influence in the running times. The tables also suggest that LK-JM has some advantages over the clustered instances.

From Tables III and IV we can assess that the LK-H achieves higher solution quality but with very heavy computational times. This is a serious drawback because the method becomes extremely difficult to use for solving the bigger instances. LK-NYYY obtains reasonably good results in this group of algorithms and is able to report solutions to all instances.

TABLE I  
SIMPLE LK AND S&C – SOLUTION QUALITY

Algorithm	Problem Size/Number of Instances - Uniformly Distributed Problems														Total	
	1000/10		3162/5		10000/3		31623/2		100000/2		316228/1		1000000/1			
	%	NBS	%	NBS	%	NBS	%	NBS	%	NBS	%	NBS	%	NBS	Average	NBS
LK-JM	1,18	1	1,27	1	2,02	--	2,02	--	1,97	--	1,96	--	1,96	--	1,77	2
LK-N	1,17		1,26	--	1,99	--	1,88	--	1,95	--	1,97	--	1,92	--	1,73	0
LK-ABCC	1,47	2	1,71	--	2,60	--	2,48	--	2,54	--	2,67	--	2,68	--	2,31	2
LK-ACR	1,61		2,18	--	2,72	--	2,72	--	2,74	--	2,75	--	2,77	--	2,50	0
SC-RGG	<b>0,79</b>	7	<b>0,95</b>	4	<b>1,68</b>	3	<b>1,61</b>	2	<b>1,65</b>	2	<b>1,86</b>	1	<b>1,91</b>	1	<b>1,49</b>	20
Problem Size/Number of Instances - Clustered Problems																
Algorithm	1000/10		3162/5		10000/3		31623/2		100000/2		316228/1				Total	
	%	NBS	%	NBS	%	NBS	%	NBS	%	NBS	%	NBS	%	NBS	Average	NBS
	<b>1,21</b>	6	<b>2,32</b>	4	3,41	2	3,72	--	<b>3,63</b>	1	<b>3,67</b>	1			<b>2,99</b>	14
LK-N	1,97	1	3,55	--	4,76	--	4,42	--	4,78	--	--	--			3,90	1
LK-ABCC	3,22	--	5,58	--	5,70	--	6,38	--	5,31	--	5,45	--			5,27	0
LK-ACR	3,34	--	5,48	--	5,92	--	6,28	--	5,55	--	5,54	--			5,35	0
SC-RGG	1,35	3	2,57	1	<b>3,24</b>	1	<b>3,16</b>	2	3,69	1	3,99	--			3,00	8

TABLE I (Cont.)

Algorithm	Problem Size/Number of Instances - TSPLIB Problems										Total Average	NBS	
	1000/4		3162/3		10000/2		31623/1		100000/1				
%	NBS	%	NBS	%	NBS	%	NBS	%	NBS				
LK-JM	1,40	--	1,28	--	1,38	--	1,23	--	1,21	--		1,30	0
LK-N	1,43	--	1,44	--	1,34	--	1,49	--	--	--		1,43	0
LK-ABCC	2,56	--	2,41	--	1,86	--	1,65	--	1,21	--		1,94	0
LK-ACR	3,49	--	2,59	--	3,17	--	2,40	--	2,00	--		2,73	0
SC-RGG	<b>0,52</b>	4	<b>0,60</b>	3	<b>0,91</b>	2	<b>1,02</b>	1	<b>1,17</b>	1		<b>0,84</b>	11

TABLE II  
SIMPLE LK AND S&C – COMPUTATIONAL TIME

Algorithm	Problem Size/Number of Instances - Uniformly Distributed Problems						
	1000/10	3162/5	10000/3	31623/2	100000/2	316228/1	1000000/1
CPU	CPU	CPU	CPU	CPU	CPU	CPU	CPU
LK-JM	0,20	0,69	2,32	7,16	22,75	60,62	322,47
LK-N	0,19	0,87	3,35	14,40	89,58	574,42	3577,74
LK-ABCC	0,09	0,34	1,49	5,95	21,43	60,79	307,17
LK-ACR	0,07	0,29	0,93	2,95	16,40	76,32	318,10
SC-RGG	4,04	19,82	100,92	733,93	5804,09	33239,39	255971,44

Algorithm	Problem Size/Number of Instances - Clustered Problems					
	1000/10	3162/5	10000/3	31623/2	100000/2	316228/1
CPU	CPU	CPU	CPU	CPU	CPU	CPU
LK-JM	1,66	4,97	15,37	59,26	173,11	495,47
LK-N	4,35	15,04	51,17	138,59	558,07	--
LK-ABCC	0,20	0,72	2,55	11,04	37,91	107,67
LK-ACR	0,11	0,45	1,40	4,49	24,97	114,19
SC-RGG	4,17	18,41	135,12	956,39	5416,85	60199,97

Algorithm	Problem Size/Number of Instances - TSPLIB Problems				
	1000/10	3162/5	10000/3	31623/2	100000/2
CPU	CPU	CPU	CPU	CPU	CPU
LK-JM	0,34	0,64	4,30	12,95	24,27
LK-N	0,41	1,08	10,26	47,09	--
LK-ABCC	0,10	0,29	1,22	3,48	8,84
LK-ACR	0,08	0,23	0,74	1,74	5,42
SC-RGG	7,59	26,47	134,99	665,85	3253,16

TABLE III  
HELSGAUN & NYYY – SOLUTION QUALITY

Algorithm	Problem Size/Number of Instances – Uniformly Distributed Problems										Total Average	NBS				
	1000/10	3162/5	10000/3	31623/2	100000/2	316228/1	1000000/1									
%	NBS	%	NBS	%	NBS	%	NBS	%	NBS	%	NBS					
LK-H	0,16	10	0,19	5	0,83	3	0,83	2	--	--	--	--	0,50	20		
LK-NYYY	0,73	--	0,74	--	1,57	--	1,48	--	1,48	2	1,53	1	1,49	1	1,29	4

Algorithm	Problem Size/Number of Instances - Clustered Problems							Average	NBS					
	1000/10	3162/5	10000/3	31623/2	100000/2	316228/1								
%	NBS	%	NBS	%	NBS	%	NBS							
LK-H	0,71	8	1,38	4	3,32	1	3,58	1	--	--	--	--	2,25	14
LK-NYYY	1,22	2	2,18	1	3,08	2	3,45	1	3,51	2	3,49	1	2,82	9

TABLE III (Cont)

Problem Size/Number of Instances - TSPLIB Problems												
Algorithm	1000/4		3162/3		10000/2		31623/1		100000/1		Average	NBS
	%	NBS	%	NBS	%	NBS	%	NBS	%	NBS		
LK-H	0,24	4	0,15	3	0,24	2	0,46	1	0,85	1	0,39	11
LK-NYYY	1,15	--	0,86	--	0,72	--	0,99	--	1,03	--	0,95	0

TABLE IV  
HELSGAUN & NYYY – COMPUTATIONAL TIME

Problem Size/Number of Instances - Uniformly Distributed Problems							
Algorithm	1000/10	3162/5	10000/3	31623/2	100000/2	316228/1	1000000/1
	CPU	CPU	CPU	CPU	CPU	CPU	CPU
LK-H	5,64	71,49	861,71	7819,27			
LK-NYYY	0,16	0,57	1,76	4,97	20,86	84,73	507,62
Problem Size/Number of Instances - Clustered Problems							
Algorithm	1000/10	3162/5	10000/3	31623/2	100000/2	316228/1	
	CPU	CPU	CPU	CPU	CPU	CPU	
LK-H	6,93	70,28	768,31	12812,46			
LK-NYYY	0,50	1,36	3,96	9,68	38,81	147,20	
Problem Size/Number of Instances - TSPLIB Problems							
Algorithm	1000/10	3162/5	10000/3	31623/2	100000/2		
	CPU	CPU	CPU	CPU	CPU		
LK-H	7,82	73,32	1063,13	7982,09	48173,84		
LK-NYYY	0,26	0,66	1,96	5,09	13,06		

### C. Advances on data structures for large STSPs

The problem of data representation is fundamental to the efficiency of search algorithms for the TSP and particularly important for large STSP instances. The nature of these algorithms necessitates the performance of certain basic tour operations involving subpath reversal and traversal. The computational effort that must be devoted to these operations becomes increasingly pronounced with larger problem instances. For example, if the tour is represented as an array (or doubly linked list) of nodes, a subpath reversal takes time  $O(n)$ , where  $n$  is the problem size.

We have recently developed a new data structure—the *k-level satellite tree*—for the purpose of minimizing the contribution of tour management toward the overall runtime cost of a given search.

The *2-level tree* [6] has for many years been considered the most practical choice for representing the tour, retaining that reputation until the recent emergence of the *k-level satellite tree* described herein. A worst-case cost of  $O(\sqrt{n})$  for tour operations may be achieved using the 2-level tree representation. The idea is to divide the tour into roughly  $\sqrt{n}$  segments, where each segment is maintained as a doubly linked list and the segments are connected in a doubly linked list.

The *k-level satellite tree* takes the segmentation idea a step further: the tour is divided into segments

containing roughly  $n^{1/k}$  nodes each, and the resulting segments are grouped into *parent segments* containing about  $n^{1/k}$  segments each. Ultimately,  $k - 1$  groupings are performed, giving the tree  $k$  levels with at least  $n^{1/k}$  parents on the top level. The leveraging effect achieved by this grouping of nodes into segments is the same as that achieved by the 2-level tree, except that we no longer assume that “2” is always the appropriate number of levels.

The 2-level tree representation reduces the time complexity of move operations but pays for it with slightly larger constant costs, also called *overhead*. One might guess that choosing higher values for  $k$  (making the tree “taller”) would further reduce complexity while driving up overhead. It turns out that when these costs are balanced, the best value for  $k$  increases logarithmically with  $n$ , but only approximately, since  $k$  must be integer. A related property is that, in most cases, the ideal size of a segment will remain the same as problem size increases. This can be shown algebraically under the assumption, simply put, that a given algorithm will splice the tree during moves about as often as it will traverse parents. Therefore, the key to choosing  $k$  is discovering the ideal segment size. This value, however, varies depending on the design and tuning of a given algorithm, and therefore should be determined experimentally.

Some of the overhead associated with introducing additional levels may be defrayed by utilizing a

*satellite* design [22]. A satellite list is similar to a doubly linked list but is symmetric in that an orientation is not inherent. Furthermore, there are no drawbacks in its practical use. The traditional 2-level tree incorporates doubly linked lists. If these are replaced with satellite lists, many of the query operations required in the course of a given algorithm may be performed more quickly than would be possible otherwise. This benefit becomes more pronounced when the tree is expanded to include more than two levels.

In summary, the tour is most efficiently represented with a  $k$ -level satellite tree in which  $k$  is chosen appropriately. The best value for  $k$  can be calculated according to the size of the instance and the ideal segment size, which is unique to each algorithm implementation and must be determined experimentally.

Recent experiments show the  $k$ -level satellite tree representation to be far more efficient than its predecessors. Particularly outstanding reductions in algorithm running times occur with large problem instances. When the tree is created with  $k$  chosen optimally in comparison to  $k=2$ , the average running time reduction balloons from a modest 7% for 1,000 node problems to 27% for 10,000 node problems and to 71% for 100,000 node problems. For these tests, a S&C algorithm implemented with the  $k$ -level satellite tree was run on Euclidean instances from the DIMACS Challenge [12].

Fortunately, leading ejection chain algorithms for the TSP are similar enough that they may all make use of the same data structures. Consequently, the improvement offered by the  $k$ -level satellite tree may be shared to a common advantage in the same way that the 2-level tree has found welcome in multiple implementations.

### III. ASYMMETRIC TSP

#### A. Ejection chain based algorithms

The Kanellakis-Papadimitriou (KP) heuristic [19] (based on the LK procedure) was the only local search ejection chain algorithm for the ATSP submitted to the “8<sup>th</sup> DIMACS Implementation Challenge” as reported by Johnson and McGeoch in the Challenge summary chapter [15]. The other two algorithms presented here were not submitted to the Challenge. These are the ATSP version of the S&C algorithm described in the previous section and a new approach for the ATSP using the doubly-rooted stem-and-cycle reference structure [12].

#### Kanellakis-Papadimitriou Heuristic (KP-JM)

Lin and Kernighan were not concerned with the ATSP when they developed their TSP heuristic in 1973 [20]. Since LK is based on 2-opt moves which always imply segment reversals that entail

exceedingly high computational effort, this method can not be directly applied to the ATSP. A variant of the LK approach presented by Kanellakis and Papadimitriou in 1980 [19] solved this problem by using segment reordering instead of segment reversals (creating and breaking cycles so that the resulting sequence corresponds to a sequence of 3-opt moves). The KP method starts with a variable-depth search based on LK but where the performed moves correspond to a  $k$ -opt move for odd values of  $k \geq 3$ . When the variable-depth search fails to improve the solution, the method searches for an improving 4-opt *double-bridge* move (with no reversals). Then KP returns to variable-depth search and iterates in this manner until neither of the searches improves the tour.

The KP algorithm implementation analyzed in this paper is due to Johnson and McGeoch and described in Cirasella et al. [5]. It takes advantage of the same speedup techniques used in the authors’ LK implementation [16], such as neighbor lists and “don’t look bits”. It also uses the dynamic programming approach introduced by Glover [13] to find the best 4-opt move in  $O(n^2)$  time.

#### Rego, Glover, Gamboa Stem-and-Cycle (SC-RGG)

This algorithm is based on the STSP algorithm by the same authors but ignores moves that generate path reversals. This implementation does not use candidate lists to reduce the neighborhood size which penalizes the computation times as is shown in Table V and discussed in the following subsection.

#### Rego, Glover, Gamboa and Osterman Doubly-Rooted S&C (DRSC-RGGO)

The main distinguishing feature of this approach is its use of the doubly-rooted reference structure defined in Glover [12], which generalizes the S&C structure by allowing for additional moves on each level of the ejection chain. The doubly rooted structure has two forms: a *tricycle* in which the two roots are connected by three paths, thereby generating three cycles and a *bicycle* in which the roots are connected by a single path, joining two cycles (see Figure 2 where  $r_1$  and  $r_2$  indicate the roots).

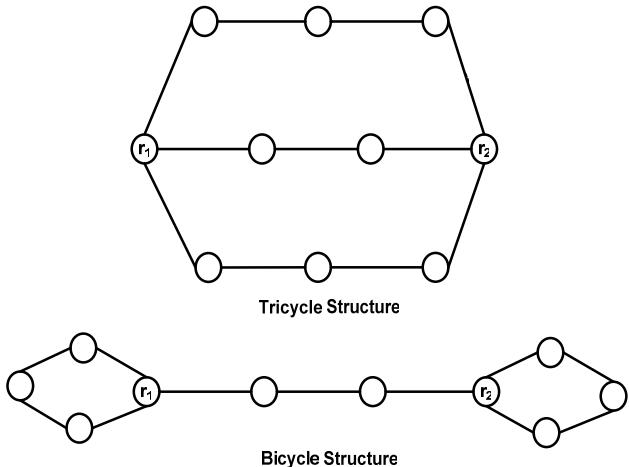


Fig. 2. The Doubly-Rooted reference structure

Ejection moves consist of adding a new edge  $(s, j)$ , where  $s$  is a subroot and deleting the edge  $(s, r)$  resulting in node  $j$  as the new root.

In this implementation additional features are integrated in the ejection chain process. At the beginning of each level of the chain a 4-opt *double-bridge* move is selected and applied, and the ejection move selected at that level is the best of  $n$  where the subroot involved in the move is randomly chosen. The results reported for this algorithm correspond to a preliminary study since its development is ongoing. For details on this implementation, we refer the reader to [10].

It is worth noting that all the implementations use *Nearest Neighbor* starting tours.

### B. Comparative analysis of performance

Table V shows the results reported for the three algorithms on all the TSP Library [25] asymmetric instances with at least 100 nodes. The table shows the percentage deviation above the optimal solution (%), the running times in seconds and average of both values for each algorithm. The best solution for each instance is indicated in bold.

Since, for our preliminary ATSP results, we did not use the benchmark code for times normalization, it is important to refer the machines where the tests were carried out. The SC-RGG and DRSC-RGGO algorithms used an Intel Centrino 1.5 GHz processor with 128 MB of RAM, and results were obtained in a single run of the algorithm with fixed parameters. The results for the KP-JM algorithm are averages over at least 5 runs for each instance, on a Silicon Graphics Power Challenge machine with 31 196 MHz MIPS R10000 processors, 1 MB 2<sup>nd</sup> level caches and 7.6 GB of main memory shared by all processors.

From Table V we can infer that the SC-RGG algorithm obtains competitive results but the DRSC-RGGO approach is clearly more effective in

producing high quality solutions. The latter achieves 16 best solutions (and 9 optimal values) as opposed to the 4 best solutions (and 3 optimal values) of the KP implementation. The overall percentage deviation average is extremely better for the doubly-rooted S&C approach, although the computational times are higher.

TABLE V  
SOLUTION QUALITY & COMPUTATIONAL TIME

Problem	KP-JM		SC-RGG		DRSC-RGGO	
	%	CPU	%	CPU	%	CPU
atex600	4,25	3,38	6,97	61,44	<b>3,54</b>	900,5
big702	2,10	6,04	3,54	104,59	<b>1,58</b>	432,59
Code198	<b>0,00</b>	0,54	<b>0,00</b>	1,17	<b>0,00</b>	0,06
Code253	0,10	1,09	0,35	2,06	<b>0,00</b>	25,64
dc112	0,39	15,47	0,91	0,83	<b>0,14</b>	16,47
dc126	0,65	22,69	1,68	1,37	<b>0,20</b>	47,59
dc134	0,57	13,43	0,80	1,50	<b>0,21</b>	20,56
dc176	0,67	20,48	2,39	1,22	<b>0,19</b>	70,48
dc188	0,59	12,98	1,19	2,86	<b>0,22</b>	97,77
dc563	<b>0,79</b>	111,95	2,47	29,08	0,93	175,30
dc849	<b>0,62</b>	114,80	0,63	64,56	0,63	134,20
dc895	0,60	144,43	2,18	136,11	<b>0,58</b>	1918,80
dc932	<b>0,26</b>	119,17	1,23	118,97	0,40	1514,72
ftv100	3,11	0,40	1,45	0,70	<b>0,00</b>	4,84
ftv110	4,04	0,40	1,28	1,09	<b>0,31</b>	15,36
ftv120	3,12	0,50	1,80	1,27	<b>0,92</b>	11,42
ftv130	2,16	0,60	2,90	1,26	<b>0,26</b>	10,73
ftv140	3,15	0,60	2,23	1,55	<b>0,25</b>	42,77
ftv150	4,43	0,70	2,68	1,51	<b>1,80</b>	11,80
ftv160	5,89	0,70	5,63	2,28	<b>0,00</b>	21,13
ftv170	4,44	0,90	3,59	2,72	<b>0,11</b>	106,89
rbg323	0,78	3,71	1,06	16,81	<b>0,08</b>	40,25
rbg358	1,50	3,33	3,44	28,92	<b>0,00</b>	50,91
rbg403	0,22	9,00	0,28	43,71	<b>0,00</b>	21,02
rbg443	0,11	11,74	1,10	67,07	<b>0,00</b>	4,95
td100.1	<b>0,00</b>	0,20	0,09	1,29	<b>0,00</b>	1,44
td1000.20	<b>0,01</b>	7,29	0,06	691,82	0,10	2733,70
td316.10	<b>0,00</b>	3,87	0,17	33,00	<b>0,00</b>	53,05
Average	1,59	22,51	1,86	50,74	<b>0,44</b>	303,03

### IV. CONCLUDING REMARKS

In this paper we describe and compare the most effective and efficient local search ejection chain algorithms for the TSP. We conclude that the S&C approach clearly outperforms the basic LK implementations. It is also clear that gains in performance are accomplished by enhancing the variable-depth search by using  $k$ -opt moves with  $k \geq 4$  and by more effective candidate lists that narrow the neighborhood size without losing solution quality. We may also add that ejection chain methods not only lead the local search TSP algorithms but also give the overall best solutions

when the local search algorithms described here are used as engines for iterated local search heuristics.

## REFERENCES

- [1] D. Applegate, R. Bixby, V. Chvátal, and W. Cook, "Concorde: A code for solving Traveling Salesman Problems," 1999, <http://www.math.princeton.edu/tsp/concorde.html>.
- [2] D. Applegate, R. Bixby, V. Chvátal, and W. Cook, "Finding Tours in TSP," Research Institut for Discrete Mathematics, Universitat Bonn, Bonn, Germany, 99885, 1999.
- [3] D. Applegate, W. Cook, and A. Rohe, "Chained Lin-Kernighan for Large Traveling Salesman Problems," *INFORMS Journal on Computing*, vol. 15, pp. 82-92, 2003.
- [4] N. Christofides and S. Eilon, "Algorithms for Large-Scale Traveling Salesman Problems," *Operations Research Quarterly*, vol. 23, pp. 511-518, 1972.
- [5] J. Cirasella, D. S. Johnson, L. A. McGeoch, and W. Zhang, "The Asymmetric Traveling Salesman Problem: Algorithms, Instance Generators and Tests," presented at Algorithm Engineering and Experimentation, Third International Workshop, ALENEX 2001, 2001.
- [6] M. L. Fredman, D. S. Johnson, L. A. McGeoch, and G. Ostheimer, "Data Structures for Traveling Salesman," *Journal of Algorithms*, vol. 18, pp. 432-479, 1995.
- [7] B. Funke, T. Grünert, and S. Irnich, "A Note on Single Alternating Cycle Neighborhoods for the TSP," Lehr- und Forschungsgebiet Operations Research und Logistik Management, RWTH Aachen, Germany 2003.
- [8] D. Gamboa, C. Rego, and F. Glover, "Implementation Analysis of Efficient Heuristic Algorithms for the Traveling Salesman Problem," *Computers and Operations Research*, 2004, in press.
- [9] D. Gamboa, C. Rego, and F. Glover, "Data Structures and Ejection Chains for Solving Large-Scale Traveling Salesman Problems," *European Journal of Operational Research*, vol. 160, pp. 154-171, 2005.
- [10] D. Gamboa, C. Rego, F. Glover, and C. Osterman, "A Doubly-Rooted Stem-and-Cycle Ejection Chain Algorithm for the Asymmetric Traveling Salesman Problem: Preliminary Study," 2005, in preparation.
- [11] F. Glover, "New Ejection Chain and Alternating Path Methods for Traveling Salesman Problems," *Computer Science and Operations Research*, pp. 449-509, 1992.
- [12] F. Glover, "Ejection Chains, Reference Structures and Alternating Path Methods for Traveling Salesman Problems," *Discrete Applied Mathematics*, vol. 65, pp. 223-253, 1996.
- [13] F. Glover, "Finding a Best Traveling Salesman 4-Opt Move in the Same Time as a Best 2-Opt Move," *Journal of Heuristics*, vol. 2, pp. 169-179, 1996.
- [14] K. Helsgaun, "An Effective Implementation of the Lin-Kernighan Traveling Salesman Heuristic," *European Journal of Operational Research*, vol. 126, pp. 106-130, 2000.
- [15] D. S. Johnson, G. Gutin, L. A. McGeoch, A. Yeo, W. Zhang, and A. Zverovitch, "Experimental Analysis of Heuristics for the ATSP," in *The Traveling Salesman Problem and Its Variations*, G. Gutin and A. Punnen, Eds. Boston: Kluwer Academic Publishers, 2002, pp. 445-487.
- [16] D. S. Johnson and L. A. McGeoch, "The Traveling Salesman Problem: A Case Study in Local Optimization," in *Local Search in Combinatorial Optimization*, E. H. L. Aarts and J. K. Lenstra, Eds.: John Wiley and Sons, Ltd., 1997, pp. 215-310.
- [17] D. S. Johnson and L. A. McGeoch, "Experimental Analysis of Heuristics for the STSP," in *The Traveling Salesman Problem and Its Variations*, G. Gutin and A. Punnen, Eds. Boston: Kluwer Academic Publishers, 2002, pp. 369-443.
- [18] D. S. Johnson, L. A. McGeoch, F. Glover, and C. Rego, "8<sup>th</sup> DIMACS Implementation Challenge: The Traveling Salesman Problem," 2000, <http://www.research.att.com/~dsj/chfsp/>.
- [19] P. C. Kanellakis and C. H. Papadimitriou, "Local search for the asymmetric traveling salesman problem," *Operations Research*, vol. 28, pp. 1086-1099, 1980.
- [20] S. Lin and B. Kernighan, "An Effective Heuristic Algorithm for the Traveling Salesman Problem," *Operations Research*, vol. 21, pp. 498-516, 1973.
- [21] D. Neto, *Efficient Cluster Compensation for Lin-Kernighan Heuristics*.: Department of Computer Science, University of Toronto, 1999.
- [22] C. Osterman, C. Rego, and D. Gamboa, "The Satellite List: A Reversible Doubly-Linked List," presented at 7th International Conference on Adaptive and Natural Computing Algorithms (ICANNGA 2005), Coimbra, Portugal, 2005.
- [23] C. Rego, "Relaxed Tours and Path Ejections for the Traveling Salesman Problem," *European Journal of Operational Research*, vol. 106, pp. 522-538, 1998.
- [24] C. Rego and F. Glover, "Local Search and Metaheuristics," in *The Traveling Salesman Problem and Its Variations*, G. Gutin and A. Punnen, Eds. Dordrecht: Kluwer Academic Publishers, 2002, pp. 309-368.
- [25] G. Reinelt, "TSPLIB - A Traveling Salesman Problem Library," *ORSA Journal on Computing*, vol. 3, pp. 376-384, 1991.

# Parallel machine scheduling with resource dependent processing times

Raúl Cortés\*, Jose Pedro García\*, Rafael Pastor† and Carlos Andrés\*

\*Universidad Politécnica de Valencia. Dpto. Organización de Empresas, Economía Financiera y Contabilidad  
Camino Vereda s/n. 46022 Valencia. Spain

Email: racorfi@omp.upv.es, jpgarcia@omp.upv.es, candres@omp.upv.es

†Universidad Politécnica de Cataluña  
C/Jordi Girona, 31. 08034 Barcelona. Spain

Email: rafael.pastor@upc.edu

**Abstract**— We consider the problem of scheduling jobs on various parallel machines under single resource constraints, where the processing time of a job depends on both the machine used to process the job and the amount of allocated resource. The aim is to minimize the total weighted earliness and tardiness with regard to the programmed delivery date. A complete review of similar problems that have been analyzed in the literature is presented. To conclude, an original formulation for the problem is proposed.

**Keywords**— Parallel machine scheduling, resource constraints, variable speed.

## I. INTRODUCTION

Traditionally, scheduling models have treated job processing times as parameters of the problem. However, in multiple productive environments it is possible to control the processing times by reallocating some of the available resources. This kind of problems require not only the assignment of the jobs to the machines, but also the choice of resource allocation, which in most cases tends to increase the problem complexity. In the present work, we analyze the problem of scheduling  $n$  jobs on  $m$  unrelated parallel machines, with controllable processing times by the allocation of a limited resource. As a general rule, there are three different types of parallel machines considered in the literature. When the processing time of a job does not depend on the machine used to process the job, ( $p_{ij} = p_i \forall j$ ) the machines are defined as identical parallel machines. On uniform parallel machines, the processing time of a job depends on the machine speed ( $p_{ij} = \frac{p_i}{s_j}$ ). Finally on unrelated parallel

machines, given a pair of jobs and a pair of machines, there is no relationship between the processing times of the two jobs on each one of the two machines.

Thus, the problem could be outlined as follows. Let  $N=\{1,2,\dots,n\}$  be the set of jobs (indexed by  $i$  or  $l$ ) that are to be processed on  $m$  unrelated parallel

machines (indexed by  $j$ ). Each job is to be processed without pre-emption on one of the machines, which can only handle one job at a time. To process the jobs on each machine, there exists a single, discrete and renewable resource, indexed by  $r$ . Let  $R$  denote the total amount of this resource, which is a constant through the planning horizon. The processing time of job  $i$  on machine  $j$  depends on the amount of resource  $r$  allocated to the machine. This processing time will be denoted as  $P_{ijr}$ . In the particular case analyzed in this paper, once the allocation of resource  $r$  to machine  $m$  is made, it remains constant to process all the jobs on that machine. Finally, for each job on each machine there exists a minimum ( $R_{min_{ij}}$ ) and a maximum ( $R_{max_{ij}}$ ) amount of resource units that could be allocated to that job on that machine.

For each job there is a programmed delivery date ( $D_i$ ), and an earliness and tardiness penalty weight ( $a_i, b_i$ ). Being  $C_i$  the completion time of job  $i$ , we define  $E_i = \max \{0, D_i - C_i\}$  and  $T_i = \max \{0, C_i - D_i\}$  the earliness and tardiness of job  $i$  respectively.

By using the widely accepted notation scheme for machine scheduling proposed by Graham et al.[9] we denote our problem as  $R/\text{res} / \sum_i (a_i E_i + b_i T_i)$ .

This problem is known to be NP-Hard (see Chen [3]) since the same problem without the added complexity of resource allocation is already NP-Hard.

The resolution of the problem focused on proves to be of great interest, since it can be easily extended and adapted to many specific problems in the industry, by assuming certain simplifications. The present study is framed inside a global research line that faces the application and adaptation of classic Operations Management tools in productive environments with disabled workers. Particularly, the industrial reality of Sheltered Work Centers for Disabled could be modelled by parallel machines with different speed, where the job processing times

depend also on the number of workers assigned to the job. However, the authors observed that this model can be extended to very disparate industries, such as varnishes manufacturing in chemical industry or components manufacturing in automotive industry.

## II. LITERATURE REVIEW

The first study on job scheduling involving processing times controlled by resource allocation was initiated by Vickson [15]. Since then, and especially in the last decade, the problem has received increasing attention. However, depending on the consideration of some important parameters (e.g. resource characteristics, machines, objectives), very different approaches to the resource allocation problem can be found in the literature.

Concerning resources characteristics, the distinction between renewable and not renewable resources is usually made. In problems involving renewable resources the total amount of resource is limited (e.g. manpower) and the job processing times are discrete depending on the amount of resource allocated to the job, see Daniels et al. [6], [7]. In Kellerer and Strusevich [10] a particular case of renewable resources where job processing times are not controlled by resource allocation analyze. Later on Kellerer and Strusevich [11] extend their work with the consideration of multiple resources. This kind of scheduling problems is known as scheduling under resource constraints problems. On the other hand, not renewable resources (e.g. electricity, fuel) usually imply continuous processing times and a related cost in the objective function, since the amount of resource is not limited. Interesting studies dealing with not renewable resources are Alidaee and Ahmadian [1], Cheng et al. [5], [4] among others.

As regards machines, we have focused particularly on parallel machine scheduling problems. A very exhaustive review of these scheduling problems can be found in [12]. To the best of our knowledge, only Trick [14] considers the problem of resource allocation with unrelated parallel machines. Alidaee and Ahmadian [1] and Chen [3] studied this problem with identical parallel machines. By means of an interesting transformation, Chen extends his previous studies on parallel machines (see Chen and Powell [2]) to the problem involving resource allocation. The author exploits a body of work that has produced interesting results that consists of transforming the problem into a set covering model. In this method,

the linear programming relaxation of the problem is solved first. Then a variety of techniques, such as cutting plane or branch-and-bound, are used to find an integer solution to the set covering problem. In the works of Daniels et al. [6], [7], and the works of Kellerer and Strusevich [10], [11] the machines are dedicated, i.e. the assignments of jobs to machines is specified. Daniels and Mazzola [8] and Nowicki [13] consider the problem of resource allocation and controllable processing times in flow shop environments. Finally, Cheng et al. [4], [5] studied the case of a single machine.

With regard to jobs, usually the studies involving processing times controlled by resource allocation consider non pre-emptive jobs. Only the works of Kellerer and Strusevich [10], [11] consider the possibility of interruption.

Finally, with reference to objectives, the usual scheduling criterions are considered. In Daniels et al. [6], [7], Daniels and Mazzola [8], and Kellerer and Strusevich [10], [11] the aim is to minimize the maximum completion time  $C_{max}$ . As a general rule, the renewable resources characteristic appears as a constraint in the model, and not as a cost to minimize. On the other hand, if the resources are not renewable, usually a cost related to the total resource consumption appears as a cost to minimize. To be precise, in most of the cases an overall cost function, which consists of a resource allocation cost and a scheduling cost, is to minimize. For instance, in Alidaee and Ahmadian [1] the aim is to minimize the production costs (related to resource consumption) plus total weighted earliness and weighted tardiness. Similarly, Trick [14] considers the minimization of processing costs plus  $C_{max}$ , and in Cheng et al. [5] and Chen [3] the objective is to minimize the resource allocation costs plus weighted number of tardy jobs.

As a general rule, parallel machine scheduling problems, due to their high analytical complexity, have been approached by means of heuristic procedures where their performance is evaluated by ratios that allow for comparisons to be made.

Only Daniels et al. [6], [7], Daniels and Mazzola [8] and Chen [3] worked on exact solution algorithms for NP-Hard problems in this area.

To the best of our knowledge, there are no formulations to be found in the literature for the problem of minimizing a delivery date related cost function, for unrelated parallel machines with processing times controlled by resource allocation.

## III. MODEL FORMULATION

For the model formulation, besides the indexes and parameters described in section I, a new index  $k=1,\dots,n$ , has to be defined, referred to the position a job takes in the machine sequence. Thus, we define the following binary variables:

$$x_{ijkr} \begin{cases} 1, & \text{if job } i \text{ is processed on machine } j \text{ in position } k \\ & \text{with } r \text{ resources} \\ 0, & \text{otherwise} \end{cases}$$

$$y_{jr} \begin{cases} 1, & \text{if } r \text{ resources are allocated to machine } j \\ 0, & \text{otherwise} \end{cases}$$

In addition, a binary variable,  $\delta_{ijk}$ , will be used to force the following condition, which will be expressed in the model by means of constraints (7) and (8)

$$x_{ijkr} \wedge x_{ljk-1r} = 1 \rightarrow C_l + P_{ijr} \leq C_i \quad (1)$$

Therefore, we define:

$$\delta_{ijk} \begin{cases} 1, & \text{if } x_{ijkr} = x_{ljk-1r} = 1 \\ 0, & \text{otherwise} \end{cases}$$

Hence, the scheduling problem we are considering can be formulated as follows:

$$\min \sum_{i=1}^n (a_i E_i + b_i T_i) \quad (2)$$

s.t.

$$E_i \geq D_i - C_i \quad \forall i \quad (3)$$

$$T_i \geq C_i - D_i \quad \forall i \quad (4)$$

$$\sum_{j=1}^m \sum_{k=1}^{R_{\max,j}} \sum_{r=R_{\min,j}}^n x_{ijkr} = 1 \quad \forall i \quad (5)$$

$$\sum_{i=1}^n \sum_{r=R_{\min,j}}^{R_{\max,j}} x_{ijkr} \leq 1 \quad \forall k, \forall j \quad (6)$$

$$\sum_{i=1}^n \sum_{r=R_{\min,j}}^{R_{\max,j}} x_{ijk+1r} \leq \sum_{i=1}^n \sum_{r=R_{\min,j}}^{R_{\max,j}} x_{ijkr} \quad \forall j, k \quad (7)$$

$$\sum_{r=R_{\min,j}}^{R_{\max,j}} x_{ijkr} + \sum_{r=R_{\min,j}}^{R_{\max,j}} x_{ljk-1r} - \delta_{ijk} \leq 1 \quad \forall i, l, j, k \quad (8)$$

$$C_l + \sum_{r=R_{\min,j}}^{R_{\max,j}} x_{ijkr} \cdot P_{ijr} + M \cdot \delta_{ijk} \leq M + C_i \quad \forall i, l, j, k \quad (9)$$

$$C_i \geq \sum_{j=1}^m \sum_{r=R_{\min,j}}^{R_{\max,j}} x_{ij1r} \cdot P_{ijr} \quad \forall i \quad (10)$$

$$\sum_{r=1}^R y_{jr} \leq 1 \quad \forall j \quad (11)$$

$$\sum_{r=R_{\min,j}}^{R_{\max,j}} \sum_{k=1}^n r \cdot x_{ijkr} \leq \sum_{r=1}^R r \cdot y_{jr} \quad \forall j, i \quad (12)$$

$$R - \sum_{r=R_{\min,j}}^{R_{\max,j}} \sum_{k=1}^n (r-R) \cdot x_{ijkr} \geq \sum_{r=1}^R r \cdot y_{jr} \quad \forall j, i \quad (13)$$

$$\sum_{r=1}^R \sum_{j=1}^m r \cdot y_{jr} \leq R \quad (14)$$

$$T_i, E_i, C_i \geq 0 \quad \forall i \quad (15)$$

Where  $M$  is an upper bound on the schedule makespan.

Equations (3), (4), and (15) define  $E_i = \max_i(0, D_i - C_i)$  and  $T_i = \max_i(0, C_i - D_i)$  for earliness and tardiness of job  $i$ .

The set of constraints expressed by equation (5) expresses that each job is to be processed once. Constraints (6) ensure that there will be only one job or less processed in  $k$ -th place on machine  $j$ . Constraints (7) force that the positions of the jobs in the sequence of a machine are consecutive. Constraints (8) and (9) and (10) define the completion time of the jobs: by means of the binary variable  $\delta_{ijk}$  the condition expressed in (1) is forced in constraints (8) and (9), whereas constraints (10) are used to guarantee that the completion time of a job is at least its processing time when it is processed in first place in the sequence.

Constraints (11) ensure one resource allocation at the most for each machine, and constraints (12) and (13) require that all the jobs processed on machine  $j$  have the same amount of resource allocated. Finally, constraint (14) limits the total amount of resource allocated among the machines.

#### IV. CONCLUSIONS

We have analyzed the last studies referring scheduling with job processing times controlled by resource allocation, and the different approaches to similar problems in this area. An original model for a due-date-related objective in an environment with unrelated parallel machines for job scheduling and resource allocation has been presented. Due to the model structure it seems rather improbable that optimal solutions can be found for problems with a large size, in a reasonable time. At the moment test problems are being generated to evaluate the performance of the model with regard to the size of the problem. Further investigations should be carried out in order to develop particular resolution algorithms to solve problems with a larger size.

## APPENDIX

## Appendix A. Notation

Term	Definition
$i, l$	Indexes for jobs
$j$	Index for machines
$k$	Index for position
$r$	Index for resources
$n$	Number of jobs
$M$	Number of machines
$R$	Amount of resources
$D_i$	Due date for job $i$
$P_{ijr}$	Processing time for job $i$ on machine $j$ with $r$ resources
$a_i$	Earliness penalty weight (per time unit) for job $i$
$b_i$	Tardiness penalty weight (per time unit) for job $i$
$E_i$	Earliness of job $i$
$T_i$	Tardiness of job $i$

## REFERENCES

- [1] B. Alidaee and A. Ahmadian. 2 Parallel Machine Sequencing Problems Involving Controllable Job Processing Times. *European Journal of Operational Research* 70 (3):335-341, 1993.
- [2] Z. L. Chen and W. B. Powell. Solving parallel machine scheduling problems by column generation. *Informs Journal on Computing* 11 (1):78-94, 1999.
- [3] Z. L. Chen. Simultaneous job scheduling and resource allocation on parallel machines. *Annals of Operations Research* 129 (1-4):135-153, 2004.
- [4] T. C. E. Cheng, A. Janiak, and M. Y. Kovalyov. Bicriterion single machine scheduling with resource dependent processing times. *Siam Journal on Optimization* 8 (2):617-630, 1998.
- [5] T. C. E. Cheng, Z. L. Chen, and C. L. Li. Single-machine scheduling with trade-off between number of tardy jobs and resource allocation. *Operations Research Letters* 19 (5):237-242, 1996.
- [6] R. L. Daniels, B. J. Hoopes, and J. B. Mazzola. An analysis of heuristics for the parallel-machine flexible-resource scheduling problem. *Annals of Operations Research* 70:439-472, 1997.
- [7] R. L. Daniels, B. J. Hoopes, and J. B. Mazzola. Scheduling parallel manufacturing cells with resource flexibility. *Management Science* 42 (9):1260-1276, 1996.
- [8] R. L. Daniels and J. B. Mazzola. Flow-Shop Scheduling with Resource Flexibility. *Operations Research* 42 (3):504-522, 1994.
- [9] R. L. Graham, Lawler E.L, Lenstra, J.K. and Rinnooy Kan A.H.G. Optimization and approximation in deterministic sequencing and scheduling: A survey. *Ann. Discrete Mathematics*, 5:287--326, 1979.
- [10] H. Kellerer and V. A. Strusevich. Scheduling parallel dedicated machines under a single non-shared resource. *European Journal of Operational Research* 147 (2):345-364, 2003.
- [11] H. Kellerer and V. A. Strusevich. Scheduling problems for parallel dedicated machines under multiple resource constraints. *Discrete Applied Mathematics* 133 (1-3):45-68, 2003.
- [12] E. Mokotoff. Parallel machine scheduling problems: A survey. *Asia-Pacific Journal of Operational Research* 18 (2):193-242, 2001.
- [13] E. Nowicki. An Approximation Algorithm for the M-Machine Permutation Flow-Shop Scheduling Problem with Controllable Processing Times. *European Journal of Operational Research* 70 (3):342-349, 1993.
- [14] M. A. Trick. Scheduling Multiple Variable-Speed Machines. *Operations Research* 42 (2):234-248, 1994.
- [15] R. G. Vickson. 2 Single-Machine Sequencing Problems Involving Controllable Job Processing Times. *Aiee Transactions* 12 (3):258-262, 1980.

# A Tabu Search algorithm for two-dimensional non-guillotine cutting problems

Francisco Parreño\*, Ramón Ivarez-Valds† and J. Manuel Tamarit†

\*University of Castilla-La Mancha. Department of Computer Science  
E.Politecnica Superior, 02071 Albacete, Spain

Email: fparreno@info-ab.uclm.es

†University of Valencia, Department of Statistics and Operations Research.  
46100 Burjassot, Valencia, Spain  
Email: ramon.alvarez@uv.es

**Abstract**—In this paper we study the two-dimensional non-guillotine cutting problem, the problem of cutting rectangular pieces from a large stock rectangle so as to maximize the total value of the pieces cut. The problem has many industrial applications whenever small pieces have to be cut from or packed into a large stock sheet. We propose a tabu search algorithm. Several moves based on reducing and inserting blocks of pieces have been defined. Intensification and diversification procedures, based on long-term memory, have been included. The computational results on large sets of test instances show that the algorithm is very efficient for a wide range of packing and cutting problems.

**Keywords**—Non-guillotine cutting; heuristics; Tabu Search

## I. INTRODUCTION

THE two-dimensional non-guillotine cutting problem consists of cutting a given finite set of small rectangular pieces from a large stock rectangle of fixed dimensions with maximum profit. The problem appears in many production processes in the textile, paper, steel, glass and wood industries.  $R = (L, W)$  is the large stock rectangle of length  $L$  and width  $W$ . Each piece  $i$  has dimensions  $(l_i, w_i)$ , and value  $v_i$ ,  $i = 1, \dots, m$ . The pieces have fixed orientation and must be cut with their edges parallel to the edges of the stock rectangle (orthogonal cuts). The problem is to cut off the rectangle  $R$  into  $x_i$  copies of each piece  $i$ , such that  $0 \leq P_i \leq x_i \leq Q_i$ , and the total values of the pieces cut,  $\sum_i v_i x_i$  is maximized. We will denote by  $M = \sum_i Q_i$  the maximum number of pieces which could be cut.

According to the values of  $P_i$  and  $Q_i$  we can distinguish three types of problems:

- 1) *Unconstrained*:

$\forall i, P_i = 0, Q_i = \lfloor L * W / l_i * w_i \rfloor$  (trivial bound).

- 2) *Constrained*:

$\forall i, P_i = 0; \exists i, Q_i < \lfloor L * W / l_i * w_i \rfloor$

- 3) *Doubly constrained*:

$\exists i, P_i > 0; \exists j, Q_j < \lfloor L * W / l_j * w_j \rfloor$

In Figure 1 we see an example with a stock rectangle of  $R = (10, 10)$ , and  $m = 10$  pieces to be cut. The first solution (Figure 1(b)) is optimal for the unconstrained problem, while the second solution (Figure 1(c)) corresponds to the constrained problem and the third (Figure 1(d)) to the doubly constrained problem, with some  $P_i \neq 0$ .

Some authors have considered the unconstrained problem: Tsai et al. [20], Arenales and Morabito [3], Healy et al. [11]. Nevertheless, the constrained problems are more interesting for applications and more research has been devoted to this case. Some exact methods have been proposed by Beasley [4], Scheithauer and Terno [18], Hadjiconstantinou and Christofides [10], Fekete and Schepers [8], and Caprara and Monaci [6].

A simple upper bound for the problem can be obtained by solving the following bounded knapsack problem, where variable  $x_i$  represents the number of pieces of type  $i$  cut in excess of its lower bound  $P_i$ :

$$\text{Max} \quad \sum_{i=1}^m v_i x_i + \sum_{i=1}^m v_i P_i \quad (1)$$

$$\text{s.t. :} \quad \sum_{i=1}^m (l_i w_i) x_i \leq LW - \sum_{i=1}^m P_i (l_i w_i) \quad (2)$$

$$x_i \leq Q_i - P_i, \quad i = 1, \dots, m \quad (3)$$

$$x_i \geq 0, \quad \text{integer}, \quad i = 1, \dots, m. \quad (4)$$

Other bounds, apart from those included in the exact methods mentioned above, have been proposed by Scheithauer and Terno [19], Amaral and Wright [2].

Several heuristic algorithms have been proposed recently. Wu et al. [22] develop a constructive algorithm for the special case in which  $P_i = Q_i \forall i$ . At each

Piece	$l_i$	$w_i$	$P_i$	$Q_i$	$v_i$
1	3	2	1	2	7
2	7	2	1	3	20
3	4	2	1	2	11
4	6	2	0	3	13
5	9	1	0	2	21
6	8	4	0	1	79
7	4	1	1	2	9
8	1	10	0	1	14
9	3	7	0	3	52
10	4	5	0	2	60

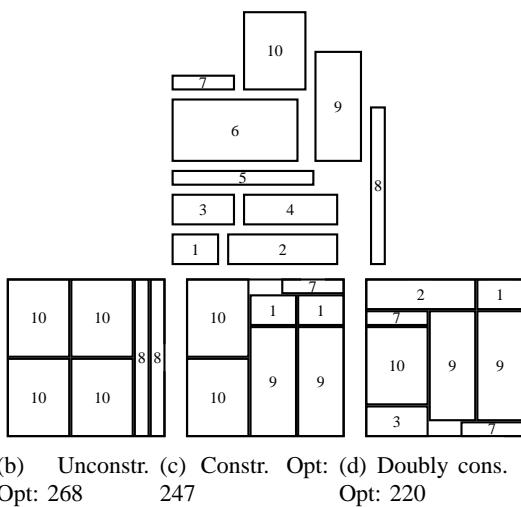


Fig. 1. Instance 3 from Beasley [4]

step a piece is cut in a corner of the current cutting pattern and the piece to cut is decided according to a fitness evaluation function which estimates the quality of the complete solution that could be obtained starting from the piece being considered. The other proposals are based on metaheuristic procedures, mainly simulated annealing and genetic algorithms. Lai and Chan [14], [15] use simulated annealing. Each solution is given by an ordered list of pieces and a list is translated into a cutting pattern by a placement algorithm. Instead of the more usual bottom-left algorithm, they propose a difference algorithm in which the piece is placed in the existing empty space which is nearest to the bottom left corner of the stock sheet. They report a limited computational experience on a small set of randomly generated instances and one real problem from a printing company. The Leung et al. [16], [17] algorithms are based on the work by Lai and Chan [14], [15] and by Jakobs [13]. Jakobs [13] develops a genetic algorithm for the related strip packing problem and uses as placement algorithm a bottom-left procedure. They combine both

metaheuristics and both placement algorithms and report computational results on a set of randomly generated instances. Beasley [5] develops a genetic algorithm based on a non-linear formulation of the problem, where variables indicate if a piece is cut or not and its position on the stock sheet. Therefore, the solutions are lists of variables and directly show the cutting pattern. No placement algorithm is needed. He presents a complete computational study on a set of standard test problems and on a number of large randomly generated problems. Alvarez-Valdes et al [1] follow a different approach and develop a GRASP algorithm. Their computational tests collect the problems used by Leung et al. [16], [17] and by Beasley [5].

In this paper, we present a Tabu Search algorithm for the two-dimensional non-guillotine cutting problem. We provide computational results obtained on four sets of test problems: the 21 problems from the literature collected by Beasley [5]; the 630 large random problems also generated by Beasley [5]; 10 problems used by Leung et al. [17], and the 21 problems used by Hopper and Turton [12]. The last set of problems were initially designed for the strip packing problem and have been adapted to the two-dimensional non-guillotine cutting problem with the aim of testing the algorithm on a set of large and difficult zero-waste instances.

## II. A CONSTRUCTIVE ALGORITHM

In this section we briefly describe a constructive algorithm that will be used in the Tabu Search algorithm. More details may be found in [1]. Constructing a solution is an iterative process in which we combine two elements: a list  $\mathcal{P}$  of pieces still to be cut, initially the complete list of pieces, and a list  $\mathcal{L}$  of empty rectangles in which a piece can be cut, initially containing only the stock rectangle  $R = (L, W)$ . At each step a rectangle is chosen from  $\mathcal{L}$ , and from the pieces in  $\mathcal{P}$  fitting in it a piece is chosen to be cut. That usually produces new rectangles going into  $\mathcal{L}$  and the process goes on until  $\mathcal{L} = \emptyset$  or none of the remaining pieces fit into one of the remaining rectangles.

### Step 0. Initialization:

$\mathcal{L} = \{R\}$ , the set of empty rectangles.

$\mathcal{P} = \{p_1, p_2, \dots, p_m\}$ , the set of pieces still to be cut.

The set  $\mathcal{P}$  is initially ordered according to 3 criteria: Order by non-increasing  $P_i * l_i * w_i$ , giving priority to pieces which must be cut. If there is a tie (for instance, if  $P_i = 0, \forall i$ ), order by non-increasing  $v_i / (l_i * w_i)$ . If there is a tie (for instance, if  $v_i = l_i * w_i, \forall i$ ), order by non-increasing  $l_i * w_i$ .

$\mathcal{B} = \emptyset$ , the set of pieces cut. Pieces of the same type may appear grouped in rectangular *blocks*.

*Step 1. Choosing the rectangle:*

Take  $R^*$ , the smallest rectangle of  $\mathcal{L}$  in which a piece  $p_i \in \mathcal{P}$  can fit.

If such  $R^*$  does not exist, stop.

Otherwise, go to Step 2.

*Step 2. Choosing the piece to cut:*

Choose a piece  $p_i$  and a quantity  $n_i \leq Q_i$ , forming block  $B^*$  to be cut in  $R^*$ .

The piece  $i$  is chosen to produce the largest increase in the objective function. Block  $B^*$  is cut in the corner of  $R^*$  which is nearest to a corner of the stock rectangle.

Update  $\mathcal{P}$ ,  $\mathcal{B}$  and  $Q_i$  which indicates the number of pieces still to be cut.

Move block  $B^*$  towards the nearest corner of the stock rectangle.

*Step 3. Updating the list  $\mathcal{L}$ :*

Add to  $\mathcal{L}$  the possible rectangles produced when cutting  $B^*$  from  $R^*$ .

Take into account the possible changes in  $\mathcal{L}$  when moving block  $B^*$ .

Merge rectangles to favor cutting new pieces of  $\mathcal{P}$ .

Go back to Step 1.

Though we keep a list of empty rectangles  $\mathcal{L}$ , we really have an irregular, polygonal empty space in which the pieces still to be cut can be considered for fitting. One way of adapting our list  $\mathcal{L}$  to the flexibility of non-guillotine cutting is to merge some of the rectangles from the list, producing some new rectangles in which the pieces to be cut could fit better.

When we merge 2 rectangles, at most 3 new rectangles may appear, usually one *large* rectangle and 2 *small* ones (see Figure 2). Among the several alternatives for merging we try to select the best, that is, the one in which it is possible to cut the pieces best situated in the ordered list  $\mathcal{P}$ . With this objective in mind, we impose some conditions:

- 1) If the order of the best piece which fits into the *large* rectangle is strictly lower than the order of the pieces in the original rectangles, we merge them.
- 2) If the order of the best piece which fits into the *large* rectangle is equal to the order of the pieces in the original rectangles, we merge them if the area of the large rectangle is bigger than the area of each one of the original rectangles.
- 3) If the order of the best piece which fits in the *large* rectangle is strictly greater than the order of the pieces in the original rectangles, we do not merge

them.

In Figure 2 we see several possible cases. In Figure 2(a) the two original rectangles will always be merged. The new rectangle is larger than them and all the pieces fitting in the original rectangles will fit in it. In Figure 2(b) the new rectangles are not larger than the original ones. These will be merged only if the new central rectangle accommodates a piece of lower order than those fitting in the original rectangles. In Figure 2(c) one of the new rectangles is larger than the original ones and therefore they will be merged unless the best piece fitting in the original vertical rectangle does not fit in the new ones.

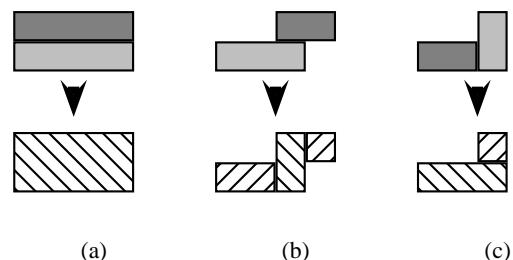


Fig. 2. Merging 2 empty rectangles

At the end of the constructive process, a solution is composed of a list of blocks  $\mathcal{B}$ , and a list of empty rectangles  $\mathcal{L}$ , with total value  $\sum_i v_i x_i$ .

### III. A TABU SEARCH ALGORITHM

*Tabu Search* is now a well-established optimization algorithm (for an introduction, refer to the book by Glover and Laguna [9]). The basic elements of the algorithm are described in the following subsections.

#### A. Definition of moves

The solution space in which we move is composed of feasible solutions only. In this space we will define several moves to go from one solution to another. An initial solution is obtained by applying the constructive algorithm described in Section 2.

We distinguish two types of moves: block reduction and block insertion. In block reduction, the size of an existing block is reduced, eliminating some of its rows or columns. In block insertion, a new block is added to the solution. For both moves we first present a scheme of the procedures and then a detailed example.

##### • Block reduction

###### Step 0. Initialization:

- $\mathcal{B}$  = the list of blocks of the current solution
- $\mathcal{L}$  = the list of empty rectangles

### Step 1. Choosing the block to reduce

Take  $B$ , one of the blocks of  $\mathcal{B}$ , with  $k$  columns and  $l$  rows of pieces  $p_i$ .

Select the number  $r$  of columns (rows) to eliminate,

$$1 \leq r \leq k \quad (1 \leq r \leq l),$$

keeping the number of pieces in the solution  $x_i \geq P_i$ .

If  $P_i = 0$ , the block may disappear completely. The new waste rectangle  $R$  is added to  $\mathcal{L}$ .

### Step 2. Move the remaining blocks to their nearest corners:

The list of waste rectangles  $\mathcal{L}$  is updated accordingly.

### Step 3. Fill the empty rectangles with new blocks:

Apply the constructive algorithm of Section 2. The algorithm starts from the current lists  $\mathcal{L}$  and  $\mathcal{B}$ , and  $\mathcal{P}$  contains the pieces which can still be cut and added to the current solution. Before proceeding to the construction process, rectangles in  $\mathcal{L}$  are considered for merging, to best accommodate the pieces of  $\mathcal{P}$ .

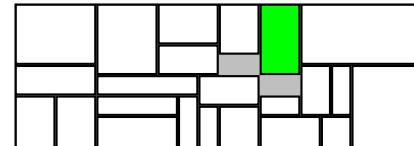
When selecting the piece to cut, the piece eliminated at Step 1 is not considered until another piece has been included in the modified solution.

### Step 4. Merge the blocks with the same structure:

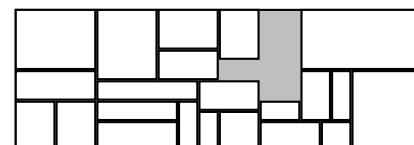
We try to merge blocks of the same piece with the same length or width if they are adjacent and have a common side, or if one of them can be moved to make them both adjacent to one common side.

In Figure 3 we see an example of a reduction move on an instance proposed by Jakobs [13] and used later by Leung et al. [17]. The stock rectangle is  $R = (120, 45)$ ,  $m = 22$  and  $M = 25$  pieces can be cut from it, completely filling it. Figure 3(a) shows a solution with 23 pieces which cannot accommodate two (6x12) pieces. The set  $\mathcal{L}$  of empty rectangles is composed of  $R_1 = (60, 24, 72, 30)$  and  $R_2 = (72, 18, 84, 24)$  (in light grey). In Step 1, a block composed of one piece (12x21) (in dark grey), is selected to be reduced and therefore it disappears from the solution, creating a new empty rectangle  $R_3 = (72, 24, 84, 45)$  which is added to  $\mathcal{L}$  (Figure 3(b)). In Step 2, a block composed of a piece (12x15) is moved to the top right corner. Therefore,  $\mathcal{L} = \{R_1, R_2, R_4, R_5\}$ , where  $R_4 =$

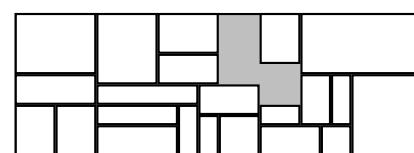
$(60, 30, 72, 45)$  and  $R_5 = (72, 24, 84, 30)$  (Figure 3(c)). In Step 3 the constructive procedure fills the empty rectangles. First,  $R_1$  and  $R_4$  are merged, forming  $R_6 = (60, 24, 72, 45)$ , and  $R_2$  and  $R_5$  are merged, forming  $R_7 = (72, 18, 84, 30)$ . Then,  $R_7$  is selected and the two pieces (6x12) are cut in it, completely filling it. Finally,  $R_6$  is taken and the piece initially eliminated is cut in it. The final solution, which is optimal, appears in Figure 3(d).



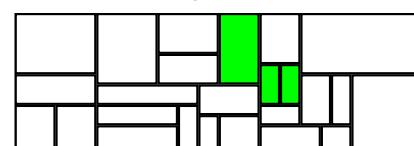
(a) Selection



(b) Reduction



(c) Moving to the corner



(d) Fill

Fig. 3. Block reduction. Instance 3 from Jakobs [13]

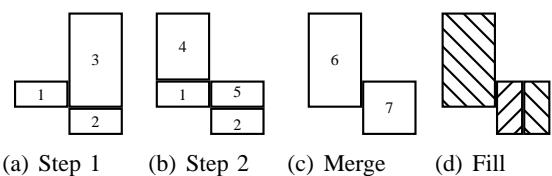


Fig. 4. Empty rectangles in the reduction move

### • Block insertion

#### Step 0. Initialization:

$\mathcal{B}$  = the list of blocks

$\mathcal{L}$  = the list of empty rectangles

#### Step 1. Choosing the block to insert

Take  $p_i$ , a piece for which  $x_i < Q_i$ , and

consider a block of these pieces with  $k$  columns and  $l$  rows ( $k * l \leq Q_i - x_i$ ).

- Step 2. Select the position to insert the new block*
- Step 3. Remove the pieces of the solution overlapping with the inserted block*

Update  $\mathcal{B}$  (some of the original blocks are reduced or eliminated)  
 Update  $\mathcal{L}$  (some new empty rectangles may appear).

- Step 4. Fill the empty rectangles with new blocks*
- Step 5. Merge the blocks with the same structure:*

Steps 4 and 5 are the same as in block reduction.

In Step 3, two strategies have been considered to select the position of the new block. In both cases, the block is placed partially or totally covering one or more empty rectangles.

- For each empty rectangle consider the four alternatives in which a corner of the rectangle is chosen for the corresponding corner of the block. If the dimensions of the block are larger than those of the rectangle, the block may overlap with other blocks or may occupy part of other empty rectangles.
- Select only one empty rectangle, that producing the largest intersection with the block if the bottom left corner of the block were placed in the bottom left corner of the rectangle. For this rectangle, the four alternatives described above are considered.

In Figure 5 we see an example of the second strategy. In Figure 5(a) two empty rectangles, in grey, are considered to accommodate the new block, in white. As the largest common area corresponds to rectangle 1, it is chosen. In Figure 5(b) the four corners of the empty rectangle are considered for situating the corner of the new block.

In Figure 6 we see an example of an insertion move on an instance proposed by Fekete and Schepers [8] and later used by Beasley [5]. The stock rectangle is  $R = (100, 100)$ ,  $m = 15$  and  $M = 50$  pieces can be cut from it. Figure 6(a) shows a solution of value  $z = 27539$ . The set  $\mathcal{L}$  of empty rectangles is composed of  $R_1 = (70, 41, 72, 81)$  and  $R_2 = (72, 80, 100, 81)$ . In Step 1 we select a piece  $i = 5$  of dimensions  $(6 \times 40)$  with  $Q_i = 5$  and only 2 copies in the current solution and consider a block  $B^*$  of one piece. In Step 2 we place  $B^*$  over  $R_1$ , selecting the top left corner of the rectangle to locate the top left corner of the block.  $B^*$  completely covers  $R_1$  and part of  $R_2$ , which becomes  $R_3 = (76, 80, 100, 81)$ .  $B^*$  also partially covers an

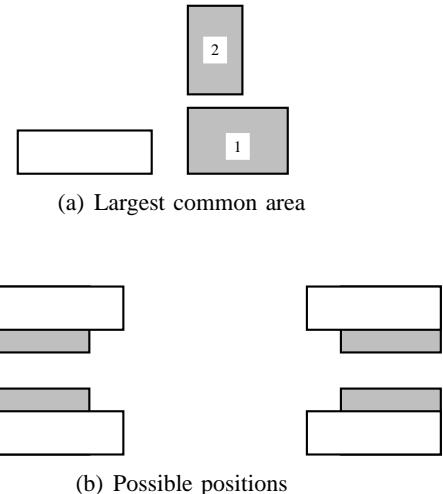


Fig. 5. Selecting the position of the new block

existing block (Figure 6(b)). Therefore, in Step 3, we remove the pieces of the initial solution overlapping with  $B^*$ . That produces two new empty rectangles  $R_4 = (76, 40, 78, 80)$  and  $R_5 = (72, 40, 76, 41)$  (Figure 6(c)). In Step 4, the filling procedure starts from this list  $\mathcal{L} = \{R_3, R_4, R_5\}$ . First,  $R_3$  and  $R_4$  are merged, producing  $R_6 = (76, 40, 78, 81)$  and  $R_7 = (78, 80, 100, 81)$ . While none of the remaining pieces could fit either in  $R_3$  or in  $R_4$ , a piece  $i = 13$  of dimensions  $(2 \times 41)$  now fits into  $R_6$ . The new solution is better than the initial one and has a value  $z^* = 27718$ , optimal for the problem (Figure 6(d)).

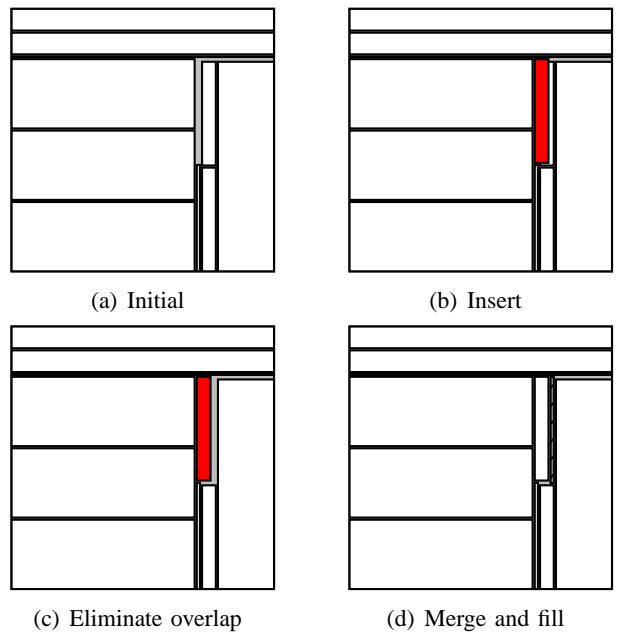


Fig. 6. Block insertion. Instance 1 from Fekete and Schepers [8]

### B. Moves to be studied

At each iteration we study all the possible reduction and insertion moves which can be applied to the current solution.

- Reduction:

- 1) Take each block of the solution, one at a time, in random order.
- 2) Consider all possibilities of reduction in the directions adjacent to empty rectangles.

- Insertion:

- 1) Select a piece for which the number of copies in the solution,  $x_i$ , is lower than  $Q_i$ , one at time, in random order.
- 2) Consider all the possible blocks which can be built with this piece.
- 3) Consider all the alternatives for placing the block onto an empty rectangle.

### C. Selection of the move

The objective function consists only of maximizing the value of pieces cut  $f(x) = \sum_i v_i x_i$ . However, if the moves are evaluated according to that function, there can be many moves with the same evaluation. Therefore, if there are ties, we break them by using a secondary objective function  $g(x) = k_1 S + k_2 |\mathcal{L}| + k_3 C + k_4 F$ .

- $S$  (*Symmetry*): We try not to explore symmetric solutions but prefer solutions in which empty rectangles are mostly concentrated to the right and to the top of the stock rectangle.

$S = 1$  if there is no symmetric solution with the wastes more concentrated to the right and to the top. Otherwise,  $S = 0$ .

- $|\mathcal{L}|$  (*Number of empty rectangles*). If possible, we prefer solutions in which the number of empty rectangles will be as low as possible.

- $C$  (*Centered and concentrated wastes*): We prefer solutions in which empty rectangles are centered and concentrated as much as possible, because that will make it easier to merge them and obtain spaces for more pieces. We consider the smallest rectangle  $ER$  containing all the empty rectangles and  $C = 1 - (0.75 * rd + 0.25 * ra)$

where  $rd$  is the distance from the center of  $ER$  to the center of the stock rectangle, divided by the distance from the center of the stock rectangle to its bottom left corner, and  $ra$  is the area of  $ER$  divided by the area of the stock rectangle.

- $F$  (*Feasibility*). In doubly constrained problems, the initial solution may not be feasible. In this case  $F = 1$ . Otherwise,  $F = 0$ .

These criteria are added to the secondary function with some weights reflecting their relative importance, according to the results of a preliminary computational experience on a subset of problems. In the current implementation, the weights are:

Criterion	Coefficient	Weight
Symmetry	$k_1$	5000
Empty rectangles	$k_2$	-950
Centered empty rectangles	$k_3$	50
Feasibility	$k_4$	-50000

### D. Tabu list

The tabu list contains for each solution the following pair of attributes: the value of the objective function and the smallest rectangle  $ER$  containing all its empty rectangles. A move is then tabu if these two attributes of the new solution match one pair of the tabu list.

The tabu list size varies dynamically. After a given number of iterations without improving the solution, the length is randomly chosen from  $[0.25 * M, 0.75 * M]$ , where  $M = \sum_i Q_i$ .

The *aspiration criterion* allows us to move to a tabu solution if it improves the best solution obtained so far.

### E. Intensification and diversification strategies

The moves we have defined involve a high level of diversification. However, we have included two more diversification strategies:

- Long term memory

Throughout the search process, we keep in memory the frequency of each piece appearing in the solutions.

This information is used for both diversification and intensification purposes. When used for diversification, we favor the moves of pieces not appearing very frequently in the solutions, then inducing new pieces to appear. When used for intensification, we consider only pieces corresponding to high quality solutions and then we favor these pieces appearing again in the new solutions.

In a diversification phase, the objective function is modified by subtracting a term that is the sum of the frequencies of the pieces appearing in the solutions:  $f(x) \rightarrow f(x) - \sum_i freq(p_i)$

In an intensification phase, the objective function is modified by adding a term reflecting the frequency of the pieces in the set of elite solutions  $\mathcal{E}$

$$f(x) \rightarrow f(x) + K \sum_{i \in \mathcal{E}} freq(p_i)$$

- Restarting

According to the secondary objective function, we tend to explore solutions satisfying the symmetry

criterion. After a given number of iterations without improving the best known solution, the current solution is changed by performing a horizontal and a vertical symmetry on it. The new solution obtained in that way would be quite different from the recently studied solutions and it is taken as a new starting point for the search.

#### F. Adjusting the bounds of the pieces

Throughout the iterative process we have the best known solution of value  $v_{best}$ . We can use this value to adjust  $P_i$  of some pieces that must appear if we want to improve the solution, and  $Q_j$  of some pieces whose inclusion would not allow us to improve the solution.

- *Increasing lower bounds  $P_i$*

Let us define  $total_{pieces} = \sum_i^m v_i * Q_i$ , the total value of the available pieces. If there is a piece  $i$  such that  $P_i < Q_i$ , and  $total_{pieces} - (Q_i - P_i) * v_i <= v_{best}$ , a solution with the minimum  $P_i$  copies of this type of piece cannot improve the best known solution. Any better solution must include more pieces of this type and  $P_i$  can be increased. If we compute  $t$  as:

$$\max t : total_{pieces} - t * v_i > v_{best}; \quad t \geq 0, \quad t \leq Q_i - P_i$$

Then,  $P_i = Q_i - t$ . This improved lower bound can be useful in the constructive phase, in which the pieces with  $P_i > 0$  are cut first, and in the improvement phase, in which pieces in their lower bounds are not considered to be removed from the current solution.

- *Decreasing upper bounds  $Q_i$*

Let us denote by  $R = \sum_{P_i > 0} P_i * l_i * w_i$  the area of the pieces which must appear in any feasible solution,  $R_v = \sum_{P_i > 0} P_i * v_i$ , the value of these pieces,  $e_i = v_i / (l_i * w_i)$  the efficiency of piece  $i$  and  $e_{max} = \max\{e_i, i = 1, \dots, m\}$ , the maximum efficiency of the pieces. If there is a piece  $i$ , with  $Q_i > P_i$  and  $e_i < e_{max}$  satisfying:

$$Q_i * l_i * w_i * (e_{max} - e_i) \geq e_{max} * (L * W - R) + R_v - v_{best}$$

any solution with  $Q_i$  copies of this piece cannot improve the best known solution. Therefore, at any better solution the number of copies of piece  $i$  should be limited to below  $Q_i$ . If we compute  $t$  as:

$$\begin{aligned} \max t : \quad t * l_i * w_i * (e_{max} - e_i) &< \\ e_{max} * (L * W - R) + R_v - v_{best} & \\ t \geq 0, \quad t \leq Q_i - P_i & \end{aligned}$$

then  $Q_i = P_i + t$ . This decrease in the upper bound can be useful when constructing and improving solutions in the subsequent iterations. In some cases,  $Q_i$  can be set to 0 and the corresponding piece is no longer considered for cutting.

#### G. Path relinking

Path Relinking is an approach for integrating intensification and diversification strategies in the context of Tabu Search (Glover and Laguna [9]). This approach generates new solutions by exploring trajectories that connect high quality solutions. Starting from one of these solutions, called the initiating solution, a path is generated in the solution space that leads towards another solution, called the guiding solution. This is done by selecting moves that introduce the attributes of the guiding solution into the new solutions.

Throughout the search process we keep a set of elite solutions, the best solutions found. We now take pairs of elite solutions and use one of them, solution A, as the initiating solution and the other, solution B, as the guiding solution. As the Tabu search algorithm favors solutions with empty rectangles preferably in the upper right part of the stock rectangle, both solutions will tend to have the empty rectangles in this zone. Therefore, we apply a vertical symmetry to the initiating solution before starting the Path Relinking process.

We follow a constructive strategy, inserting the blocks of solution B, one at a time, into solution A. The insertion move follows the procedure described in Section 3.1. The pieces overlapping with the block are eliminated, the remaining blocks are moved to their nearest corners and the resulting empty rectangles are filled. At the end of the process, we will have reproduced solution B, but along the path new solutions will have been generated. When inserting the blocks of solution B, we first insert the blocks adjacent to the sides of the stock rectangle and then the blocks in the center.

An example of the first step of Path Relinking appears in Figure 7. Solution A with value 5376 is selected as the initiating solution (Figure 7(a)), and solution B with value 5352 as the guiding solution (Figure 7(b)). The vertical symmetry applied to solution A produces the solution of Figure 7(c). Then a block of B is inserted on it (Figure 7(d)). The pieces overlapping with it are deleted as well as one piece of the same type of the piece being inserted so that  $Q_i$  is not exceeded. The remaining blocks are moved to their nearest corners, as indicated by the arrows in Figure 7(e). Finally the empty spaces are merged and filled with new pieces, producing the solution in Figure 7(f). This solution is different from solutions A and B and has a value of 5395.

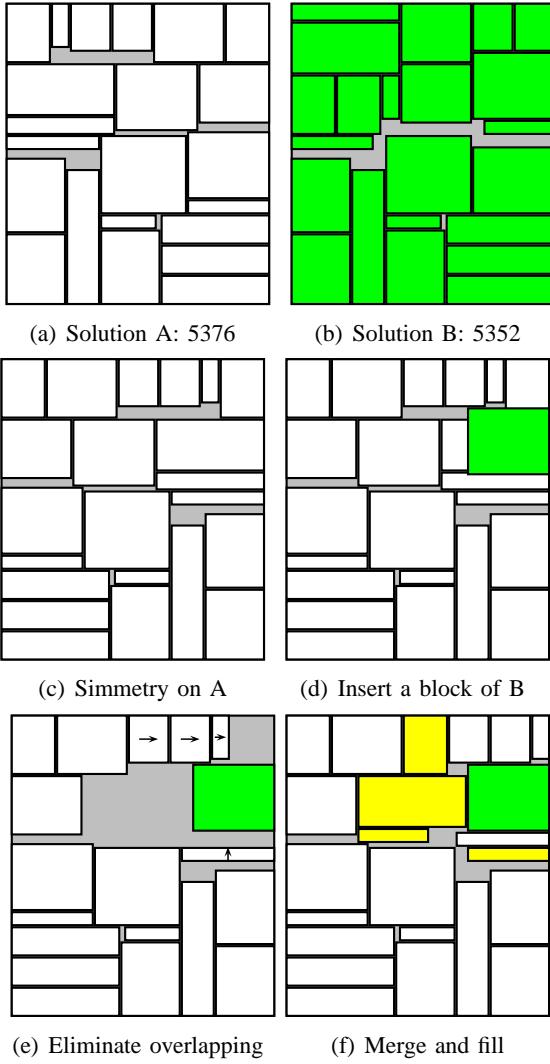


Fig. 7. Path Relinking. Instance 2 from Jakobs [13]

#### IV. COMPUTATIONAL EXPERIENCE

##### A. Test problems

We have used several sets of test problems:

- 1) A set of 21 problems from the literature: 15 from Beasley [4], 2 from Hadjiconstantinou and Christofides [10], 1 from Wang [21], 1 from Christofides and Whitlock [7], 5 from Fekete and Schepers [8]. For all of them the optimal solutions are known. They have also been solved by Beasley [5].
- 2) A set of 630 large problems generated by Beasley [5], following the work by Fekete and Schepers [8]. All the problems have a stock rectangle of size (100, 100). For each value of  $m$ , number of piece types ( $m = 40, 50, 100, 150, 250, 500, 1000$ ), 10 problems are randomly generated with  $P_i = 0$ ,  $Q_i = Q^*, \forall i = 1, \dots, m$  where  $Q^* = 1; 3; 4$ . These 630 instances are divided into 3 types,

according to the percentages of the types of pieces of each class:

Class	Description	Length	Width	
1	Short and wide	[1,50]	[75,100]	
2	Long and narrow	[75,100]	[1,50]	
3	Large	[50,100]	[50,100]	
4	Small	[1,50]	[1,50]	
Type	Percentages of pieces of each class			
	1	2	3	4
1	20	20	20	40
2	15	15	15	55
3	10	10	10	70

The value assigned to each piece is equal to its area multiplied by an integer randomly chosen from  $\{1, 2, 3\}$ .

- 3) The 21 test problems mentioned first were transformed by Beasley [5] into doubly constrained problems by defining some lower bounds  $P_i$ . Specifically, for each type of piece from  $i = 1, \dots, m$  satisfying:

$$\sum_{j=1, j \neq i}^m (l_j w_j) P_j + l_i w_i \leq (LW)/3, \text{ the lower bound } P_i \text{ is set to 1.}$$

This set of problems would allow us to test the algorithm in the general case of doubly constrained problems.

- 4) Finally, we have included the test problems used by Leung et al. [17], consisting of 3 instances from Lai and Chan [14], 5 from Jakobs [13], and 2 from Leung et al. [17]. We have also included 21 larger instances from Hopper and Turton [12]. There are unweighted problems in which the value of each piece corresponds to its area and the objective is to minimize the waste of the stock rectangle. The problems have been generated in such a way that the optimal solution is a cutting pattern with zero waste.

We have included the Leung et al. [17] and Hopper and Turton [12] sets of problems because they have characteristics which can be considered complementary to the two first sets used by Beasley, as can be seen in Table I, in which we show the ratios of total pieces available to be cut to the upper bound of pieces fitting into the stock rectangles. We can see that the problems of the second set, Types I, II and III, can be considered *selection* problems because there are many available pieces and only a small fraction of them will make part of the solution. However, the Leung et al. and Hopper and Turton problems are *jigsaw* problems. All available pieces will make part of the solution and the difficulty here is to find their correct position in the cutting pattern.

An algorithm working well on both types of problems can be considered a general purpose algorithm.

Sets of problems	Averages	
	Total value of pieces/ Upper bound of value	Total area of pieces/ Upper bound of area
Literature problems	3,13	3,61
Type I	123,69	185,60
Type II	101,69	152,71
Type III	79,67	119,20
Zero-waste problems	1,00	1,00

TABLE I  
*Test problems – Characteristics.*

### B. Implementation

Our algorithm has been coded in *C++* and run on a *PentiumIII* at 800 *Mhz*. After 100 iterations without improving the best known solution, the length of the tabu list is changed. After 400 iterations without improvement we do a diversification phase based on long term frequencies over 100 iterations or until an improved solution is found. We then recover the original objective function and continue the search. After 400 iterations without improvement we do an intensification phase with  $K = 100$  over 100 iterations or until an improved solution is found. We recover the original objective function but if the solution has not been improved, instead of continuing the search from it we do a restarting move and proceed from the new solution.

In Section 4.1 we have described three types of problems: constrained selection problems, constrained jigsaw problems and doubly constrained problems. Some of the strategies described in Section 3 are more adequate for some of these types of problems. In Step 2 of the Block insertion move, if we have a selection problem, we use the second strategy, identifying the empty rectangle with the largest common area and only studying the possible positions of the new block on it. If we have a jigsaw problem, we study all the empty rectangles as possible placements for the block. The algorithm automatically detects the type of problem and applies the adequate strategy.

The algorithm runs until it reaches the optimal solution, if known, or the corresponding upper bound, or until a limit of 1500 iterations. The strategy of stopping at the optimal solution or the upper bound has been used by Beasley [5] and we have adopted it in order to compare our results with those obtained by him. The limit of 1500 has been set to keep the running times similar to those of the GRASP algorithm used by Alvarez-Valdes et al. [1].

Throughout the search we keep a set of 10 elite solutions upon which the Path Relinking procedure will act. However, as the solutions provided by Tabu Search were so good, Path Relinking could not improve the initial solutions and this procedure has not been included in the final implementation.

### C. Computational results

The computational results appear in Tables II, III and IV. The first two tables include a direct comparison with Beasley's results [5] and with the GRASP algorithm results [1] in terms of solution quality. The computing times cannot be directly compared with Beasley's time. Beasley coded his algorithm in FORTRAN and used a Silicon Graphics O2 workstation (R10000 chip, 225MHz, 128 MB). An approximate comparison (<http://www.spec.org>) indicates that his computer is twice as fast as ours. On Table II we see that our Tabu Search algorithm optimally solves all the problems in very short computing times, clearly outperforming the other two algorithms in terms of quality and running times. For the large problems in Table III the optimal solutions are not known and the comparisons are made with knapsack upper bounds. Table III shows that the Tabu Search algorithm again obtains better results on every type of problem, except for  $m = 50$ ,  $Q^* = 1$  in which GRASP is slightly better. The computing times are much shorter than those of Beasley's algorithm, though they are larger than those required by the GRASP procedure. Both algorithms are based on similar ideas. The more elaborated Tabu Search algorithm obtains better solutions but needs longer processing times.

The adjustment of lower bounds does not have significant effects on the performance of the algorithms, but the adjustment of upper bounds does have a dramatic effect, especially in these large random problems in which there are important differences in the efficiency of the pieces. For instance, for problems with  $m = 1000$  types of pieces, more than 60% of the pieces are discarded as soon as good solutions are found. That increases significantly the speed of GRASP and Tabu Search algorithms which use these adjustments.

The direct comparison with Leung et al. [17] is not possible, though their 19 test instances are a subset of those appearing in Table IV. On the one hand, they do not give CPU times. On the other hand, they propose two versions of their algorithm, each of them with several mutation rates, and give minimum and mean waste in 15 runs of 30000 iterations. The best that can be said is that our average distance to optimum is slightly better than the average distances of their algorithms and quite

similar to the best distances they obtain after 15 runs. We can also point out, as Beasley [5] illustrates, that there are some optimal cutting patterns that cannot be obtained by the Leung et al. [17] procedure, a situation that does not arise with our procedure. In Table IV we again compare the GRASP and the Tabu Search algorithms. Tabu Search clearly outperforms GRASP in terms of the quality of the solution, with quite similar computing times.

Finally, Table V shows the results of the algorithms on the set of doubly constrained test problems. The upper bound corresponds to the solution of the constrained problem. The problems for which the algorithms do not find solutions are not feasible, but they are maintained in the set of test problems and therefore are included in the table. The Tabu Search algorithm obtains the best result for each instance of the set but its computing times are slightly longer.

## V. CONCLUSIONS

We have developed a new heuristic algorithm based on Tabu Search techniques for the non-guillotine two-dimensional cutting stock problem. Two moves have been proposed, based on the reduction and insertion of blocks of pieces. The efficiency of the moves is based on a *merge and fill* strategy that accommodates the empty rectangles to the pieces still to be cut. Some intensification and diversification strategies, based on long-term memory, have also been included.

The computational results show that these ideas work well for the constrained and doubly constrained test problems proposed by Beasley [5]. For the Leung et al. [17] and Hopper and Turton [12] zero-waste problems the results are also good and the proposed algorithm can be considered to work consistently well for a wide range of cutting problems.

## ACKNOWLEDGEMENTS

This work has been partially supported by Project PBC-02-002, Consejería de Ciencia y Tecnología, JCCM, the Spanish Ministry of Science and Technology DPI2002-02553, and the Valencian Science and Technology Agency, GRUPOS03/174.

## REFERENCES

- [1] R. Alvarez-Valdes, F. Parreño, J.M. Tamarit, A GRASP algorithm for constrained two-dimensional non-guillotine cutting problems, Journal of Operational Research Society (2004), in press.
- [2] A. Amaral, A. Letchford, An improved upper bound for the two-dimensional non-guillotine cutting problem, Working paper available from the second author at Department of Management Science, Management School, Lancaster University, Lancaster LA1 4YW, England, 2003.
- [3] M. Arenales, R. Morabito, An AND/OR-graph approach to the solution of two-dimensional non-guillotine cutting problems, European Journal of Operational Research 84 (1995) 599-617.
- [4] J.E. Beasley, An exact two-dimensional non-guillotine cutting tree search procedure, Operations Research 33 (1985) 49-64.
- [5] J.E. Beasley, A population heuristic for constrained two-dimensional non-guillotine cutting, European Journal of Operational Research 156 (2004) 601-627.
- [6] A. Caprara, M. Monaci, On the two-dimensional Knapsack problem, Operations Research Letters 32 (2004) 5-14.
- [7] N. Christofides, C. Whitlock, An algorithm for two-dimensional cutting problems, Operations Research 25 (1977) 30-44.
- [8] S.P. Fekete, J. Schepers, On more-dimensional packing III: Exact Algorithms, Report 97290 available from the first author at Department of Mathematics, Technical University of Berlin, Germany (1997), revised (2000).
- [9] F. Glover, M. Laguna, Tabu Search, Kluwer Academic Publishers, Boston, 1997.
- [10] E. Hadjiconstantinou, N. Christofides, An exact algorithm for general, orthogonal, two-dimensional knapsack problems, European Journal of Operational Research 83 (1995) 39-56.
- [11] P. Healy, M. Creavin, A. Kuusik, An optimal algorithm for placement rectangle, Operations Research Letters 24 (1999) 73-80.
- [12] E. Hopper, B.C.H. Turton, An empirical investigation of meta-heuristic and heuristic algorithms for a 2D packing problem, European Journal of Operational Research 128 (2001) 34-57.
- [13] S. Jakobs, On genetic algorithms for the packing of polygons, European Journal of Operational Research 88 (1996) 165-181.
- [14] K.K. Lai, J.W.M. Chan, Developing a simulated annealing algorithm for the cutting stock problem, Computers and Industrial Engineering 32 (1997) 115-127.
- [15] K.K. Lai, J.W.M. Chan, A evolutionary algorithm for the rectangular cutting stock problem, International Journal of Industrial Engineering 4 (1997) 130-139.
- [16] T.W. Leung, C.H. Yung, M.D. Troutt, Applications of genetic search and simulated annealing to the two-dimensional non-guillotine cutting stock problem, Computers and Industrial Engineering 40 (2001) 201-214.
- [17] T.W. Leung, C.H. Yung, M.D. Troutt, Application of a mixed simulated annealing-genetic algorithm heuristic for the two-dimensional orthogonal packing problem, European Journal of Operational Research 145 (2003) 530-542.
- [18] G. Scheithauer, J. Terno, Modeling of packing problems, Optimization 28 (1993) 63-84.
- [19] G. Scheithauer, LP-based bounds for the Container and Multi-Container Loading Problem, International Transactions in Operations Research 6 (1999) 199-213.
- [20] R.D. Tsai, E.M. Malstrom, H.D. Meeks, A two-dimensional palletizing procedure for warehouse loading operations, IIE Transactions 20 (1988) 418-425.
- [21] P.Y. Wang, Two algorithms for constrained two-dimensional cutting stock problems, Operations Research 31 (1983) 573-586.
- [22] Y.L. Wu, W. Huang, S.C. Lau, C.K. Wong, G.H. Young, An effective quasi-human based heuristic for solving rectangle packing problem, European Journal of Operational Research 141 (2002) 341-358.

Source of problem	I	Problem size (L,W)		Beasley's solution	GRASP solution	TABU solution	Optimal solution	CPU time (seconds)		
		m	M					Beasley	GRASP	TABU
Beasley [4]	1	(10, 10)	5	10	164	164	164	0,02	0,00	0,06
	2	(10, 10)	7	17	230	230	230	0,16	0,00	0,00
	3	(10, 10)	10	21	247	247	247	0,53	0,00	0,00
	4	(15, 10)	5	7	268	268	268	0,01	0,00	0,00
	5	(15, 10)	7	14	358	358	358	0,11	0,00	0,00
	6	(15, 10)	10	15	289	289	289	0,43	0,00	0,00
	7	(20, 20)	5	8	430	430	430	0,01	0,00	0,00
	8	(20, 20)	7	13	834	834	834	3,25	0,77	0,16
	9	(20, 20)	10	18	924	924	924	2,18	0,00	0,05
	10	(30, 30)	5	13	1452	1452	1452	0,03	0,00	0,00
	11	(30, 30)	7	15	1688	1688	1688	0,60	0,05	0,00
	12	(30, 30)	10	22	1801	1865	1865	3,48	0,05	0,06
Hadjiconstantinou and Christofides [10]	3	(30, 30)	7	7	1178	1178	1178	0,03	0,00	0,00
	11	(30, 30)	15	15	1270	1270	1270	0,04	0,00	0,00
Wang [21]		(70, 40)	19	42	2721	2726	2726	6,86	0,77	0,11
		(40, 70)	20	62	1720	1860	1860	8,63	0,39	0,06
Christofides and Whitlock [7]	3	(100, 100)	15	50	27486	27589	27718	27718	19,71	2,31
	1	(100, 100)	30	30	21976	21976	22502	22502	13,19	4,17
	2	(100, 100)	30	30	23743	23743	24019	24019	11,46	3,68
	3	(100, 100)	33	61	31269	32893	32893	32893	32,08	0,00
	5	(100, 100)	29	97	26332	27923	27923	27923	83,44	0,00
Mean percentage of deviation from optimum				1,21%	0,19%	0,00%		8,87	0,58	0,32
Number of optimal solutions (out of 21)				13	18	21				

TABLE II  
*Computational results – Problems from literature*

Mean percentages of deviation from knapsack upper bound								
m	Q*	M	Beasley's solution	GRASP solution	TABU solution	CPU time (seconds)		
						Beasley	GRASP	TABU
40	1	40	7,77	6,97	6,55	13,57	2,33	10,97
	3	120	3,54	2,22	1,95	47,43	6,62	14,20
	4	160	3,24	1,81	1,65	63,30	4,44	18,26
50	1	50	5,48	4,80	4,85	14,60	4,71	15,49
	3	150	2,35	1,50	1,27	59,27	7,05	22,50
	4	200	2,63	1,18	0,96	80,07	5,34	18,19
100	1	100	2,26	1,51	1,50	27,20	5,36	38,79
	3	300	1,27	0,47	0,31	119,47	9,41	32,11
	4	400	1,06	0,26	0,18	175,10	6,99	19,67
150	1	150	1,31	0,89	0,84	40,60	5,53	54,90
	3	450	0,60	0,14	0,07	190,53	11,71	31,76
	4	600	0,92	0,11	0,05	323,83	6,75	19,87
250	1	250	0,88	0,51	0,45	76,70	5,27	90,07
	3	750	0,57	0,04	0,01	439,47	13,89	13,70
	4	1000	0,39	0,03	0,00	693,67	6,65	4,50
500	1	500	0,26	0,05	0,03	203,10	3,24	86,17
	3	1500	0,18	0,00	0,00	1210,80	12,24	1,10
	4	2000	0,18	0,00	0,00	1790,83	1,15	0,84
1000	1	1000	0,09	0,00	0,00	667,23	1,01	7,80
	3	3000	0,07	0,00	0,00	3318,47	6,53	1,54
	4	4000	0,07	0,00	0,00	4840,57	0,29	1,19
Type I			1,64	1,04	0,95	558,11	5,13	19,61
Type 2			1,70	1,14	1,06	668,41	5,90	23,84
Type 3			1,66	1,03	0,94	830,02	7,28	32,56
All			1,67	1,07	0,98	685,51	5,91	25,34

TABLE III  
*Computational results – Large random problems.*

Source of problem	I	Problem size (L,W)		m	M	GRASP solution	TABU solution	Optimal solution	CPU time GRASP	CPU time TABU
Lai and Chan [14]	1	(400,200)		9	10	80000	80000	80000	0,00	0,00
	2	(400,200)		7	15	79000	79000	79000	0,00	0,02
	3	(400,400)		5	20	154600	160000	160000	4,12	0,38
Jakobs [13]	1	(70,80)		14	20	5447	5600	5600	10,16	1,89
	2	(70,80)		16	25	5455	5540	5600	15,44	16,88
	3	(120,45)		22	25	5328	5400	5400	12,57	0,42
	4	(90,45)		16	30	3978	4050	4050	10,28	1,97
	5	(65,45)		18	30	2871	2925	2925	14,94	1,53
Leung et al. [17]	1	(150,110)		40	40	15856	16280	16500	90,52	52,36
	2	(160,120)		50	50	18628	19044	19200	132,26	63,95
Hopper and Turton [12]	1-1	(20,20)		16	16	400	400	400	0,94	0,42
	1-2	(20,20)		17	17	386	400	400	9,28	4,23
	1-3	(20,20)		16	16	400	400	400	0,06	0,95
	2-1	(40,15)		25	25	590	600	600	19,44	0,44
	2-2	(40,15)		25	25	597	600	600	17,36	4,16
	2-3	(40,15)		25	25	600	600	600	0,71	0,00
	3-1	(60,30)		28	28	1765	1800	1800	26,80	4,91
	3-2	(60,30)		29	29	1755	1800	1800	37,35	10,11
	3-3	(60,30)		28	28	1774	1800	1800	30,92	5,52
	4-1	(60,60)		49	49	3528	3580	3600	102,05	45,27
	4-2	(60,60)		49	49	3524	3564	3600	110,79	68,59
	4-3	(60,60)		49	49	3544	3580	3600	94,41	51,11
	5-1	(60,90)		73	73	5308	5342	5400	212,07	135,97
	5-2	(60,90)		73	73	5313	5361	5400	231,56	96,80
	5-3	(60,90)		73	73	5312	5375	5400	231,24	82,06
	6-1	(80,120)		97	97	9470	9548	9600	480,44	240,39
	6-2	(80,120)		97	97	9453	9448	9600	465,49	399,86
	6-3	(80,120)		97	97	9450	9565	9600	478,02	206,78
	7-1	(160,240)		196	196	37661	38026	38400	3760,14	3054,38
	7-2	(160,240)		197	197	37939	38145	38400	2841,96	1990,70
	7-3	(160,240)		196	196	37745	37867	38400	3700,99	5615,75
Mean percentage of deviation from optimum						1,68%	0,42%		423,95	392,19
Number of optimal solutions (out of 31)						5	16			

TABLE IV  
*Computational results – Zero-waste problems*

Source of problem	I	Problem size (L,W)		m	M	Beasley's solution	GRASP solution	TABU solution	Upper bound	CPU time (seconds) Beasley	CPU time (seconds) GRASP	CPU time (seconds) TABU
Beasley [4]	1	(10, 10)		5	10	164	164	164	164	0,02	0,00	0,00
	2	(10, 10)		7	17	225	225	225	230	5,53	0,71	1,70
	3	(10, 10)		10	21	220	220	220	247	7,85	1,21	2,26
	4	(15, 10)		5	7	268	268	268	268	0,01	0,00	0,00
	5	(15, 10)		7	14	301	301	301	358	5,05	0,72	1,48
	6	(15, 10)		10	15	265	252	265	289	6,81	1,81	1,59
	7	(20, 20)		5	8	430	430	430	430	0,01	0,00	0,00
	8	(20, 20)		7	13	819	819	819	834	6,54	1,32	1,76
	9	(20, 20)		10	18	924	924	924	924	5,64	0,00	0,00
	10	(30, 30)		5	13	n/f	n/f	n/f	n/f	2,38	0,22	0,94
	11	(30, 30)		7	15	1505	1518	1518	1688	2,96	1,59	2,52
	12	(30, 30)		10	22	1666	1648	1672	1865	3,78	1,65	3,73
Hadjiconstantinou and Christofides [10]	3	(30, 30)		7	7	1178	1178	1178	1178	0,25	0,00	0,00
	11	(30, 30)		15	15	1216	1216	1216	1270	2,60	2,08	3,18
	Wang [21]	(70, 40)		19	42	2499	2700	2716	2726	6,36	1,48	6,16
Christofides and Whitlock [7]	3	(40, 70)		20	62	1600	1720	1720	1860	6,81	0,88	5,27
	1	(100, 100)		15	50	25373	24869	25384	27718	11,86	3,73	25,27
	2	(100, 100)		30	30	17789	19083	19657	22502	5,80	3,02	18,35
	3	(100, 100)		30	30	n/f	n/f	n/f	n/f	4,03	0,66	12,41
	4	(100, 100)		33	61	27556	27898	28974	32893	20,42	2,80	37,46
Fekete and Schepers [8]	5	(100, 100)		29	97	21997	22011	22011	27923	18,41	3,30	61,90
	Mean percentage of deviation from upper bound						8,11%	7,36%	6,62%	5,86	1,29	8,86
n/f: No feasible solution found												

TABLE V  
*Computational results – Doubly constrained problems*

# Problem of time-consistency in model of Kyoto Protocol realization

Maria Dementieva\*, Pekka Neittaanmäki\* and Victor Zakharov†

\*University of Jyväskylä/Dept. of Mathematical Information Technology

P.O. Box 35, 40014 University of Jyväskylä, Finland

Email: madement@cc.jyu.fi

†St. Petersburg State University/Faculty of Applied Mathematics

Universitetskii prospekt 35, Peterhof, St. Petersburg, Russia 198504

Email: mcvictor@icape.nw.ru

**Abstract**—In this paper we consider a multistage cooperative model of the Kyoto Protocol realization. An important problem in dynamic cooperative games is time-consistency of a solution. Time-consistency provides the optimality of the solution at any moment of the process with respect to relevant initial states. Otherwise, the absence of time-consistency in the optimality principle involves the possibility that the previous “optimal” decision are abandoned at some current moment of time, thereby making meaningless the problem for seeking an optimal control. This is why particular emphasis is placed on the construction of time-consistent optimality principles. We use two different approaches to this problem and apply them in real-life cooperative game.

**Keywords**—Game theory, flexibility mechanisms, environment.

## I. INTRODUCTION

In this work we construct multistage cooperative model of the Kyoto Protocol realization and suggest time-consistent solutions to numerical examples with three country groups.

Without a doubt, climate change is the first among the global environmental threats to civilization at the beginning of the XXI Century. The importance of this problem is demonstrated by the adaptation costs the global community pays to protect itself from a growing number of natural disasters. The United Nations Framework Convention on Climate Change was signed at the World Summit on the Environment and Development in Rio de Janeiro in 1992, and the Kyoto Protocol to the Convention was adopted in 1997 [15]. The Kyoto Protocol proposes six innovative “mechanisms:” joint implementation, clean development, emission trading, joint fulfilment, banking and sinks. The mechanisms aim to reduce the costs of curbing emissions by allowing Parties (Party is a term of Kyoto Protocol and means

a country, or group of countries, that has ratified the Kyoto Protocol) to pursue opportunities to cut emissions more cheaply abroad than at home. The cost of curbing emissions varies considerably from region as a result of differences in, for example, energy sources, energy efficiency and waste management. It makes economic sense to cut emissions where it is cheapest to do so, given that the impact on the atmosphere is the same.

The Kyoto protocol defines six flexibility mechanisms and three of them have the following sense: “joint implementation” provides for Annex B Parties (mostly highly developed industry countries) to implement projects that reduce emission, or remove carbon from the air, in other Annex B Parties, in return for emission reduction units (ERUs); the “clean development” mechanism provides for Annex B Parties to implement projects that reduce emissions in non-Annex B Parties, in return for certified emission reductions (CERs), and assist the host Parties in achieving sustainable development and contributing to the ultimate objective of the Convention; “emission trading” provides for Annex B Parties to acquire units from other Annex B Parties. The emission reduction units and certified emission reductions generated by the flexibility mechanisms can be used by Annex B Parties to help meet their emission targets.

That flexibility mechanisms are the base of the cooperation because joint implementation, clean development, and emission trading comprehend that Parties work together and receive common “benefit” (emission reduction), which should be allocated fairly. It is natural to use the dynamic cooperative theory to model the Kyoto Protocol realization [3]. For other models connected with the flexibility mechanisms of the Kyoto Protocol see [1], [5], [6], [10], [12], [13].

## II. PRELIMINARIES

In this section we give a preliminary information and basic definitions.

Let us denote a division of time period  $t_0 < t_1 < \dots < t_m$  by  $\mathbb{T} = \{t_r\}_{r=0}^m$ ,  $m \in \mathbb{N}$ . We call a pair  $(N, v)$  a multistage cooperative game. Here  $N$  is a finite set of players and  $v : 2^N \times \mathbb{T} \mapsto \mathbb{R}$  is a characteristic function of the game,  $v(\emptyset, t) = 0$  for all  $t \in \mathbb{T}$ ,  $v(S, t_m) = 0$  for all  $S \subset N$ . The sign  $(N, v(t^*))$  means the subgame at a moment  $t^* \in \mathbb{T}$ . We assume that  $v(N, t)$  is the decreasing function with respect to  $t$ .

A vector  $\xi = (\xi_1, \dots, \xi_n)$  is called an imputation in a cooperative game  $(N, v(t))$  if its components satisfy the following conditions

$$1) \quad \xi_i \geq v(i, t), \quad \forall i \in N, \quad (1)$$

$$2) \quad \sum_{i \in N} \xi_i = v(N, t). \quad (2)$$

A subset  $C(N, v(t))$  of imputation set is called the core of cooperative game  $(N, v(t))$  if all its elements satisfy inequalities

$$\sum_{i \in S} \xi_i \geq v(S, t), \quad \forall S \subset N. \quad (3)$$

The set  $X^0(t)$  is the solution set of the following linear programming problem

$$\text{minimize } \sum_{i \in N} \xi_i, \quad (4)$$

$$\text{subject to } \sum_{i \in S} \xi_i \geq v(S, t), \quad S \subset N, \quad S \neq N. \quad (5)$$

Let  $Y^0(t)$  be the union

$$Y^0(t) = \bigcup_{\xi^0 \in X^0(t)} Con(\xi^0) = \bigcup_{\xi^0 \in X^0(t)} \{\xi | \xi \geq \xi^0\}.$$

Now let us redefine some solution concepts for a multistage cooperative game  $(N, v)$  using  $Y^0(t)$ . We call a set

$$SC(v(t), \xi^0(t)) = \{\xi \in Con(\xi^0(t)) | \sum_{i \in N} \xi_i = v(N, t)\}$$

subcore of the game  $(N, v(t))$  with respect to  $\xi^0(t)$  from  $X^0(t)$ ; the set

$$\begin{aligned} GSC(N, v(t)) &= \bigcup_{\xi^0(t) \in X^0(t)} SC(v(t), \xi^0(t)) \\ &= \{\xi \in Y^0(t) | \sum_{i \in N} \xi_i = v(N, t)\} \end{aligned}$$

is called grand subcore of the game  $(N, v(t))$ ; and the set

$$\begin{aligned} TCGSC(N, v(t_k)) \\ &= \{\xi \in \bigcap_{r=k}^m Y^0(t_r) | \sum_{i \in N} \xi_i = v(N, t_k)\} \end{aligned}$$

is time-consistent grand subcore of the game  $(N, v(t_k))$ ,  $t_k \in \mathbb{T}$ . Clearly, the grand subcore is a subset of the core in the balanced TU-games. Moreover, the subcore and the grand subcore are non-empty if and only if the game is balanced [18].

To formalize the notion of time-consistency for cooperative games let us introduce the following definitions [11].

**Definition 2.1:** The solution concept  $\phi(t)$  of a multistage cooperative game  $(N, v)$  is called time-consistent if for every  $\xi \in \phi(t)$  and for all  $t^* \in \mathbb{T}$  there exists a vector  $\alpha(t^*) \geq 0$ , such that  $\xi - \alpha(t^*) \in \phi(t^*)$ .

**Definition 2.2:** Suppose that  $\xi = (\xi_1, \dots, \xi_n) \in \phi(t_0)$ . Any matrix  $\alpha = \{\alpha_{ik}\}$ ,  $i = 1, \dots, n$ ,  $k = 0, \dots, l$ , such that

$$\xi_i = \sum_{k=0}^l \alpha_{ik}, \quad \alpha_{ik} \geq 0,$$

is called imputation distribution procedure (IDP).

The necessary and sufficient condition for time-consistency of an imputation from the grand subcore is the following [19].

**Theorem 2.3:** In a balanced multistage game  $(N, v)$ ,  $t \in \mathbb{T}$ , a vector  $\xi(t_0) \in GSC(N, v(t_0))$  is time-consistent if and only if  $\xi(t_0) \in Y^0(t)$  for all  $t \in \mathbb{T}$ .

Let us consider now an algorithm based on the maximization of total payoffs at every step. Assume that the Theorem 2.3 is fulfilled for a multistage cooperative game  $(N, v)$ .

### A. Algorithm

Let  $\alpha(t_k)$  be a total payoff vector at a period  $(t_k, t_m]$ ,  $\alpha(t_{k-1}, t_k)$  be a payoff vector at a moment  $t_k$ , and  $\tilde{v}(N, t_k)$  be a new guaranteed gain at a period  $[t_k, t_m]$ .

We define the set  $Z^0(t_k)$  as a solution set of the following minimization problem

$$\text{minimize } \sum_{i \in N} \omega_i, \quad (6)$$

$$\text{subject to } \omega \in \bigcap_{r=k}^m Y^0(t_r). \quad (7)$$

Remark that for all  $t_k \in \mathbb{T}$  there exists a solution of the problem (6), (7). It ensues by construction of  $Y^0(t_k)$ .

*Initial step.* We choose a vector  $\xi(t_0) \in TCGSC(N, v(t_0))$ . The players will receive this imputation by the end of the game (the moment  $t_m$ ). We set

$$\alpha(t_0) := \xi(t_0), \quad \tilde{v}(N, t_0) := v(N, t_0).$$

*Step number k.* At this step we find a non-negative payoff vector to the players at the moment  $t_k$  with

respect to the vector  $\alpha(t_{k-1})$  and  $\tilde{v}(N, t_{k-1})$  from the previous step.

Consider a set  $Z^0(t_k)$ . If for a vector  $\omega \in Z^0(t_k)$  we have  $\sum_{i \in N} \omega_i \leq v(N, t_k)$ , then we set

$$\alpha(t_k) := \xi(t_k), \text{ and } \tilde{v}(N, t_k) := v(N, t_k).$$

Here  $\xi(t_k) \in TCGSC(N, v(t_k))$  and  $\xi(t_k) \leq \xi(t_{k-1})$ . Otherwise we set

$$\alpha(t_k) := \omega, \text{ and } \tilde{v}(N, t_k) := \sum_{i \in N} \omega_i.$$

Here  $\omega$  is a vector from  $Z^0(t_k)$ . Finally, we set

$$\alpha(t_{k-1}, t_k) := \alpha(t_{k-1}) - \alpha(t_k).$$

*The last step.* By the definition of a multistage game we have

$$\alpha(t_m) := (0, 0), \tilde{v}(N, t_m) := 0,$$

and

$$\alpha(t_{m-1}, t_m) = \alpha(t_{m-1}).$$

The existence of the feasible  $\alpha(t_{k+1})$  with respect to  $\alpha(t_k)$  is proved in [19]. As the result of the algorithm we have the non-negative payoff sequence  $\{\alpha(t_{k-1}, t_k)\}_{k=1}^m$  (i.e. the imputation distribution procedure), which guarantees the time-consistent solution  $\xi(t_0)$ .

It should also be stated that this method works if there is at least a time-consistent solution and it helps to construct IDP providing this solution. In the following subsection we consider the regularization of a balanced multistage cooperative game without time-consistent imputations in the grand subcore.

### B. MDM-reduced game

In [4] the method of minimal reduction is introduced to provide time-consistent solution. The distinctive of this regularization method is the use of reduced games to change the player set at the moment when the property of time-consistency is broken. That is at such a moment  $t^* \in \mathbb{T}$  the corresponding imputation  $\xi(t^*)$  does not belong to the grand subcore  $GSC(N, v(t^*))$ . We find the minimal coalition  $K \subset N$  such that  $\xi_{N \setminus K}(t^*) \in GSC(N \setminus K, v_{\xi^0}^K(\cdot))$ . Here  $(N \setminus K, v_{\xi^0}^K(\cdot))$  is a modification of the Davis–Maschler reduced game [2]. In a multistage cooperative game  $(N, v)$ ,  $t \in \mathbb{T}$ , for a given removing coalition  $K \subset N$ ,  $\xi^0(t) \in X^0(N, v(t))$  and a payoff vector  $\xi(t)$  the characteristic function of the MDM-reduced game is the following

$$v_{\xi^0}^K(\cdot) = v_{\xi^0}^K(S, \xi_K(t))$$

$$= \begin{cases} 0, & \text{if } S = \emptyset, \\ v(N, t) - \sum_{j \in K} \xi_j(t), & \text{if } S = N \setminus K, \\ \max_{R \subseteq K} \{v(S \cup R, t) - \sum_{j \in R} \xi_j^0(t)\}, & \text{otherwise.} \end{cases}$$

In particular, the minimal coalition depends on the vector  $\xi^0(t^*)$ . For details see [4].

### III. KYOTO PROTOCOL MODEL

In this section we describe a cooperative model of relations of countries (or groups of countries) under Kyoto Protocol. The players pursue two main goals: to achieve the required amount of emission reduction units and to decrease the reduction costs. The participants of the corresponding projects can get significant income from realization of the flexibility mechanisms. To define the cooperative model we should set a method of calculation the characteristic function  $v$  of the game. We assume that  $v(S)$ , where  $S$  is a coalition of players, is the difference between the sum of the personal costs of players, when they act individually, and total cost of coalition  $S$  under co-operation. Here player is Party in Kyoto Protocol. In the model we use the following notations

$K_i$	— emission quota of player $i$ ;
$c_i^e$	— price of emission unit for player $i$ ;
$c_i^q$	— price of emission unit on account of a pollution quota of player $i$ ;
$\Delta E_i$	— required emission reduction of player $i$ ;
$\Delta L_i$	— ecological sinks <sup>1</sup> of player $i$ ;
$\Delta K_i$	— a fraction of pollution quota that player $i$ wants to use.

Let us consider a game with two players. The individual cost of player  $i$  is

$$H_i^0 = c_i^e(\Delta E_i - \Delta L_i - \Delta K_i). \quad (8)$$

Under co-operation a more developed country (player **1**) can invest money into the emission reduction in the territory of another country (player **2**). That is,  $c_1^e > c_2^e$  and  $c_1^q > c_2^q$ . Let us assume that the reduction costs are

$$H_1 = c_2^e \cdot \delta_1 (\Delta E_1 - \Delta L_1 - \Delta K_1) + c_2^q \cdot \delta_2 (\Delta E_1 - \Delta L_1 - \Delta K_1) + c_1^e (1 - \delta_1 - \delta_2) (\Delta E_1 - \Delta L_1 - \Delta K_1); \quad (9)$$

$$H_2 = c_2^e (\Delta E_2 - \Delta L_2 - \Delta K_2 + \delta_2 (\Delta E_1 - \Delta L_1 - \Delta K_1)) - c_2^e \cdot \delta_1 (\Delta E_1 - \Delta L_1 - \Delta K_1) - c_2^q \cdot \delta_2 (\Delta E_1 - \Delta L_1 - \Delta K_1). \quad (10)$$

Here  $\delta_1$  and  $\delta_2$  are parameters. It is possible to specify that parameters in different ways by some appropriate limits, for example by  $Q_1$  (the limit of emission units

<sup>1</sup>The Protocol allows industrialized countries to meet part of their emissions targets through activities that absorb CO<sub>2</sub> so-called carbon sinks. [17]

that the player **1** wants to buy from the player **2** at the price  $c_2^q$  on account of quota  $K_2$ ),  $Q_2$  (the limit of emission unites that the player **2** wants to sell to the player **1** on account of quota  $K_2$ ), and  $M_1$  (financial limit of the player **1**) in the following way

$$\begin{aligned} \min\{Q_1, Q_2\} &= \delta_2(\Delta E_1 - \Delta L_1 - \Delta K_1) := Q, \\ M_1 &\geq c_2^e \delta_1(\Delta E_1 - \Delta L_1 - \Delta K_1). \end{aligned}$$

From (8)–(10) we have the value of characteristic function for the coalition of two players

$$\begin{aligned} v(\{1, 2\}) &= H_1^0 + H_2^0 - H_1 - H_2 \\ &= (\Delta E_1 - \Delta L_1 - \Delta K_1)(\delta_1 c_1^e + \delta_2 c_1^e - \delta_2 c_2^q) \\ &= (\Delta E_1 - \Delta L_1 - \Delta K_1)(\delta_1 c_1^e + \delta_2(c_1^e - c_2^q)). \end{aligned} \quad (11)$$

In the case of three players' joint action we calculate

$$v(\{1, 2, 3\}) = H_1^0 + H_2^0 + H_3^0 - H_1 - H_2 - H_3. \quad (12)$$

We assume that  $c_1^e > c_2^e > c_3^e$  and  $c_1^q > c_2^q > c_3^q$ . Then the player **1**'s cost under cooperation is

$$\begin{aligned} H_1 &= c_3^q \cdot \delta_2(13)(\Delta E_1 - \Delta L_1 - \Delta K_1) \\ &\quad + c_3^e \delta_1(13)(\Delta E_1 - \Delta L_1 - \Delta K_1) \\ &\quad + c_2^q \cdot \delta_2(12)(\Delta E_1 - \Delta L_1 - \Delta K_1) \\ &\quad + c_3^e \cdot \delta_1(12)(\Delta E_1 - \Delta L_1 - \Delta K_1) \\ &\quad + c_1^e(1 - \delta_2(13) - \delta_1(13) - \delta_2(12) - \delta_1(12)) \\ &\quad \cdot (\Delta E_1 - \Delta L_1 - \Delta K_1). \end{aligned} \quad (13)$$

The first line of (13) is the cost due to the realization of the flexibility mechanisms between the players **1** and **3**, the second line is the cost due to the realization of the flexibility mechanisms between the players **1** and **2**, and the third is the cost of emission reduction in the territory of player **1**. Using the limits  $Q_1(13)$ ,  $Q_3(13)$ ,  $Q_1(12)$ ,  $Q_2(12)$ ,  $Q_2(23)$ ,  $Q_3(23)$ ,  $M_1$  and  $M_2$  defined as before, and the following notations

$$\begin{aligned} \min\{Q_1(13), Q_3(13)\} \\ = \delta_2(13)(\Delta E_1 - \Delta L_1 - \Delta K_1) := Q(13), \end{aligned}$$

$$\begin{aligned} \min\{Q_1(12), Q_2(12)\} \\ = \delta_2(12)(\Delta E_1 - \Delta L_1 - \Delta K_1) := Q(12), \end{aligned}$$

$$\begin{aligned} \min\{Q_2(23), Q_3(23)\} \\ = \delta_2(23)(\Delta E_2 - \Delta L_2 - \Delta K_2 + Q(12)) := Q(23), \end{aligned}$$

$$\begin{aligned} M_1 &\geq c_3^e \delta_1(13)(\Delta E_1 - \Delta L_1 - \Delta K_1) \\ &\quad + c_3^e \cdot \delta_1(12)(\Delta E_1 - \Delta L_1 - \Delta K_1), \end{aligned}$$

$$M_2 \geq c_3^e \cdot \delta_1(23)(\Delta E_2 - \Delta L_2 - \Delta K_2 + Q(12)),$$

let us write down the costs of the players **2** and **3** under cooperation

$$\begin{aligned} H_2 &= c_3^q \cdot \delta_2(23)(\Delta E_2 - \Delta L_2 - \Delta K_2 + Q(12)) \\ &\quad + c_3^e \cdot \delta_1(23)(\Delta E_2 - \Delta L_2 - \Delta K_2 + Q(12)) \\ &\quad + c_2^e(1 - \delta_2(23) - \delta_1(23)) \\ &\quad \cdot (\Delta E_1 - \Delta L_1 - \Delta K_1 + Q(12)) \\ &\quad - c_2^q \cdot Q(12) - c_2^e \cdot \delta_1(12)(\Delta E_2 - \Delta L_2 - \Delta K_2), \end{aligned} \quad (14)$$

$$\begin{aligned} H_3 &= \\ &c_3^e(\Delta E_3 - \Delta L_3 - \Delta K_3 + Q(13) + Q(23)) - c_3^q \cdot Q(13) \\ &- c_3^e \cdot \delta_1(13)(\Delta E_1 - \Delta L_1 - \Delta K_1) - c_3^q \cdot Q(23) \\ &- c_3^e \cdot \delta_1(23)(\Delta E_2 - \Delta L_2 - \Delta K_2 + Q(12)). \end{aligned} \quad (15)$$

Consequently from (12)–(15) and (8) we calculate the value  $v(\{1, 2, 3\})$ .

By analogy to the previous formulas we can define the characteristic function for any number of players. The values  $v(\{i\}) = 0$  for every player  $i$  conclude the construction of the characteristic function.

#### IV. IMPUTATION DISTRIBUTION PROCEDURES FOR KYOTO PROTOCOL MODEL

In this section we consider two multistage cooperative games corresponding to the model of realization of flexibility mechanisms. The characteristic function  $v(S, t)$  is the guaranteed economy in million dollars due to the co-operation (joint implementation, clean development and emission trading) during 5 years. The data for calculations were taken from [8] and [16]. Here  $S$  is a coalition of Parties (groups of Parties) in co-operation on a period  $[t, T]$ . Characteristic function values depend on a set of parameters: limitations of the emission reduction investment, emission quota, etc. Depending on the parameters we have the different variants of the game. In the following examples we have three players: **1** is European Union (EU), **2** is the new members of EU (EU-A), and **3** is Russian Federation.

##### A. Example with a time-consistent solution

Let us consider 3-person multistage cooperative game (the characteristic function is in Table I).

The sets  $X^0(t)$ ,  $t \in \mathbb{T}$ , are the following

$$\begin{aligned} X^0(t_0) &= \{\xi^0(t_0) = (54050, 14100, 25850)\}, \\ X^0(t_1) &= \{\xi^0(t_1) = (35250, 11750, 23500)\}, \\ X^0(t_2) &= \{\xi^0(t_2) = (20000, 10000, 17000)\}, \\ X^0(t_3) &= \{\xi^0(t_3) = (21000, 7000, 10000)\}, \end{aligned}$$

TABLE I  
CHARACTERISTIC FUNCTION OF THE MULTISTAGE COOPERATIVE GAME.

$t$	$v(\{1, 2, 3\}, t)$	$v(\{2, 3\}, t)$	$v(\{1, 3\}, t)$	$v(\{1, 2\}, t)$
$t_0$	94000	39950	79900	68150
$t_1$	70500	35250	58750	47000
$t_2$	47000	27000	37000	30000
$t_3$	38000	17000	31000	28000
$t_4$	22500	10480	17250	14730
$t_5$	0	0	0	0

$$X^0(t_4) = \{\xi^0(t_4) = (10750, 3980, 6500)\},$$

$$X^0(t_5) = \{\xi^0(t_5) = (0, 0, 0)\}.$$

At the moments  $t_k$ ,  $k = 0, 1, 2, 3, 5$ , the grand subcore is equal to the set  $X^0(t_k)$

$$GSC(N, v(t)) = X^0(t)$$

and the unique imputation  $\xi(t_0) = \xi^0(t_0)$  is time-consistent (see [19, Theorem 2.1]).

Let us now apply the algorithm to construct IDP for this game.

*Initial step.* We set

$$\alpha(t_0) = \xi(t_0) = (54050, 14100, 25850).$$

*Step 1.* We set

$$\alpha(t_1) = \xi(t_1) = (35250, 11750, 23500)$$

and

$$\alpha(t_0, t_1) = (18800, 2350, 2350).$$

*Step 2.* The solution of the problem (6), (7) is the vector  $\omega = (21000, 1000, 17000)$ . The sum of  $\omega$ 's components are greater than  $v(N, t_2)$ , hence we set

$$\alpha(t_2) = \omega, \quad \tilde{v}(N, t_2) = 48000,$$

$$\alpha(t_1, t_2) = (14250, 1750, 6500).$$

*Step 3.* At this step  $\omega = \xi^0(t_3)$ , hence we can set

$$\alpha(t_3) = \xi(t_3) = (21000, 7000, 10000)$$

and the payoff vector is

$$\alpha(t_2, t_3) = (0, 3000, 7000).$$

*Step 4.* We should find  $\alpha(t_4) \in GSC(N, v(t_4))$  such that  $\alpha(t_4) \leq \alpha(t_3)$ . Let us choose

$$\alpha(t_4) = \xi^0(t_4) + (400, 470, 400) = (11150, 4450, 6900),$$

then

$$\alpha(t_3, t_4) = (9850, 2550, 3100).$$

TABLE II  
CHARACTERISTIC FUNCTION OF THE MULTISTAGE COOPERATIVE GAME.

$t$	$v(\{1, 2, 3\}, t)$	$v(\{2, 3\}, t)$	$v(\{1, 3\}, t)$	$v(\{1, 2\}, t)$
$t_0$	94000	39950	79900	68150
$t_1$	70500	35250	58750	47000
$t_2$	47000	41125	21150	22325
$t_3$	33950	15680	30520	20300
$t_4$	18800	6800	15680	10620
$t_5$	0	0	0	0

*The last step.* Here  $\alpha(t_5) = (0, 0, 0)$  and

$$\alpha(t_4, t_5) = \alpha(t_4) = (11150, 4450, 6900).$$

Consequently, the corresponding payoff trajectory is

$$\begin{aligned} & \left( \begin{array}{c} 54050 \\ 14100 \\ 25850 \end{array} \right) \rightarrow \left( \begin{array}{c} 35250 \\ 11750 \\ 23500 \end{array} \right) \rightarrow \left( \begin{array}{c} 21000 \\ 10000 \\ 17000 \end{array} \right) \\ & \rightarrow \left( \begin{array}{c} 21000 \\ 7000 \\ 10000 \end{array} \right) \rightarrow \left( \begin{array}{c} 11150 \\ 4450 \\ 6900 \end{array} \right) \rightarrow \left( \begin{array}{c} 0 \\ 0 \\ 0 \end{array} \right) \end{aligned} \quad (16)$$

and the imputation distribution procedure  $\alpha$  is

$$\alpha = \left( \begin{array}{cccccc} 0 & 18800 & 14250 & 0 & 9850 & 11150 \\ 0 & 2350 & 1750 & 3000 & 2550 & 4450 \\ 0 & 2350 & 6500 & 7000 & 3100 & 6900 \end{array} \right)$$

This IDP provides the time-consistent imputation  $\xi(t_0)$  by non-negative payoffs to every player at every step of the game.

#### B. Example without time-consistent solution

In this part we use a reduced game to construct the imputation distribution procedure in the multistage cooperative game presented in Table II.

The sets  $X^0(t)$ ,  $t \in \mathbb{T}$ , are the following

$$X^0(t_0) = \{\xi^0(t_0) = (54050, 14100, 25850)\},$$

$$X^0(t_1) = \{\xi^0(t_1) = (35250, 11750, 23500)\},$$

$$X^0(t_2) = \{\xi^0(t_2) = (11150, 21150, 19975)\},$$

$$X^0(t_3) = \{\xi^0(t_3) = (17570, 2730, 12950)\},$$

$$X^0(t_4) = \{\xi^0(t_4) = (9710, 910, 5890)\},$$

$$X^0(t_5) = \{\xi^0(t_5) = (0, 0, 0)\}.$$

The unique solution  $\xi(t_0) \in GSC(N, v(t_0))$  is not time-consistent because there is no vector  $\xi^0(t) \in X^0(t)$  such that  $\xi_i(t_0) \geq \xi_i^0(t)$  for all  $i \in N$  and  $t \in \mathbb{T}$  (see [19, Theorem 2.1]). The property of time-consistency is violated at the moment  $t = t_2$ . Let us choose the vector  $\theta =$

$(18000, 9000, 20000)$  instead of  $\xi(t_2) \in GSC(N, v(t_2))$ . The vector  $\theta$  does not belong to the grand subcore of the subgame  $(N, v(t_2))$  due to the player 2; we call him a “disturbing” player. Let us create the MDM-reduced game  $(\{1, 3\}, v_{\xi^0}^{\{2\}})$ ,  $t \in \{t_2, t_3, t_4, t_5\}$ . To do this we fix the vectors  $\xi(t_3) = (17770, 3030, 13150)$  and  $\xi(t_4) = (10510, 1600, 6690)$ . Then the characteristic function of the reduced game is the following

$$\begin{aligned} v_{\xi^0}^{\{2\}}(\{1, 3\}, \theta, t_2) &= v(N, t_2) - \theta_2 = 38000, \\ v_{\xi^0}^{\{2\}}(\{1\}, \theta, t_2) &= v(\{1, 2\}, t_2) - \xi_2^0(t_2) = 1175, \\ v_{\xi^0}^{\{2\}}(\{3\}, \theta, t_2) &= v(\{2, 3\}, t_2) - \xi_2^0(t_2) = 19875; \\ v_{\xi^0}^{\{2\}}(\{1, 3\}, \xi(t_3)) &= v(N, t_3) - \xi_2(t_3) = 30920, \\ v_{\xi^0}^{\{2\}}(\{1\}, \xi(t_3)) &= v(\{1, 2\}, t_3) - \xi_2^0(t_3) = 17570, \\ v_{\xi^0}^{\{2\}}(\{3\}, \xi(t_3)) &= v(\{2, 3\}, t_3) - \xi_2^0(t_3) = 12950; \\ v_{\xi^0}^{\{2\}}(\{1, 3\}, \xi(t_4)) &= v(N, t_4) - \xi_2(t_4) = 17200, \\ v_{\xi^0}^{\{2\}}(\{1\}, \xi(t_4)) &= v(\{1, 2\}, t_4) - \xi_2^0(t_4) = 9710, \\ v_{\xi^0}^{\{2\}}(\{3\}, \xi(t_4)) &= v(\{2, 3\}, t_4) - \xi_2^0(t_4) = 5890. \end{aligned}$$

In the MDM-reduced multistage game the players 1 and 3 realize the solution from the grand subcore of this game. The corresponding payoff trajectory can be, for example, the following

$$\begin{aligned} \left( \begin{array}{c} 54050 \\ 14100 \\ 25850 \end{array} \right) &\rightarrow \left( \begin{array}{c} 35250 \\ 11750 \\ 23500 \end{array} \right) \rightarrow \left( \begin{array}{c} 18000 \\ 9000 \\ 20000 \end{array} \right) \\ &\rightarrow \left( \begin{array}{c} 17770 \\ 9000 \\ 13150 \end{array} \right) \rightarrow \left( \begin{array}{c} 10510 \\ 9000 \\ 6900 \end{array} \right) \rightarrow \left( \begin{array}{c} 0 \\ 0 \\ 0 \end{array} \right) \end{aligned}$$

and the imputation distribution procedure  $\alpha$  is

$$\alpha = \left( \begin{array}{cccccc} 0 & 18800 & 17250 & 230 & 7260 & 10510 \\ 0 & 2350 & 2750 & 0 & 0 & 9000 \\ 0 & 2350 & 3500 & 6850 & 6250 & 6200 \end{array} \right).$$

Decision maker should fix the moment when the player 2 can get the payoff of 9000. For example, it can be the moment  $t = t_5$ . This method combines both classical methods of regularization — regularization of the optimality principle and delays of total payoffs. It constructs an IDP in the time-inconsistent case.

## V. CONCLUSION

In this paper we considered two different approaches to the problem of time-consistency in the real-life multistage cooperative game. The first method let us to construct the imputation distribution procedure providing

a time-consistent solution from the grand subcore of the game. The second method works if there is no time-consistent imputations in a balanced game and it helps to regularize the game and the optimality principle. We applied both methods to the problem connected with Kyoto Protocol realization.

## REFERENCES

- [1] L. Barreto and S. Kypreos, “Emissions trading and technology deployment in an energy-systems “bottom-up” model with technology learning,” *European Journal of Operational Research* vol. 158, no. 1, pp. 243–261, 2004.
- [2] M. Davis and M. Maschler, “The kernel of a cooperative game,” *Naval Research Logistics Quarterly*, vol. 12, pp. 223–259, 1965.
- [3] M. Dementieva, P. Neittaanmäki and V. Zakharov, “Time-consistent decision making in models of co-operation,” Proc. 4th European Congress on Computational Methods in Applied Sciences and Engineering (ECCOMAS 2004), Jyväskylä, Finland, 2004, vol. 2, pp. 435.
- [4] M. Dementieva, P. Neittaanmäki and V. Zakharov, “Time-consistency and the problem of minimal reduction,” in *Game Theory and Application*, L. Petrosjan and V. Mazalov, Eds., vol. 10, 2004.
- [5] M. Finus, “Game theory and international environmental co-operation: a survey with an application to the Kyoto Protocol,” Fondazione Eni Enrico Mattei Working paper NOTA DI LAVORO 86.2000, 2000. Available online at <http://www.feem.it/Feem/Pub/Publications/WPapers/default.htm>.
- [6] F. Forgó, J. Fülop and M. Prill, “Game theoretic models for climate change negotiations,” *European Journal of Operation Research*, vol. 160, pp. 252–267, 2005. Available online at <http://www.sciencedirect.com>.
- [7] D.B. Gillies, *Some theorems on n-person games*. Ph.D. thesis, Princeton University Press, Princeton, NJ, 1953.
- [8] M. Grubb, Ch. Vrolijk and D. Brack, *The Kyoto protocol — a guide and an assessment*. Royal Institute of International Affairs, London, 1999.
- [9] L. Petrosjan, “Stability of solutions in  $n$ -person differential games,” *Bulletin of Leningrad University*, vol. 19, pp. 46–52, 1977. (in Russian)
- [10] L.A. Petrosjan and G. Zaccour, “Time-consistent Shapley value allocation of pollution cost reduction,” *Journal of Economic Dynamics & Control*, vol. 27, no. 3, pp. 381–398, 2003.
- [11] L. A. Petrosjan and N. A. Zenkevich, *Game Theory*, World Scientific Publishing, 1996.
- [12] St. Pickl, “Convex games and feasible sets in control theory,” *Mathematical Methods of Operations Research*, vol. 53, no. 1, pp. 51–66, 2001.
- [13] St. Pickl and G.-W. Weber, “Optimization of a time-discrete nonlinear dynamical system from a problem of ecology. An analytical and numerical approach,” *Vychislitelnye Tekhnologii*, vol. 6, no. 1, pp. 43–51, 2001. (in Russian)
- [14] M. Simaan and J.B. Cruz, “On the Stackelberg strategy in non-zero sum games,” *Journal of Optimization Theory and Applications*, vol. 11, pp. 533–535, 1973.
- [15] <http://unfccc.int/resource/docs/convkp/kpeng.pdf>
- [16] <http://www.wwf.ru>, <http://www.unfccc.int>, <http://www.ipcc.ch>
- [17] <http://europa.eu.int/comm/environment/press/bio00172.htm>
- [18] V. Zakharov and O-Hun Kwon, “Selectors of the core and consistency properties,” in *Game Theory and Applications*, L. Petrosjan and V. Mazalov, Eds., vol. 4, pp. 237–250, 1999.
- [19] V. Zakharov and M. Dementieva, “Multistage cooperative games and problem of time-consistency,” *International Game Theory Review*, vol. 6, no. 1, pp. 1–14, 2004.



