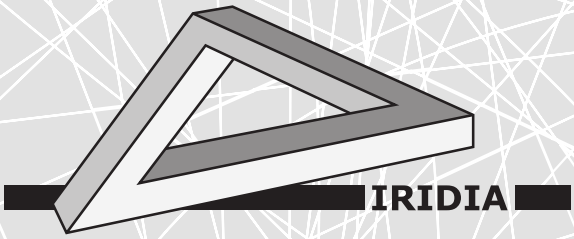


Université Libre de Bruxelles

Institut de Recherches Interdisciplinaires
et de Développements en Intelligence Artificielle

Matheuristics 2016 - Proceedings of the Sixth International Workshop on Model-based Metaheuristics

V. Maniezzo and T. Stützle

IRIDIA – Technical Report Series

Technical Report No.
TR/IRIDIA/2016-007

September 2016
Last revision: September 2016

IRIDIA – Technical Report Series
ISSN 1781-3794

Published by:

IRIDIA, *Institut de Recherches Interdisciplinaires et de Développements en Intelligence Artificielle*
Université Libre de Bruxelles
Av F. D. Roosevelt 50, CP 194/6
1050 Bruxelles, Belgium

Technical report number TR/IRIDIA/2016-007

Revision history:

TR/IRIDIA/2016-007.001 September 2016
TR/IRIDIA/2016-007.002 September 2016

Matheuristics 2016

Proceedings

Editors

Vittorio Maniezzo
Thomas Stützle

Preface

These proceedings comprise the contributions that have been presented at Matheuristics 2016, the Sixth International Workshop on Model-based Metaheuristics, which was held in Brussels from September 4–7, 2016. The Matheuristics series started in 2008 with the first workshop held in the scenic Bertinoro, Italy, organized by Vittorio Maniezzo, and since then is held every two years. The Matheuristics workshop series has been established as a primary forum for researchers working on exploiting mathematical programming techniques in a (meta)heuristic framework, granting to mathematical programming approaches the problem robustness and time effectiveness which characterize heuristics, or exploiting the mathematical programming model formulation in the customization of a heuristic for specific or general problems.

Matheuristics 2016 collects contributions that define the state of the art for the computational effectiveness and efficiency or the theoretical properties of *matheuristics*, which are algorithms and codes that integrate (meta)-heuristics and MIP strategies and software. The workshop was entirely devoted to this subject of research and to its applications. The workshop program consisted of three invited talks:

- Christian Blum talked about *Combining Metaheuristics based on Solution Construction with Exact Techniques.*
- Marco Luebbecke presented his research on *GCG: A Generic Branch-Price-and-Cut Solver.*
- Kenneth Sörensen discussed the recent controversy on supposedly new metaheuristics and asked *Metaphor-Based Metaheuristics: Who's to Blame?*

The invited talks were complemented by an invited hands-on tutorial, given by Leslie Perez, on setting up and tuning automatically algorithms with the *irace* software package, which is a software for the automated configuration of algorithms.

Apart from these invited presentation, the conference programme comprised 18 further presentation on the rich research topics in the area of Matheuristics.

We would like to use this opportunity to thank the many people who have contributed to this workshop and helped to make it a success. We would like to thank all the authors for submitting their research to Matheuristics, the international Programme Committee for the careful evaluation of the articles, the COMEX PAI project and ULB for the support of this workshop, and to the staff at IRIDIA for helping in organizational matters. Special thanks go to Alberto Franzin for the continuous support in all technical details from webpage design to practical help in preparation of the conference material.

We hope that the reader will find these proceedings useful for a reference on the current work in Matheuristics and an inspiring source for own new developments.

September 2016

Vittorio Maniezzo
Thomas Stützle

Organization

Matheuristics 2016 was organized by IRIDIA, CoDE, Université Libre de Bruxelles, Belgium.

Workshop Chair

Thomas Stützle	Université Libre de Bruxelles, Belgium

Program Chairs

Thomas Stützle	Université Libre de Bruxelles, Belgium
Vittorio Maniezzo	Università di Bologna, Italy

Organization Committee

Alberto Franzin	Université Libre de Bruxelles, Belgium
Federico Pagnozzi	Université Libre de Bruxelles, Belgium
Leslie Pérez Cáceres	Université Libre de Bruxelles, Belgium

Steering Committee

Vittorio Maniezzo	Università di Bologna, Italy
Karl F. Dörner	University of Vienna, Austria
Celso C. Ribeiro	Universidade Federal Fluminense, Brasil
Thomas Stützle	Université Libre de Bruxelles, Belgium
Stefan Voß	University of Hamburg, Germany

Program Committee

Claudia Archetti	Università di Brescia, Italy
Christian Blum	Universitaty of the Basque Country, Spain
Yves Crama	Université de Liège, Belgium
Patrick de Causmaecker	KU Leuven, Belgium
Cid C. de Souza	Universidade de Campinas, Brazil
Bernard Fortz	Université Libre de Bruxelles, Belgium
Frédéric Gardi	Innovation 24
Michel Gendreau	Université de Montréal, Canada
Peter Greistorfer	Karl-Franzens-Universität Graz, Austria
Walter Gutjahr	Universität Wien, Austria

Said Hanafi	Université de Valenciennes, France
Manuel Iori	Università di Modena e Reggio Emilia, Italy
Martine Labbé	Université Libre de Bruxelles, Belgium
Andrea Lodi	Polytechnique Montréal, Canada
Nenad Mladenovic	Université de Valenciennes, France
Sophie Parragh	Universität Wien, Austria
Günther Raidl	Vienna University of Technology, Austria
Helena Ramalhinho Lourenço	University Pompeu Fabra, Spain
Mauricio Resende	AT&T Labs Research, USA
Andrea Schaerf	Università degli Studi di Udine, Italy
Marc Sevaux	Université de Bretagne-Sud, France
Maria Grazia Speranza	Università ai Brescia, Italy
Frits Spieksma	KU Leuven, Belgium
Michael Trick	Carnegie Mellon University, USA
Pascal Van Hentenryck	University of Michigan, USA
Jean-Paul Watson	Sandia National Labs, USA
David Woodruff	University of California, USA

Sponsoring Institutions

National Funds for Scientific Research, Belgium
`http://www.fnrs.be`

Belgian Science Policy Office (through the research project COMEX)
`http://comex.ulb.ac.be`

Table of Contents

Short Papers

Abstracts

A Fix-and-Optimize VNS Algorithm Applied to the Nurse Rostering Problem

Toni Wickert, Carlo Sartori, and Luciana Buriol

Instituto de Informática
Universidade Federal do Rio Grande do Sul, Porto Alegre, Brazil
{tiwickert,buriol,cssartori}@inf.ufrgs.br

Paper ID: **104** (Full)

A Fix-and-Optimize VNS Algorithm Applied to the Nurse Rostering Problem

Toni I. Wickert, Carlo S. Sartori, and Luciana S. Buriol

Instituto de Informática
Universidade Federal do Rio Grande do Sul, Porto Alegre, Brazil
{tiwickert,buriol,cssartori}@inf.ufrgs.br

Abstract. The Nurse Rostering Problem (NRP) aims to generate schedules to nurses according to certain restrictions. Constraints could be related to work laws, hospital interests, improvements on the patient care, nurse's availability, among others. This paper presents an integer programming formulation and a fix-and-optimize variable neighborhood search algorithm to solve the NRP. The proposed algorithm was applied on the static version of the instances of the Second International Nurse Rostering Competition (INRC-II). The experimental results show that the proposed approach generates high quality solutions. In comparison with the results presented by the winners of the competition, which run a dynamic version of the problem, our results on average are not as good as the winner method, but on average better than the ranked second. However, our method generates feasible solutions for all instances within the time limits of the competition, what is not the case for the winner. Preliminary tests indicate that the difference in computation time and solution quality when working with dynamic and static versions of the problem is small, and showing this will be a future work.

Keywords: Nurse Rostering Problem, Fix and Optimize, Integer Programming, Matheuristics, Second Nurse Rostering Competition.

1 Introduction

Scheduling problems are found in a wide variety of institutions and companies. For instance, the scheduling of teachers in schools and universities, attendants in call centers, drivers in transport companies, doctors and nurses in health institutions, among others. The use of computational techniques for generating these schedules brings several benefits such as decreased time to assemble the work schedules, cost reduction and scales that best meet the employees needs.

The Nurse Rostering Problem (NRP), also known as Nurse Scheduling Problem, aims to generate schedules for health care professionals like nurses and doctors. There are different approaches in the literature to solve the problem like heuristics [11], hyper-heuristic [1], hybrid models [6], mixed integer programming (MIP) [12], and branch and price [4].

In the context of nurse rostering, the problem can vary among hospitals. To provide a better comparison among different techniques to solve the problem,

there were created libraries with synthetic instances. The main libraries available are the NSPLib [14] developed by researches from the Ghent University, Belgium and libraries provided by the International Nurse Rostering Competition I [9] and II [7] (INRC-I and INRC-II).

In this paper, we present a mixed integer programming model and a fix-and-optimize matheuristic combined with a variable neighborhood search method. The developed approach uses four different decompositions – week, nurse, day and shift – in order to solve the NRP. The method was applied on the static version of the the instances of the INRC-II and the results compared with the best known solutions (BKS) obtained by the winners of the competition.

In Section 2 we introduce the problem tackled in this work, and in Section 3 we present the integer programming formulation. In Section 4 the proposed fix-and-optimize VNS approach is detailed. In Section 5 we present experimental results, and we conclude our work in Section 6.

2 Nurse Rostering Problem

The NRP usually has two types of constraints, the hard constraints that must be satisfied, and the soft constraints that, in case of violation, are penalized in the objective function. If at least one hard constraint is violated the solution is considered infeasible. Our MIP model attends to the following constraints, according to the specification provided by the INRC-II [7]:

Hard Constraints

H1. A nurse can be assigned to at most one shift per day;
H2. Minimum number of nurses by day/shift/skill;
H3. A shift type succession must belong to a valid succession (ex.: night shift followed by morning shift is not allowed);
H4. A shift of a given skill must necessarily be fulfilled by a nurse having that skill.

Soft Constraints

S1. Optimal number of nurses by day/shift/skill;
S2. Minimum and maximum number of consecutive assignments by shift and global;
S3. Minimum and maximum number of consecutive days off;
S4. An undesired working day/shift is penalized;
S5. It is preferred that a nurse works on the two days of a weekend, or none;
S6. Minimum and maximum number of working days by schedule;
S7. Maximum number of working weekends.

2.1 Related works

Below, we present a review of the main solving techniques that have been successfully applied to the NRP.

The winner method of the INRC-I [13] partitioned each problem instance into sub-problems of manageable computational size and solved sequentially using

Table 1. Standard Formulation - INRC-II

Symbol	Definition
$n \in N$	set of nurses;
$d \in D$	set of days;
$s \in S$	set of shifts (Early, Day, Late, Night);
$k \in K$	set of skills (HeadNurse, Nurse, Caretaker, Trainee);
$l \in L$	set of skills of each nurse. Ex.: (1, 2, 0, 0) represents that a nurse has the skills HeadNurse and Nurse;
$u \in U$	set of undesired working day/shift, ex.: (1 0 0 | 0 0 0 | 0 0 0) represents that a nurse prefers to avoid working on first day at shift early;
r_{dsk}	number of required nurses at day d, shift s, skill k;
$p \in P$	set of invalid shift succession patterns, ex.: (0 0 0 | 0 0 0 | 1 1 1 || 1 1 1 | 1 1 1 | 0 0 0) represents that night shift cannot be followed by an early or late shift;
$t \in T$	set of patterns used to calculate the minimum consecutive constraints, ex.: (0 1 0 | 0 1 1 0 | 0 1 1 1 0). For instance, considering 4 as the minimum working days, the first pattern represents three violations, the second pattern has two violations, while the third pattern has a single violation.
$w \in W$	set of all Saturdays indexes;
α^1_{dsk}	limit of soft constraint 1 for day d, shift s, skill k;
β^i_n	limit of soft constraint 2..5, 8..11 for nurse n;
γ^i_s	limit of soft constraint 6..7 for shift s;
ω^i_n	weight for violating the lower and upper limits of soft constraint i for nurse n.
Decision Variables	
$x_{ndsk} \in \{0, 1\}$	1 if nurse n is allocated to shift s and day d, 0 otherwise;
$y_{nw} \in \{0, 1\}$	1 if nurse n works at weekend w, 0 otherwise.
Aux. Variables	
$a^1_{dsk} \in \mathbb{Z}^+$	store the violations of the soft constraint 1 for day d, shift s, skill k;
$b^i_{ndt} \in \mathbb{Z}^+$	store the violations of the soft constraint $i \in (2, 4)$ for nurse n on day d, pattern t;
$c^i_{nd} \in \mathbb{Z}^+$	store the violations of the soft constraint $i \in (3, 5)$ for nurse n on day d;
$e^6_{ndst} \in \mathbb{Z}^+$	store the violations of the soft constraint 6 for nurse n on day d, shift s, pattern t;
$f^7_{nds} \in \mathbb{Z}^+$	store the violations of the soft constraint 7 for nurse n on day d, shift s;
$g^8_{nds} \in \mathbb{Z}^+$	store the violations of the soft constraint 8 for nurse n on day d, shift s;
$h^9_{nw} \in \mathbb{Z}^+$	store the violations of the soft constraint 9 for nurse n on weekend w;
$j^i_n \in \mathbb{Z}^+$	store the violations of the soft constraint $i \in (10..12)$ for nurse n.

Table 2. Soft constraints index.

Index	Constraint	Constraint
1	Optimal coverage	S1
2	Minimum consecutive assignments (working days)	S2
3	Maximum consecutive assignments (working days)	S2
4	Minimum number of consecutive days off	S3
5	Maximum number of consecutive days off	S3
6	Minimum consecutive assignments to the same shift	S2
7	Maximum consecutive assignments to the same shift	S2
8	Nurse undesired working day/shift	S4
9	Complete weekend	S5
10	Minimum number of assignments over the scheduling period	S6
11	Maximum number of assignments over the scheduling period	S6
12	Total working weekends	S7

Integer Mathematical Programming. A two phase strategy was implemented. In the first phase the workload for each nurse and for each day of the week was decided, while in the second phase the specific daily shifts were assigned. In addition, local optimization techniques for searching across combinations of nurses' partial schedules were also applied. This sequence is repeated several times depending on the available computational time.

In [3] they used two different algorithms that were applied to the instances of the INCR-I. The first algorithm, a variable depth search, is an ejection chain based method. It was applied to the small instances for which a maximum computation time of 10 seconds was allowed. The second algorithm, a branch-and-price method, was applied to the medium and large instances for which maximum computation times of 10 minutes and 10 hours respectively were allowed. The pricing problem was modelled as a resource constrained shortest path problem and solved using a dynamic programming approach.

The work of [2] proposed a hyper-heuristic split into two phases. In the first phase the hyper-heuristic is deployed in the first 80% of the computation time. Next, the greedy shuffle heuristic consumes the remainder of the available time. The hyper-heuristic consists basically of two main parts: a heuristic selection method and a move acceptance criterion, where a simple random heuristic selection method was applied. That is the most straightforward selection method since it randomly takes one low-level heuristic from a list. For the move acceptance criterion, simulated annealing was applied. On the second part an extended version of the greedy shuffle heuristic introduced in [5] was applied to the best solution obtained by the hyper-heuristic. It intends to improve the solution by greedily exploring swaps of partial rosters between nurses.

Table 3. Table 3 represents a simple scheduling presented in the nurse-day view, and containing 3 nurses (N1, N2, N3), 3 shifts (Early [E], Late [L], Night [N]), and 3 skills (HeadNurse [H], Nurse [N], Caretaker [C]). As an example, Nurse 1 works on Monday on shift Late with skill Nurse, on Tuesday on shift Late with skill HeadNurse, and on Wednesday on shift Night with skill Headnurse. Character "–" represents a non-working day.

Nurse	Mon	Tue	Wed
N1	L[N]	L[H]	N[H]
N2	N[N]	N[N]	N[C]
N3	–	E[C]	–

3 Integer Programming Formulation

In this section, we present the proposed Integer Programming Formulation, considering all the hard and soft constraints of the INRC-II.

Constraints (2)-(5) assure the hard constraints H1-H4, respectively. Constraints (6) calculate the optimal coverage violations. Constraints (7) calculate the minimum consecutive assignments (working days) violations. In the equations, $S1$ = (sum of the working days) + (two extreme bits × C) + (complement of middle bits × C). For example, if the minimum are 3 consecutive working days and in the scheduling we have found the pattern 010 it means that there are 2 violations. It is calculated as $S1 = 1 + (0 \times 10) + (0 \times 10) + (0 \times 10) + b^2_{ndt} >= 3$, and as a result b^2_{ndt} will assume value 2.

Constraints (8) calculate the maximum consecutive assignments (working days) violations. Constraints (9) calculate the minimum consecutive days off violations. $S2$ is evaluated similarly like explained in equation (7), however the bits are inverted and the sum is related to free days instead of working days. Constraints (10) calculate the maximum consecutive days off violations. Constraints (11) calculate the minimum consecutive assignments to the same shift violations.

$$\textbf{Min} \quad \left[\sum_{d\in D}\sum_{s\in S}\sum_{k\in K} a^1_{dsk}\omega^1\right] + \left[\sum_{n\in N}\sum_{d\in D}\sum_{t\in T}\sum_{i=2,4} b^i_{ndt}\omega^i\right] + \left[\sum_{n\in N}\sum_{d\in D}\sum_{i=3,5} c^i_{nd}\omega^i\right] + \left[\sum_{n\in N}\sum_{d\in D}\sum_{s\in S}\sum_{t\in T} e^6_{ndst}\omega^6\right] + \left[\sum_{n\in N}\sum_{d\in D}\sum_{s\in S} f^7_{nds}\omega^7\right] + \left[\sum_{n\in N}\sum_{d\in D}\sum_{s\in S} g^8_{nds}\omega^8\right] + \left[\sum_{n\in N}\sum_{w\in W} h^9_{nw}\omega^9\right] + \left[\sum_{n\in N}\sum_{i=10..12} j^i_n\omega^i\right] \tag{1}$$

Subject to

$$\sum_{s \in S} \sum_{k \in K} x_{ndsk} \leq 1 \qquad \forall n \in N, d \in D \tag{2}$$

$$\sum_{n \in N} x_{ndsk} \geq r_{dsk} \qquad \forall d \in D, s \in S, k \in K \tag{3}$$

$$\sum_{s \in S} \sum_{k \in K} (x_{ndsk} p_{ask}) + (x_{nd+1sk} p_{ask}) \leq 1 \qquad \forall a \in P, n \in N, d \in D-1, d+1 \in D \tag{4}$$

$$(k - l_{nk}) x_{ndsk} = 0 \qquad \forall n \in N, d \in D, s \in S, k \in K \tag{5}$$

$$\sum_{n \in N} x_{ndsk} + a^1_{ndsk} \geq \alpha^1_{ndsk} \qquad \forall d \in D, s \in S, k \in K \tag{6}$$

$$S1_{ndt} + b^2_{ndt} \geq \beta^2_n \qquad \forall n \in N, d \in D, t \in T \tag{7}$$

$$\sum_{d2=d}^{\beta^3_n+1} \sum_{s \in S} \sum_{k \in K} x_{nd2sk} - c^3_{nd} \leq \beta^3_n \qquad \forall n \in N, d \in D - \beta^3_n - 1 \tag{8}$$

$$S2_{ndt} + b^4_{ndt} \geq \beta^4_n \qquad \forall n \in N, d \in D, t \in T \tag{9}$$

$$\sum_{d2=d}^{\beta^5_n+1} \sum_{s \in S} \sum_{k \in K} 1 - x_{nd2sk} - c^5_{nd} \leq \beta^5_n \qquad \forall n \in N, d \in D - \beta^5_n - 1 \tag{10}$$

$$S3_{ndst} + e^6_{ndst} \geq \gamma^6_s \qquad \forall n \in N, d \in D, s \in S, t \in T \tag{11}$$

$$\sum_{d2=d}^{\gamma^7_s+1} \sum_{k \in K} x_{nd2sk} - f^7_{nds} \leq \gamma^7_s \qquad \forall n \in N, d \in D - \gamma^7_s - 1, s \in S \tag{12}$$

$$u_{nds} - \left[\sum_{k \in K} (x_{ndsk} g^8_{nds}) \right] = 0 \qquad \forall n \in N, d \in D, s \in S \tag{13}$$

$$\sum_{s \in S} \sum_{k \in K} x_{ndsk} + x_{nd+1sk} + h^9_{nw} \leq 2B y_{nw} \qquad \forall n \in N, w \in W \tag{14}$$

$$\sum_{d \in D} \sum_{s \in S} \sum_{k \in K} x_{ndsk} + j^{10}_n \geq \beta^{10}_n \qquad \forall n \in N \tag{15}$$

$$\sum_{d \in D} \sum_{s \in S} \sum_{k \in K} x_{ndsk} - j^{11}_n \leq \beta^{11}_n \qquad \forall n \in N \tag{16}$$

$$\sum_{s \in S} \sum_{k \in K} x_{ndsk} + x_{nd+1sk} \leq 2 y_{nw} \qquad \forall n \in N, d \in W, w \in W \tag{17}$$

$$\sum_{w \in W} y_{nw} - j^{12}_n \leq \beta^{12}_n \qquad \forall n \in N \tag{18}$$

$S3$ is evaluated exactly like explained in equation (7), however the violations are stored by nurse/day/shift/pattern. Constraints (12) calculate the maximum consecutive assignments to the same shift violations. Constraints (13) calculate

the undesired day/shift assignments violations. Constraints (14) calculate the complete weekend violation. In the equation B is 1 if the complete weekend is present in the nurse's working contract, and 0 otherwise. Constraints (15) calculate the minimum total working days violations over the scheduling period. Constraints (16) calculate the maximum total working days violations over the scheduling period. Constraints (17) calculate the working weekends. Constraints (18) calculate the total working weekends violations.

4 Fix-And-Optimize VNS Algorithm

The use of fix-and-optimize heuristics with integer programming has grown in the last years. Our developed algorithm is inspired on the successful application of this type of technique in a high school timetabling problem proposed by [8]. In this section, we present the fix-and-optimize variable neighborhood search algorithm developed to solve the NRP.

First, a feasible solution is generated only considering the hard constraints. Once having a feasible solution, we choose variables to be fixed and call the solver to optimize the variables that are free.

```
1  FixAndOptimize(kMaxWeek, kLimitWeek, kMaxShift, kLimitShift, kMaxDay, kLimitDay,
   kMaxNurse, kLimitNurse, TL, STL);
2  x = generateInitialSolution();
3  kWeek = kShift = kDay = kNurse = 1;
4  do
5  |   x = fixByDay(x, kDay, kLimitDay, STL);
6  |   kDay++;
7  |   x = fixByNurse(x, kNurse, kLimitNurse, STL);
8  |   kNurse++;
9  |   x = fixByWeek(x, kWeek, kLimitWeek, STL);
10 |   kWeek++;
11 |   x = fixByShift(x, kShift, kLimitShift, STL);
12 |   kShift++;
13 |   if (kWeek > kLimitWeek) kWeek = 1;
14 |   if (kShift > kLimitShift) kShift = 1;
15 |   if (kDay > kLimitDay) kDay = 1;
16 |   if (kNurse > kLimitNurse) kNurse = 1;
17 while TL;
18 return x;
```

Algorithm 1: Fix-and-Optimize VNS algorithm.

Algorithm 1 receives several parameters where *kMax{Week, Shift Day, Nurse}* represents the maximum number of free variables of each type and *kLimit{Week, Shift Day, Nurse}* are the limits of permutations generated of each type of neighborhood. For instance, considering that there are 5 nurses and *kNurse = 1*, it is possible to generate 5 types of free nurses to optimize *NurseFreePermutationSet([1], [2], [3], [4], [5])*. However if *kNurse = 2* we have 10 possibilities *NurseFreePermutationSet([1,2], [1,3], [1,4], [1,5], [2,3], [2,4], [2,5], [3,4], [3,5], [4,5])*. If *kLimitNurse = 5*, randomly 5 items will be selected from the set *Nurse-*

FreePermutationSet, if the *kLimitNurse >= 10* the whole set *kLimitNurse* is selected.

In the beginning, the Algorithm 1 generates a feasible solution considering only the hard constraints (line 2) and using a MIP solver (CPLEX). The number of free variables to optimize are initialized with one (line 3) and the loop (lines 4 to 17) is iterated until the time limit (TL) is reached.

Inside the loop in lines 5-12, the algorithm calls the different kinds of neighborhoods, since each neighborhood is explored until no improvement is found. At each step the *kLimit{Week, Shift Day, Nurse}* is increased by 1. If the limit of each type is exceeded the variables are set to 1 (lines 13 to 16).

```
1  FixByNurse(x, kNurse, kLimitNurse, STL);
2  List<Integer,Integer[]> perm = permutation(kNurse, kLimitNurse);
3  do
4  |   currentSolutionValue = OF(x);
5  |   bestNeighborValue = OF(x);
6  |   foreach Integer free : perm do
7  |   |   fixAll(x);
8  |   |   unFix(free, x);
9  |   |   solve(x, STL);
10 |   |   if OF(x) < bestNeighborValue then
11 |   |   |   bestNeighborValue = OF(x);
12 |   |   end
13 |   end
14 |   improved = bestNeighborValue*1.2 < currentSolutionValue;
15 while improved;
16 return x;
```

Algorithm 2: Fix by nurse.

The Algorithm 2 begins generating the permutations of the nurses that will be free to be optimized, until the *kLimitNurse* is reached (line 2). The loop (lines 3 to 15) is iterated until no improvement of 20% is found. The loop starts storing the current solution value and the best neighbor value (lines 4 and 5). The nested loop in lines 6 to 13 explores the neighborhood fixing the whole problem (line 7) and unfixing only the free variables that will be optimized (line 8). On line 9 the solver CPLEX is called and executed until the optimality or Subproblem Time Limit (STL) is reached, stopping when the first stopping condition occurs. If the objective function of the current subproblem x has lower value lower than the best neighbor, then the *bestNeighborValue* variable is updated.

Tables 4 and 5 demonstrate an iteration of a fix by nurse neighborhood with *kNurse = 1* and *kNurse = 2*, respectively. The lines (Nurses) shown in gray are free to be optimized by the solver. Since the fix by week, fix by shift and fix by day follow the same idea, the pseudo-code of these algorithms were omitted.

5 Computational results

The code was written in Java and compiled/executed with OpenJDK 1.7. The experiments were ran on a Intel Core i5-2410M CPU @ 2.30GHz x 2 with 6Gb of RAM memory running Linux Mint 17.2 64-bits. The solver used was CPLEX

Table 4. Fix by Nurse (kNurse = 1)

Nurse	Mon	Tue	Wed
N1	L[N]	L[H]	N[H]
N2	N[N]	N[N]	N[C]
N3	–	E[C]	–

Nurse	Mon	Tue	Wed
N1	L[N]	N[H]	N[H]
N2	N[N]	N[N]	N[C]
N3	–	E[C]	–

Nurse	Mon	Tue	Wed
N1	L[N]	N[H]	N[H]
N2	N[H]	N[N]	N[C]
N3	–	E[C]	–

Table 5. Fix by Nurse (kNurse = 2)

Nurse	Mon	Tue	Wed
N1	L[N]	N[H]	N[H]
N2	N[H]	N[N]	N[C]
N3	–	E[C]	–

Nurse	Mon	Tue	Wed
N1	L[N]	N[N]	N[H]
N2	N[H]	L[N]	N[C]
N3	–	E[C]	–

Nurse	Mon	Tue	Wed
N1	L[N]	N[N]	N[H]
N2	E[H]	L[N]	N[C]
N3	E[C]	E[C]	–

version 12.6.2. The default CPLEX parameters were used and the parallel mode was disabled, i.e. using only 1 core. To run the experiments, the hidden instances of the INRC-II [7] were used. The *GAP* is calculated using the equation $gap = 100 \times (obj - BKS)/min(BKS, obj)$. All parameters of the fix-and-optimize VNS were tuned using irace [1].
The values are: *STL = 8s; neighborSeq=day,nurse,week,shift; kMaxWeek=2; kLimitWeek=7; kMaxShift=3; kLimitShift=3; kMaxNurse=15; kLimitNurse=34; kMaxDay=8; kLimitDay=8.*

Tables 6, 7 and 8 present the average results of 10 runs of the proposed algorithm and a comparison with the three first best results [2] obtained by the competitors of the INRC-II. *#sol* means the number of feasible solutions obtained by the respective solver and *gap* is the difference from the BKS. The first column shows the Instance ID for 35, 70 and 110 nurses, respectively. The second column (BKS) presents the values of the best known solutions. The third, fourth and fifth columns presents the results obtained by the winner, second and third positions of the INRC-II. The sixth column (F&O{6,16,27}min) presents the results obtained by the proposed algorithm using the time provided by the benchmark tool [3]. The last column (F&O 2h) is an experiment with a time limit of 2h.

[1] The irace package implements the iterated racing procedure, which is an extension of the Iterated F-race procedure. Its main purpose is to automatically configure optimization algorithms by finding the most appropriate settings given a set of instances of an optimization problem [10].

[2] The results of the 1st, 2nd and 3rd are available at http://mobiz.vives.be/inrc2.

[3] The benchmark tool is available at http://mobiz.vives.be/inrc2 and provides the time that each instance size may be run on each computer, according to the processing power.

Table 6. Instances with 35 nurses and 4 weeks.

		1st		2nd		3rd		F&O 6 min.		F&O 2h	
Instance Id	BKS	#sol	gap	#sol	gap	#sol	gap	#sol	gap	#sol	gap
n035w4_0_1-7-1-8	1630	10	3.99	10	7.76	10	29.17	10	7.36	10	-3.16
n035w4_0_4-2-1-6	1800	10	4.06	10	12.31	10	30.11	10	12.22	10	-1.12
n035w4_0_5-9-5-6	1755	10	6.38	10	4.53	10	26.44	10	6.27	10	-0.85
n035w4_0_9-8-7-7	1540	10	5.00	10	11.92	10	34.35	10	13.96	10	1.62
n035w4_1_0-6-9-2	1500	10	4.63	10	15.80	10	37.67	10	17.67	10	0.67
n035w4_2_8-6-7-1	1490	10	3.66	10	10.37	10	33.72	10	12.08	10	-0.34
n035w4_2_8-8-7-5	1255	10	7.77	10	9.20	10	47.01	10	11.55	10	-3.72
n035w4_2_9-2-2-6	1705	10	4.40	10	14.22	10	29.41	10	9.97	10	-0.59
n035w4_2_9-7-2-2	1650	10	7.55	10	19.42	10	39.67	10	12.73	10	-2.48
n035w4_2_9-9-2-1	1620	10	7.35	10	18.98	10	37.38	10	17.59	10	2.16
average	-	10.00	5.48	10.00	12.45	10.00	34.49	10.00	12.14	10.00	-0.61

Table 6 presents the results for 35 nurses and 4 weeks. The column *#sol* shows that all algorithms obtained feasible solutions on 10 runs. Last line (in gray) presents the average *gap* where the proposed algorithm obtained a value of *12.14* that is, a value between the first *5.48* and the second *12.45* winners.

Table 7. Instances with 70 nurses and 4 weeks.

		1st		2nd		3rd		F&O 16 min.		F&O 2h	
Instance Id	BKS	#sol	gap	#sol	gap	#sol	gap	#sol	gap	#sol	gap
n070w4_0_3-6-5-1	2700	10	7.11	10	16.70	10	36.83	10	32.59	10	10.56
n070w4_0_4-9-6-7	2430	10	7.22	10	18.89	10	36.07	10	42.80	10	11.93
n070w4_0_4-9-7-6	2475	10	7.94	10	19.11	10	37.13	10	28.89	10	10.10
n070w4_0_8-6-0-8	2435	10	9.34	10	23.86	10	46.18	10	43.12	10	9.86
n070w4_0_9-1-7-5	2320	10	9.33	10	23.45	10	37.87	10	55.17	10	14.66
n070w4_1_1-3-8-8	2700	10	8.07	10	16.09	10	31.59	10	32.41	10	8.52
n070w4_2_0-5-6-8	2520	10	8.75	10	19.52	10	40.30	10	27.18	10	11.71
n070w4_2_3-5-8-2	2615	10	5.70	10	20.13	10	30.42	10	51.05	10	8.60
n070w4_2_5-8-2-5	2540	10	7.44	10	18.33	10	33.80	10	45.08	10	11.22
n070w4_2_9-5-6-5	2615	10	6.14	10	16.48	10	35.45	10	29.64	10	8.03
average	-	10.00	7.70	10.00	19.26	10.00	36.56	10.00	38.79	10.00	10.52

Table 7 presents the results for 70 nurses and 4 weeks. All algorithms obtained feasible solutions on 10 runs (column *#sol*). Last line (in gray) presents the average *gap* where the proposed algorithm obtained a value of *38.79* close to the third competitor (*36.56*), whereas the first obtained *7.70* and the second *19.26*.

Table 8. Instances with 110 nurses and 4 weeks.

		1st		2nd		3rd		F&O 27 min.		F&O 2h	
Instance Id	BKS	#sol	gap	#sol	gap	#sol	gap	#sol	gap	#sol	gap
n110w4_0_1-4-2-8	2710	10	2.05	10	30.59	10	42.47	10	18.08	10	14.58
n110w4_0_1-9-3-5	2920	10	3.42	10	25.45	10	44.83	10	28.60	10	21.58
n110w4_1_0-1-6-4	2850	9	18.03	10	32.25	10	45.60	10	23.68	10	20.53
n110w4_1_0-5-8-8	2820	10	3.67	10	26.58	10	49.49	10	23.40	10	16.49
n110w4_1_2-9-2-0	3345	10	2.80	10	22.33	10	38.36	10	27.50	10	2.24
n110w4_1_4-8-7-2	2805	10	5.42	10	30.52	10	48.02	10	22.64	10	26.20
n110w4_2_0-2-7-0	3005	5	107.19	10	29.90	10	45.81	10	23.79	10	13.64
n110w4_2_5-1-3-0	2925	10	3.68	10	24.36	10	50.22	10	17.95	10	36.58
n110w4_2_8-9-9-2	3415	10	2.40	10	17.86	10	38.93	10	21.96	10	4.69
n110w4_2_9-8-4-9	3135	10	3.14	10	20.22	10	39.79	10	19.94	10	18.80
average	-	9.40	15.18	10.00	26.01	10.00	44.35	10.00	22.75	10.00	17.53

Table 8 presents the results for 110 nurses and 4 weeks. The solver of the winner competitor did not obtain feasible solutions for two instances (see column *#sol*). On the instance *n110w4_1_0-1-6-4* they obtained 9 feasible solutions out of 10 runs, and on instance *n110w4_2_0-2-7-0* they obtained 5 feasible solutions. Last line (in gray) presents the average *gap* where the proposed algorithm obtained a value of *22.75* that is, a value between the first *15.18* and the second *26.01*. Also, in all runs the proposed algorithm obtained feasible solutions.

The average costs (rounded to integer) of soft constraints violations S1 to S7 (*F&O 2h*) for 35 nurses and 4 weeks were 596, 222, 69, 141, 241, 18, and 300. For 70 nurses and 4 weeks the costs were 1016, 341, 230, 209, 518, 27, and 459. For 110 nurses and 4 weeks the costs were 2170, 229, 78, 335, 552, 3, and 123.

In the competition the problem was solved week by week, i.e, once having the subproblem related to one week solved, the next week could then be solved. In this work we solve the whole problem at once, instead of solving it week by week. The advantage of solving the whole problem at once is that we do not work with uncertainty in the constraints of minimum/maximum working days over the schedule period and the maximum working weekends.

6 Conclusions and Future Works

In this work we presented a fix-and-optimize VNS applied to the instances of the INRC-II. We fixed the time according to the limit provided by the benchmark tool. When compared to the BKS, the algorithm, in average, presented results between the first and third competitor of the INRC-II.

In future works we intend to implement cuts to improve the performance of the proposed algorithm, provide a new integer programming formulation based on network flows, adapt the fix-and-optimize algorithm to work with the new formulation, and run on the the dynamic version of the problem of the INRC-II.

7 Acknowledgments

The authors acknowledge the partial support of CNPq project 462425/2014-2.

References

1. Asta, S., Özcan, E., Curtois, T.: A tensor based hyper-heuristic for nurse rostering. Knowledge-Based Systems 98, 185 – 199 (2016)
2. Bilgin, B., Demeester, P., Mısır, M., Vancroonenburg, W., Berghe, G.V., Wauters, T.: A hyper-heuristic combined with a greedy shuffle approach to the nurse rostering competition. Proceedings of the 8th International Conference on the Practice and Theory of Automated Timetabling (PATAT'10) (2010)
3. Burke, E.K., Curtois, T.: New computational results for nurse rostering benchmark instances. Tech. rep., http://www.cs.nott.ac.uk/~psztc/new_computational_results_for_nurse_rostering_benchmark_instances.pdf
4. Burke, E.K., Curtois, T.: New approaches to nurse rostering benchmark instances. European Journal of Operational Research 237(1), 71 – 81 (2014)
5. Burke, E.K., De Causmaecker, P., Vanden Berghe, G.: A hybrid tabu search algorithm for the nurse rostering problem. In: McKay, B., Yao, X., Newton, C., Kim, J.H., Furuhashi, T. (eds.) Simulated Evolution and Learning, Lecture Notes in Computer Science, vol. 1585, pp. 187–194. Springer Berlin Heidelberg (1999)
6. Burke, E.K., Li, J., Qu, R.: A hybrid model of integer programming and variable neighbourhood search for highly-constrained nurse rostering problems. European Journal of Operational Research 203(2), 484 – 493 (2010)
7. Ceschia, S., Dang, N.T.T., Causmaecker, P.D., Haspeslagh, S., Schaerf, A.: Second international nurse rostering competition (2014), http://mobiz.vives.be/inrc2/wp-content/uploads/2014/10/INRC2.pdf
8. Dorneles, A.P., Araujo, O.C., Buriol, L.S.: A fix-and-optimize heuristic for the high school timetabling problem. Computers and Operations Research 52, Part A, 29 – 38 (2014), http://www.sciencedirect.com/science/article/pii/S0305054814001816
9. Haspeslagh, S., Causmaecker, P.D., Stølevik, M., Schaerf, A.: First international nurse rostering competition (2010), https://www.kuleuven-kulak.be/~u0041139/nrpcompetition/nrpcompetition_description.pdf
10. López-Ibáñez, M., Dubois-Lacoste, J., Stützle, T., Birattari, M.: The irace package, iterated race for automatic algorithm configuration (TR/IRIDIA/2011-004) (2011), http://iridia.ulb.ac.be/IridiaTrSeries/IridiaTr2011-004.pdf
11. Osogami, T., Imai, H.: Classification of various neighborhood operations for the nurse scheduling problem. In: Goos, G., Hartmanis, J., van Leeuwen, J., Lee, D., Teng, S.H. (eds.) Algorithms and Computation, Lecture Notes in Computer Science, vol. 1969, pp. 72–83. Springer Berlin Heidelberg (2000)
12. Santos, H.G., Toffolo, T.A.M., Gomes, R.A.M., Ribas, S.: Integer programming techniques for the nurse rostering problem. Annals of Operations Research pp. 1–27 (2014)
13. Valouxis, C., Gogos, C., Goulas, G., Alefragis, P., Housos, E.: A systematic two phase approach for the nurse rostering problem. European Journal of Operational Research 219(2), 425 – 433 (2012)
14. Vanhoucke, M., Maenhout, B.: Nsplib – a nurse scheduling problem library: A tool to evaluate (meta-)heuristic procedures (2005), http://www.projectmanagement.ugent.be/sites/default/files/files/nsp/PaperNSPLib.pdf

A Hybrid Tabu Search and LP Heuristic for the Maximum Total Flow with Flexible Arc Outages*

Qipeng Xu, Xi Liu, and Lanbo Zheng

Port Logistics Research Center, Wuhan University of Technology,
1040,Heping road, Wuhan, China, 430063
{xqpchina,liuxichina,lanbozheng}@whut.edu.cn

Abstract. We present a hybrid tabu search and integer linear programming method for the maximum total flow with flexible arc outages (MaxTFFAO). The MaxTFFAO problem is motivated from maintenance scheduling for infrastructures of a coal chain network. In this problem, a network with arc capacities is given, together with, for every arc of the network, a set of maintenance shutdowns that need to be carried out within specific time windows. The objective is to find a feasible maintenance schedule of arcs on the network so that the total flow over the planning time horizon is maximized. The MaxTFFAO problem has a very attractive structure that blends maximum flow with scheduling, and is strongly $\mathcal{NP}$-hard even when the underlying network is a path. In this paper, we present a tabu search strategy that uses neighborhood solutions that the effectively explored by the primal and dual information from the maximum flow solver. We implement and compare this approach with previous local and global optimization approaches under the same experimental settings. The computational results show the efficiency of the new hybrid method.

Keywords: network flows, scheduling, tabu search, hybrid optimization

1 Introduction

Networks are often observed in our everyday life. Network infrastructures, such as track sections in a railway network or a pipe section in a water network are to be maintained regularly to ensure a stable function of the network. However maintenance activities may cause capacity reductions on network projects and therefore the overall efficiency of the network. The MaxTFFAO problem, first appearing in [1], was motivated by a study of a bulk goods export supply chain [2, 3], in which maintenance jobs on railway tracks and terminal equipments were scheduled so as to maximize the total throughput of the system.

In this problem, a network with arc capacities is given, together with, for every arc of the network, a set of maintenance jobs that need to be carried out

* This research was supported by the national natural science foundation of China, NO. 71501152

within specific time windows. When maintenance jobs are being carried out on an arc, the arc is on an outage, leaving no flow to pass through. The time windows for maintenance jobs provide certain flexibilities in making maintenance scheduling plans and careful coordination of the arc maintenance jobs can dramatically reduce the impact of the shutdowns on the flow carried by the network. The objective is to find a feasible maintenance schedule so that the total flow over the planning time horizon is maximized. Strong $\mathcal{NP}$-hardness of the problem is established in [1] and the complexity of a variety of special cases is investigated in [4].

When a maintenance schedule is given, the MaxTFFAO problem resembles the dynamic network flow problems which were first presented by Ford and Fulkerson [5] and have been the subject of intense studies in recent years; see, for example, Koch et al. [6] and Skutella [7]. In the application of interest to us, there are no transit times on arcs, but the capacities vary over time. For a given maintenance schedule, the capacities on the arcs jump between zero and their natural capacity, and so are piecewise constants. Thus the problem of evaluating a maintenance schedule could be viewed as a dynamic maximum flow of this type. However, in the MaxTFFAO case the piecewise constant function is a function of the maintenance schedule, and hence of the schedule decision variables. This makes the problem quite different.

This problem does have a superficial resemblance to machine scheduling problem (see, e.g., the book by Pinedo [8]), but there is no underlying machine, and the association of jobs with network arcs and a maximum flow objective give it quite a different character. Classical machine scheduling, in some sense, seeks to carry out jobs as quickly as possible. The maximum flow objective motivates quite different strategies. For example, if arcs are "in sequence" in some sense, it is better to overlap the corresponding maintenance jobs in time as much as possible, whereas if they are "in parallel", it is better to schedule them with as little overlap as possible.

The MaxTFFAO problem naturally mixes network flows with scheduling and exhibits a rich structure, making it attractive for a combination study of combinatorial algorithms, integer programming and heuristics. Recent studies on hybridization of mathematical programming and meta-heuristics algorithms prove their strengths in solving hard combinatorial optimization problems concerning real-world applications [9, 10]. Voß and Fink successfully combine tabu search with simulated annealing for the minimum weight vertex cover problem [11]. Pellegrini et al. investigate in details the sensitivity of reactive tabu search to its meta-parameters in addressing the quadratic assignment problem and the maximum clique problem [12].

In a previous study of the MaxTFFAO problem [1], several local search algorithms integrating the information from a network flow solver are explored. Several methodologies in finding and evaluating effective moves are discovered. However, all these algorithms are based on a greedy searching approach where only monotonic moves are considered and knowledge attained during the search procedure is not well utilized. Tabu search, as suggested by Glover [13, 14], us-

ing tabu list somehow as a proper learning method, provides a more intelligent searching approach. In this paper, an integer and linear programming based maximum flow algorithm is integrated within a tabu search framework. The primal and dual information of the flow variables is explored to define effective neighborhood structures. The tabu list and aspiration criteria are adaptively maintained to implement various search strategies. We examine the effectiveness of this approach through extensive experimental comparisons with the existing algorithms on the same problem instances.

The rest of the paper is organized as follows. In Section 2, we give definition to the problem, elaborate its features, and formulate it as an integer program. In Section 3, we first describe in turn, the way to evaluate objective values, neighborhood structures and effects of moves that are used in a local search framework. Then the tabu search procedure for the MaxTFFAO problem is presented. In Section 4, we report the computational results from testings of the new and existing algorithms on the same instances within the same experimental environment. In Section 5, the overall conclusions are presented, followed by some discussions on further research.

2 Problem description and formulation

The MaxTFFAO problem is defined over a network $G = (N, A, s, s', u)$, with node set N, arc set A, source $s \in N$, sink $s' \in N$, and a nonnegative integral capacity vector $u = (u_a)_{a \in A}$. By $\delta^-(v)$ and $\delta^+(v)$ we denote the set of incoming and outgoing arcs of node v, respectively. We consider the network over a time horizon $[T] := \{1, 2, \ldots, T\}$. A *maintenance job* j is specified by its associated arc $a_j \in A$, its processing time $p_j \in \mathbb{N}$, its release date $r_j \in [T]$, and its deadline $d_j \in [T]$. Let J be the set of maintenance jobs, and let J_a denote the set of jobs $j \in J$ with $a_j = a$. For each job $j \in J$ we have to choose a starting time $S_j \in [r_j, d_j - p_j + 1]$ within the *time window* for the job, and our objective is to maximize the total flow from s to s'. In this problem, all maintenance jobs must be performed exactly once and scheduling a maintenance job to start at time S_j means that the arc a_j has capacity zero at time $t \in [S_j, S_j + P_j - 1]$, and u_a otherwise.

We can now define the problem as a mixed integer program which we will use as a basis for our solution techniques. The decision variables are:

- $\phi_{at} \in \mathbb{R}_+$ is the flow variable on arc a at time t for $a \in A$,and $t \in [T]$.
- $x_{at} \in \{0, 1\}$ is the auxiliary binary variable indicating the availability of arc a at time t.
- $y_{jt} \in \{0, 1\}$ is a decision variable indicating if job j starts at time t for $j \in J$ and $t \in [r_j, d_j - p_j + 1]$.

The *maximum total flow with flexible arc outages(MaxTFFAO)* problem is formally defined as the following integer programming formulation.

$$z = \max \sum_{i=1}^{T} \sum_{a \in \delta^+(s)} \phi_{at} \tag{1}$$

s.t.

$$\sum_{a\in\delta^-(v)} \phi_{at} - \sum_{a\in\delta^+(v)} \phi_{at} = 0 \quad (v \in N \setminus \{s, s'\}, t \in [T]), \tag{2}$$

$$\phi_{at} \leq u_a x_{at} \quad (a \in A, t \in [T]), \tag{3}$$

$$\sum_{t=r_j}^{d_j-p_j+1} y_{jt} = 1 \quad (j \in J), \tag{4}$$

$$x_{at} + \sum_{t'=\max\{r_j, t-p_j+1\}}^{\min\{t, d_j-p_j+1\}} y_{jt'} \leq 1 \quad (a \in A, t \in [T], j \in J_a). \tag{5}$$

The objective (1) is to maximize the total throughput. Constraints (2) and (3) are flow conservation and capacity constraints, respectively, (4) requires that every job j is scheduled exactly once, and (5) ensures that an arc is not available while a job is being processed. The MIP solution derived from a commercial software provides a baseline for our computational testing.

3 Tabu based local search for MaxTFFAO

The approaches from [1] provide an efficient way in finding good neighbors and evaluating the effects of moves and can be integrated as a good base in a local search framework. The tabu search presented in this paper is based, again, on those findings. In this section, we first briefly introduce how these parts are done in order to provide a whole picture of the tabu search algorithm, then the detailed settings and implementation of the tabu search algorithm are presented.

3.1 Evaluating the objective function

It is not hard to observe from the problem structure that, if the start time indicator variables y_{jt} are given for all jobs $j \in J$, the values x_{at} can be fixed, and then the best solution for the given $\mathbf{y}$ can be determined by solving T max flow problems. However, to implement an efficient local search algorithm, it would be infeasible to solve T max flow problems every time to evaluate a candidate movement. Fortunately, as observed in [1], the problem structure has a nice property that enables a more intelligent way to calculate the objective function. We briefly describe the method here to complete the description of a tabu search algorithm, and the audiences are referred to [1] for more details.

For a given solution vector $\mathbf{y}$, that is, for each job j, a unique time t with $y_{jt} = 1$, let S_j denote the start time of job j, we can associate with each solution a set of times $R = \{S_j, S_j + p_j : j \in J\} \bigcup \{1, T + 1\}$. If we denote the elements of R by $1 = t_0 < t_1 < \cdots < t_{M-1} < t_M = T + 1$, then they divide the planning time horizon $[T]$ into a set of *time slices*. The set $[t_{i-1}, t_i - 1]$ is called time slice i, and its length is denoted by $l_i = t_i - t_{i-1}$. The network structure within a

time slice is constant and any pair of two consecutive time slices differs in at least one arc.

Based on the above observations, the objective function can be evaluated as described in Algorithm 1.

Algorithm 1 Objective evaluation

Input: Schedule given by S_j for $j \in J$,
$R = \{S_j, S_j + p_j : j \in J\} \bigcup \{1, T+1\} = \{1 = t_0 < t_1 < \cdots < t_{M-1} < t_M = T+1\}$
Construct the network (N, A, s, s', u)
for i=1 to M **do**
  Update upper bounds of the flow variables according to the outages in time slice i
  (Re) solve the network flow problem and store the max flow z_i
Output: $z = \sum_{i=1}^{M} z_i * l_i$

3.2 Determining the optimal neighbours

In our design of the tabu search algorithm, we consider only the most simple neighbourhood structure induced by single job movements. Next, this paper characterizes the optimal neighbours, implying an linear programming based exact method to determine an optimal neighbour.

Preliminary considerations Moving a job from its current start time S_j to another start time S'_j has the following two different effects.

1. For any time $t \in [S_j, S_j + p_j - 1] \setminus [S'_j, S'_j + p_j - 1]$ the arc a_j is released and we gain capacity on this arc which could increase the max flow for time t.
2. For any time $t \in [S'_j, S'_j + p_j - 1] \setminus [S_j, S_j + p_j - 1]$, the arc is lost, if the lost arc in the current max flow has positive flow for time t, the movement might decrease the max flow for time t.

In order to conveniently measure the impact of the above effects, we use two more parameters.

1. z^+_{ai}: the maximum flow of time slice i, with arc a added if it is missing in the current solution.
2. z^-_{ai}: the maximum flow of time slice i, with arc a removed if it is present in the current solution.

It is not hard to observe that $z^-_{ai} \le z_i \le z^+_{ai}$ for all $a \in A$ and $i \in [M]$. Also, if the availability of an arc a for some time slice i is not effected by the move, i.e. $x_{at} = 1$ for $t \in [t_{i-1}, t_i - 1]$ (or $x_{at} = 0$ for $t \in [t_{i-1}, t_i - 1]$), then we have $z^+_{ai} = z_i$ (or $z^-_{ai} = z_i$). Otherwise, if an unavailable arc a (for $x_{at} = 0, t \in [t_{i-1}, t_i - 1]$) is released, we denote the possible increase by $\Delta^+_{ai} = z^+_{ai} - z_i$; and if an available

arc a (for $x_{at} = 0, t \in [t_{i-1}, t_i - 1]$) is removed, we denote the possible decrease by $\Delta_{ai}^- = z_i - z_{ai}^-$.

Now, we only have to focus on the set of time slices effected by the move. Let $\tau_j^+(S_j')$ denote the set of time slices that are covered by $[S_j, S_j + p_j - 1]$ and will be (at least partially) uncovered by the move, and let $\tau_j^-(S_j')$ denote the set of time slices that are not covered by $[S_j, S_j + p_j - 1]$ and will be (at least partially) covered by the move. Also for each $i \in \tau_j^+(S_j') \bigcup \tau_j^-(S_j')$, the length of the time slice covered by $[S_j', S_j' + p_j - 1]$, is denoted by $l_{ij}^-(S_j')$. Then the overall effect on the objective can be evaluated in function (6).

$$\Delta_j(S_j') = \sum_{i \in \tau_j^+(S_j')} \Delta_{ai}^+ \cdot (l_i - l_{ij}^-(S_j')) - \sum_{i \in \tau_j^-(S_j')} \Delta_{ai}^- \cdot l_{ij}^-(S_j'). \tag{6}$$

Provided Δ_{ai}^+ and Δ_{ai}^- have been calculated for the appropriate time slices, it is thus straightforward to calculate $\Delta_j(S_j')$ for any j and S_j', and hence to determine an optimal neighbour.

Therefore, finding an optimal neighbour of the given schedule $(S_j)_{j \in J}$ is equivalent to

$$\max\{\Delta_j(S_j') : j \in J, S_j' \in [r_j, d_j - p_j + 1]\}.$$

If $\Delta_j(S_j') \leq 0$ for all pairs (j, S_j'), there is no improving solution neighborhood for the current schedule.

An LP-based method The above preliminary considerations suggest an immediate local search strategy: compute $\Delta_j(S_j')$ for (a subset of) all pairs (j, S_j'), make a move according to some selection criteria on $\Delta_j(S_j')$, and iterate. We use a parameter *size* to control the size of the search space. This approach looks computational prohibitive as it requires solving two max flow problems for each pair of arc a and time slice i. However, the process could be more efficient than it appears because of the following insights of the problem.

- Only the jobs on arc a covered by some time slices i with $\Delta_{ai}^+ > 0$ (the *promising jobs*) can be moved to give a better solution, and Δ_{ai}^+ can only be positive if the reduced cost of arc a in a maximum flow problem is positive;
- If an arc is added in a time slice where it was previously blocked, the flow stays primal feasible but no longer be optimal, the current max flow for time slice i can be used as a "warm start" to calculate Δ_{ai}^+ with the primal simplex method;
- Similarly, if an arc with nonzero flow is taken out, the dual stays feasible, the dual simplex method can be used to calculate Δ_{ai}^- from the current max flow for time slice i.

For ease of implementation, and so as to more readily exploit reduced cost information, and access primal and dual algorithm variants, a linear programming solver is used other than some combinatorial algorithms to solve the max flow subproblems in the algorithm. The overall method is more formally described in Algorithm 2.

Algorithm 2 Finding promising jobs

Input: *size*
Output: $PromisingJobs$, Δ_{ai}^{+}, Δ_{ai}^{-}
  for $i = 1$ to M **do**
    $A_i^{out} = \{a \in A : x_{at} = 0 \text{ for } t \in [t_{i-1}, t_i - 1]\}$
    for $a \in A_i^{out}$ **do**
      $\Delta_{ai}^{+} = z_{ai}^{+} - z_i$
      if $\Delta_{ai}^{+} > 0$ **then**
        Add the job j with $a_j = a$ and time window containing slice i to $PromisingJobs$
  for $j \in PromisingJobs$ **do**
    Put $i_0 = min\{i : t_i \geq r_j\}$ and $i_1 = max\{i : t_i \leq d_j + 1\} - 1$
    for $i = i_0$ to i_1 **do**
      $\Delta_{a_j i}^{-} = z_i - z_{ai}^{-}$

3.3 Tabu search procedure for the MaxTFFAO problem

The local search algorithms presented in [1] all follow a greedy approach and easily get trapped into local optima. A different and always better procedure involves the use of non-monotomic moves in an attempt to stay away from a possible local minimum and thus achieve a better overall solution. In particular, whenever some move is done the knowledge attained is stored in some memory space in order to utilize it at subsequent steps. This learning procedure is often considered as an important element of most artificial intelligence type approaches for the solution of difficult combinatorial problems. In this section, we describe how tabu search which belongs in this class of methods can be used to help solving the MaxTFFAO problem. Main features of the tabu search algorithm are as follows.

1) Neighbourhood: as stated in section 3.2, we only consider neighbourhood structure induced by single job movements. Within each iteration, we consider a subset of all pairs (j, S'_j), and select the optimal move. With a proper control of the size of the searching space and a handy exploit of the flow information by a linear programming solver, the neighourhood search is performed in a more efficient way.

2) Tabu list: the tabu list is maintained as a vector of (j, S'_j) pairs that each item denotes the start time of a particular job. As a small tabu list can lead to unnecessary cycles and a too large list may decrease the possibility of locating better solutions, we carefully set the length of the tabu list to a fixed number verified by the experimental results. In addition, we also maintain a *long list* for the purpose of diversification.

3) Stopping criteria: a time limit and a total number of iterations.

4) Diversification: a variable NI is maintained to record the number of consecutive iterations without improvement. Once a pre-setting parameter L is reached, half of the jobs are randomly selected to start from new times. The new start times must not be on the *long list*. With using the longer tabu list the

new starting point of the next major iteration will allow the algorithm to better explored different previously unexplored parts of the solution space.

The complete tabu search procedure is shown in Algorithm 3 where *size* denotes the size of the search space within each iteration.

Algorithm 3 TabuWithLP Heuristic

Input: Initial schedule S
Output: z_best
  Initialize time slicing and flow problems with S (Algorithm 1)
  Set $z_{best} = z$, $LongList = \emptyset, totalIter = 0, Numb =$ number of jobs
  while (not Stopping Criteria) **do**
    set $iteration = 1$, $NI = 0$, $TabuList = \emptyset$,
    while (not Stopping Criteria) **do**
      $size = \max\{\frac{Numb}{N}, \frac{Numb}{iteration}\}$
      Finding promising jobs based on $size$ (Algorithm 2)
      for $j \in PromisingJobs$ and $S'_j \in [r_j, d_j - p_j + 1]$ **do**
        if $(j, S'_j) \notin TabuList$ **then**
          calculate $\Delta_j(S'_j)$
      if $max_{j,S'_j} \Delta_j(S'_j) < 0$ **then**
        Diversification with respecting to the $LongList$
        Break
      else
        Choose (j, S'_j) with maximal $\Delta_j(S'_j)$
        Update time slicing and resolve the max flow problems with changed input data to get $z_{current}$
        if $z_{current} > z_{best}$ **then**
          $z_{best} = z_{current}$
        if $max_{j,S'_j} \Delta_j(S'_j) = 0$ **then**
          $NI = NI + 1$
          if $NI = L$ **then**
            Diversification with respecting to the $LongList$
            Break
        if $|TabuList| > size$ **then**
          remove the earliest added item from $TabuList$
        add (j, S'_j) to $TabuList$, add (j, S'_j) to $LongList$
        $iteration++$, $totalIter++$

4 Computational experiments

The computational experiments run on the all-in-one PC with an Intel dual core processor running at 2.50GHz, with 12GB of RAM and 64-bit Windows operating system. Previous local search algorithms are reimplemented, together with this tabu search algorithm under the same environment with Python 2.7.9 and we use Gurobi 6.5.1 as a baseline linear programming solver for our experiments.

All algorithms are tested with the randomly generated instances with exactly the same parameter settings as described in [1].

4.1 Experimental setup

Our tests are carried out for a time horizon with $T = 1000$. The test instances include eight easy to difficult networks with the numbers of nodes range from 12 to 64 and the numbers of edges range from 32 to 240. The test instances are divided into two sets that for data set 1, there are a variety of time window sizes of jobs (randomly selected in $[1, 35]$), and for data set 2, all time windows are large ($[25, 35]$).

Three existing algorithms are implemented and compared with the tabu search algorithm in our computational study, where GUROBI denotes the exact method by GUROBI with default settings, and GreedyResched, GreedyRandResched represent the greedy local search algorithms with single job movement presented in [1]. For all algorithms, we impose a time limit of 1800 seconds and a total iteration limit of 1000. All of them start with an initiate solution given by $S_j = \lfloor \frac{r_j + d_j}{2} \rfloor$.

For parameter settings of the tabu search algorithm, most of them are motivated by experimental experience. We observed that if we set $N = 5$, at least cover 90% of all *PromisingJobs* will be checked and 2 consecutive iterations without improvement shows a better performance. We also observed that local optima is rare for the specialness of the problem, so we fix the tabu list length to 10.

4.2 Results

Considering the randomness of the algorithms, each instance is tested 10 rounds through the experiment, and we use the relative gap (average and maximal) to compare solutions. Here the relative gap is computed as $(z' - z)/z'$ where z' is the best upper bound obtained by GUROBI in 30 min, and z is the objective value of the best solution found by the considered algorithm within a time limit (5 min or 30 min). The number of 1 to 8 in the table represent the eight different networks mentioned in 4.1.

The computational results show that GUROBI outperforms other algorithms for small to median instances. But for very difficult instances, i.e. instances from data set 2 with network structures $6-8$, the tabu search algorithm gives the best performance. This computational study again provide a solid evidence that tabu search algorithm can easily beat greedy heuristics in solving large and complex combinatorial optimization problems.

5 Conclusion and future work

In this paper, we present a hybrid tabu search and linear programming approach in solving the maximum total flow with flexible arc outages problem. The special structure of the problem which blends dynamic network flow and scheduling

Table 1. Computational results for data set 1 with running time limited to 5 min.

		1	2	3	4	5	6	7	8
	avg gap	0.6	5.1	0.6	7.2	15.8	21.5	12.6	16.0
GUROBI	max gap	1.6	12.1	1.5	17.5	22.5	29.8	21.2	24.2
	# best sol	10	10	10	10	8	9	7	3
	avg gap	8.6	15.4	8.7	14.3	17.1	25.4	14.2	15.8
GR	max gap	13.8	18.0	12.3	20.1	22.5	28.5	18.9	23.5
	# best sol	0	0	0	0	1	0	0	1
	avg gap	9.1	15.6	8.7	14.0	17.2	25.5	14.1	16.1
GRR	max gap	13.7	18.3	12.5	19.5	22.7	28.7	18.5	23.3
	# best sol	0	0	0	0	1	0	1	0
	avg gap	7.9	14.7	8.4	14.4	17.8	25.3	14.1	15.6
TabuWithLP	max gap	12.2	17.5	11.4	20.1	23.6	28.5	18.9	23.3
	# best sol	0	0	0	0	0	1	2	6

Table 2. Computational results for data set 1 with running time limited to 30 min.

		1	2	3	4	5	6	7	8
	avg gap	0.4	2.4	0.4	6.0	14.3	9.0	9.0	12.9
GUROBI	max gap	1.0	3.5	1.1	14.9	22.5	28.3	14.4	24.2
	# best sol	9	10	10	7	3	9	7	6
	avg gap	7.2	11.4	5.2	9.9	13.9	20.2	11.4	13.4
GR	max gap	10.9	13.6	7.5	13.5	17.1	23.3	15.6	20.4
	# best sol	0	0	0	0	1	0	1	1
	avg gap	7.7	11.8	5.6	10.1	14.1	20.5	11.6	13.7
GRR	max gap	11.7	13.8	7.9	14.1	17.5	23.6	16.0	20.5
	# best sol	0	0	0	1	0	0	0	0
	avg gap	0.9	5.1	1.8	9.8	13.7	20.2	11.1	12.7
TabuWithLP	max gap	1.7	6.8	3.5	12.9	16.9	23.1	15.5	18.9
	# best sol	1	0	0	2	6	1	2	3

Table 3. Computational results for data set 2 with running time limited to 5 min.

		1	2	3	4	5	6	7	8
	avg gap	2.1	5.7	1.3	10.8	20.3	30.9	16.4	15.8
GUROBI	max gap	4.3	9.7	4.6	19.8	23.6	38.4	21.2	25.6
	# best sol	10	8	10	9	1	3	3	4
	avg gap	8.8	12.2	8.2	14.2	19.1	30.6	15.4	14.9
GR	max gap	12.7	24.3	14.4	23.1	22.7	36.8	20.5	22.6
	# best sol	0	0	0	0	1	1	0	0
	avg gap	9.6	12.6	8.7	14.7	19.2	30.9	15.7	15.1
GRR	max gap	14.1	24.4	14.9	23.4	22.8	37.3	20.9	22.8
	# best sol	0	0	0	0	2	1	0	0
	avg gap	7.9	10.9	7.9	14.5	19.0	30.3	15.4	14.9
TabuWithLP	max gap	11.0	23.0	13.9	21.4	23.3	37.4	20.5	22.6
	# best sol	0	2	0	1	6	5	7	6

Table 4. Computational results for data set 2 with running time limited to 30 min.

		1	2	3	4	5	6	7	8
	avg gap	0.5	3.0	0.0	1.2	11.0	25.0	16.4	15.8
GUROBI	max gap	1.7	4.6	0.0	4.6	23.6	38.4	21.2	25.6
	# best sol	8	8	10	10	5	3	0	1
	avg gap	6.6	12.0	4.7	10.3	15.1	25.6	12.0	12.8
GR	max gap	9.2	18.3	8.5	18.0	19.7	31.4	16.1	18.9
	# best sol	0	0	0	0	2	1	0	3
	avg gap	6.8	12.1	5.2	11.1	15.6	26.4	12.4	13.3
GRR	max gap	9.5	18.3	8.9	19.4	20.2	31.7	16.1	19.2
	# best sol	0	0	0	0	0	0	1	1
	avg gap	0.7	6.0	1.7	10.1	15.1	25.5	11.8	12.9
TabuWithLP	max gap	1.3	12.3	4.2	17.6	19.5	31.3	16.0	21.4
	# best sol	2	2	0	0	3	6	9	5

makes it a suitable case to be solved by matheuristic algorithms. The knowledge attained in tabu list is learned and utilized at the subsequent steps. Computational analysis prove the effectiveness of our approach. However, there are several possible ways to try so as to improve the performance. Firstly, in this approach, we only considered single job movement. This is because, algorithm based on multiple jobs movement did not dominate other algorithms from previous study. The lack of an effective way to search neighborhood involving changing of starting points of multiple jobs may be the main reason. Information from linear programming solver is to be explored further in aiding this kind of decisions. Secondly, the tabu search routine designed in this paper is rather simple and elementary, more careful tailoring on parameter settings are to be analyzed.

References

1. Boland, N., Kalinowski, T., Waterer, H., Zheng, L.: Scheduling arc maintenance jobs in a network to maximize total flow over time. Discrete Applied Mathematics. 163, 34-52(2014)
2. Boland, N., Kalinowski, T., Waterer, H., Zheng, L.: Mixed integer programming based maintenance scheduling for the Hunter Valley coal chain. Journal of Scheduling. 16, 649-659(2013)
3. Boland N., Savelsbergh, W.M.P. : Optimizing the Hunter Valley coal chain. In: Gurnani, H., Mehrotra, A., Ray, S. (eds.) Supply chain disruptions: theory and practice of managing risk, Springer-Verlag Ltd., London (2011)
4. Boland N., Kalinowski T., Kapoor R., Kaur. R.: Scheduling unit time arc shutdowns to maximize network flow over time: complexity results. Networks, 63 (2) (2014), pp. 196-202.
5. Ford, L.R., Fulkerson, D R.: Flows in Networks, Princeton University Press, Princeton, NJ, 1962.
6. Koch R., Nasrabadi E., Skutella M.: Continuous and discrete flows over time. Mathematical Methods of Operations Research vol. 73, No. 3, pp301-337, 2011.
7. Skutella M.: An introduction to network flows over time. In: Cook W., Lovasz L., Vygen J. Research trends in combinatorial optimization. Berlin: Springer; 2009. pp451-482.
8. Pinedo, M.: Scheduling: Theory, Algorithms, and Systems. Springer, 3rd edition, 2008.
9. Maniezzo V., Stützel T., Voß S.: Matheuristics: hybridizing metaheuristics and mathematical programming. Springer 2009.
10. Ribeiro C.: Sports scheduling: Problems and applications. International transactions in operations research. 19, 201-226(2012)
11. Voß, S., Fink, A.: A hybridized tabu search approach for the minimum weight vertex cover problem. Journal of Heuristics. 18, 869-876(2012)
12. Pellegrini, P., Mascia, F., Stützle, T., Birattari, M.: On the sensitivity of reactive tabu search to its meta-parameters. Soft Computing. 18, 2177-2190(2013)
13. Glover F.: Tabu search – Part I. ORSA J Comput. 1989; 1:190 - 206.
14. Glover F.: Tabu search – Part II. ORSA J Comput. 1990; 2:4 - 32.

A Matheuristic Approach for Solving the Edge-Disjoint Paths Problem

Bernardo Martín, Ángel Sánchez,
Cesar Beltran-Royo and Abraham Duarte

Dept. Informática y Estadística, Universidad Rey Juan Carlos
b.martinn@alumnos.urjc.es, angel.sanchez@urjc.es,
cesar.beltran@urjc.es, abraham.duarte@urjc.es

Abstract. In this paper, we introduce a matheuristic procedure for the Edge-Disjoint Paths (EDP) problem. This $\mathcal{NP}$-hard optimization problem has applications in real-time communications, VLSI-design, scheduling, bin packing, or load balancing. The proposed approach hybridizes an Integer Linear Programming formulation of the problem with a Particle Swarm Optimization strategy. Empirical results using 100 previously reported instances show that the proposed procedure compares favorably to previous metaheuristics for this problem, such as Ant Colony Optimization and Multi-Start Greedy Algorithm. We finally confirm the significance of the results by using non-parametric statistical tests.

1 Introduction

There exists a wide variety of network problems where several connection requests occur simultaneously [11]. In general, each request is attended by finding a route in the corresponding network, where the origin and destination of such a route are those hosts that wish to establish a connection for information exchange. As it is well documented in the related literature, the exchange of information through disjoint routes increases the effective bandwidth, velocity and the probability of receiving the information. Simultaneously, congestion and possible failures are reduced (see [2][4] for further details). The definition of disjoint paths may refer to nodes, edges, or both. As far as we know, the most studied variant is the one where disjointness implies not to share edges. This optimization problem is usually known as the maximum edge-disjoint paths (EDP) problem.

In communication networks, a node may either be a data communication equipment (modems, hubs, bridges, or switches) or a data terminal equipment (telephone, printer, or host computer). A network is usually modeled as an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where the set of nodes $\mathcal{V}$ corresponds to terminal/communication equipment and the set of edges $\mathcal{E}$ corresponds to the physical links. Each link $e \in \mathcal{E}$, with $e = (i, j)$ and $i, j \in \mathcal{V}$, has a weight $w(e) \in \mathbb{R}^+$, which usually represents the physical distance between both endpoints. Let $\mathcal{T} = \{[i_k, j_k] \mid i_k \neq j_k \in \mathcal{V}, \ 1 \leq k \leq K\}$ be a set of pairs, called

terminal nodes, that request to be connected with a route (path) in $\mathcal{G}$. The EDP problem then consists of maximizing the number of disjoint paths that connect terminal nodes in $\mathcal{T}$. This $\mathcal{NP}$-hard optimization problem [9] has applications in a large variety of areas such as real-time communications, VLSI-design, scheduling, bin packing, or load balancing, among others.

Figure 1 shows an example with three pairs of terminals $[i_1, j_1], [i_2, j_2]$, and $[i_3, j_3]$, joint with paths represented with solid, dotted, and dashed lines, respectively. As it is easy to check, this is a feasible solution (i.e., each edge is used exclusively in one path) its the value is 3.

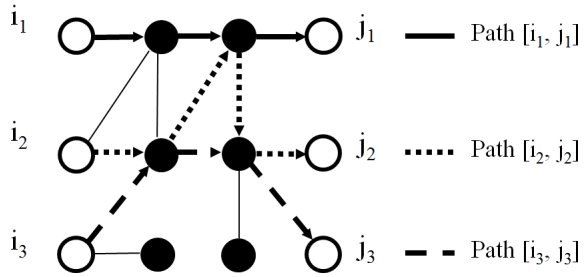


Fig. 1. Example of a solution to the EDP problem.

The EDP problem has been addressed in [5] by means of an Ant Colony Optimization algorithm. The procedure is based on a decomposition of the problem into simpler subproblems. In the computational experience, these authors showed that the proposed method outperformed a multi-start greedy method in both, solution quality and computation time. The same authors improved the previous algorithm in [6]. The new approach incorporated a parallel construction of all paths and the use of candidate list strategies for the exploitation of the promising choices at each construction step. Additionally, the authors improved a local search strategy by considering two different criteria of the objective function, and the partial destruction of the currently best solution as an escape mechanism. In this paper we present a matheuristic approach for the EDP problem. Specifically, the proposed algorithm hybridizes an Integer Linea Programming (ILP) model and a Particle Swarm Optimization (PSO) algorithm. The computational results show that our procedure obtains better performance than previous approaches. The rest of the paper is organized as follows. Section 2 introduces the proposed matheuristic approach devised for this problem. Section 3 describes the computational experience. Finally, the paper ends in Section 4 by drawing the most relevant conclusions.

2 The Matheuristic Solver

The optimization algorithm proposed in this paper is focused on calculating high quality solutions for the EDP. It is composed of an Integer Linear Programming

(ILP) formulation combined with a Particle Swarm Optimization (PSO) algorithm. First, the exact procedure to solve the ILP formulation is run for a fixed computing time. If the optimum value is found in a shorter computing time than the limit, naturally, the corresponding execution is interrupted, returning the optimal value. Otherwise, the exact method returns the best feasible solution found so far. Then, the Particle Swarm Optimization method uses that solution as a seed to guide the optimization strategy.

2.1 Integer linear programming formulation for the EDP problem

In the EDP problem there are, essentially a set of pairs of terminal nodes $\mathcal{T}$ that one wishes to connect by edge-disjoint paths. The EDP problem can be formulated as a Binary Multicommodity Network Flow (BMNF) problem [11]. However, the formulation in [11] requires a number of variables which is exponential in the size of the graph. In this paper we propose a BMNF formulation of the EDP problem which requires a polynomial number of variables in the size of the graph.

Thinking in terms of flow, the EDP problem is focused on sending a single unit of flow from terminal node i_k to terminal node j_k, for all pair $[i_k, j_k] \in \mathcal{T}$. In this way, the commodity-k flow defines a path from node i_k to node j_k associated to the k-th commodity, for all $k \in \mathcal{K} = \{1, \ldots, K\}$.

We have used the notation shown in Table 1. Basically, graph nodes are classified into three groups: supply nodes ($\mathcal{V}^-$), demand nodes ($\mathcal{V}^+$) and transshipment nodes ($\mathcal{V}^0$).

The nodes of the first group are labeled by the commodity index k, and the nodes of the second group are labeled by $K + k$. With this notation, each pair or terminal nodes $[i_k, j_k]$ is labelled as $[k, K + k]$.

In this model, variable x_{ijk} represents commodity-k flow in edge (i, j).

Parameter	Values	Description	
$\mathcal{V}$	$\{1, \ldots, K, \ldots, V\}$	Set of nodes	
$\mathcal{V}^-$	$\{1, \ldots, K\}$	Supply (source) nodes	
$\mathcal{V}^+$	$\{K+1, \ldots, K+K\}$	Demand (sink) nodes	
$\mathcal{V}^0$	$\mathcal{V} \setminus (\mathcal{V}^- \cup \mathcal{V}^+)$	Transshipment nodes	
i, j, v	-	Indexes for nodes	$i, j, v \in \mathcal{V}$
$\mathcal{E}$	-	Set of edges	
(i, j)	-	Index pair for edges	$(i, j) \in \mathcal{E}$
$\mathcal{G}$	$(\mathcal{V}, \mathcal{E})$	Undirected graph	
$\mathcal{K}$	$\{1, \ldots, K\}$	Set of *commodities*	
k	-	Index for *commodities*	$k \in \mathcal{K}$
x_{ijk}	$\{0, 1\}$	Commodity-k flow in edge (i, j)	$(i, j) \in \mathcal{E},\ k \in \mathcal{K}$

Table 1. Notation for the ILP formulation

We propose the following ILP formulation for the EDP problem (in Section 2.2 we will show its validity):

$$\max_{x} \quad z_{EDP} = \sum_{v \in \mathcal{V}^-} \sum_{(v,j) \in \mathcal{E}} x_{vjv} \tag{1}$$

$$\text{s.t.} \quad \sum_{(v,j) \in \mathcal{E}} x_{vjv} \leq 1 \qquad v \in \mathcal{V}^- \tag{2}$$

$$\sum_{(i,v) \in \mathcal{E}} x_{ivv} = 0 \qquad v \in \mathcal{V}^- \tag{3}$$

$$\sum_{(i,v) \in \mathcal{E}} x_{ivk} - \sum_{(v,j) \in \mathcal{E}} x_{vjk} = 0 \qquad v \in \mathcal{V}^-, \quad k \in \mathcal{K} \backslash \{v\} \tag{4}$$

$$\sum_{(i,v) \in \mathcal{E}} x_{ivk} - \sum_{(v,j) \in \mathcal{E}} x_{vjk} = 0 \qquad v \in \mathcal{V}^0, \quad k \in \mathcal{K} \tag{5}$$

$$\sum_{(i,v) \in \mathcal{E}} x_{ivk} \leq 1 \qquad v \in \mathcal{V}^+, \quad k = v - K \tag{6}$$

$$\sum_{(v,j) \in \mathcal{E}} x_{vjk} = 0 \qquad v \in \mathcal{V}^+, \quad k = v - K \tag{7}$$

$$\sum_{(i,v) \in \mathcal{E}} x_{ivk} - \sum_{(v,j) \in \mathcal{E}} x_{vjk} = 0 \qquad v \in \mathcal{V}^+, \quad k \in \mathcal{K} \backslash \{v - K\} \tag{8}$$

$$\sum_{k \in \mathcal{K}} x_{ijk} + \sum_{k \in \mathcal{K}} x_{jik} \leq 1 \qquad (i,j) \in \mathcal{E}, \quad i < j \tag{9}$$

$$x_{ijk} \in \{0,1\} \qquad (i,j) \in \mathcal{E}, \quad k \in \mathcal{K}, \tag{10}$$

where the following comments are in order:

- Eq. (1): We maximize the total flow sent from source nodes. In Section 2.2 we will see that this is equivalent to maximize the number of edge-disjoint paths.
- Eq. (2): Each source node v can sent at most one unit of commodity-v flow.
- Eq. (3): Each source node v cannot receive commodity-v flow. This avoids the generation of cycles beginning and ending at source nodes.
- Eq. (4): At each source node v and for each commodity k, the flow supply is equal to the flow demand, for all $k \in \mathcal{K} \backslash \{v\}$.
- Eq. (5): At each transshipment node and for each commodity, the flow supply is equal to the flow demand.
- Eq. (6): Each sink node v can receive at most one unit of commodity-$(v-K)$ flow.
- Eq. (7): Each sink node v cannot send commodity-$(v-K)$ flow.
- Eq. (8): At each sink node v and for each commodity k, the flow supply is equal to the flow demand, for all $k \in \mathcal{K} \backslash \{v - K\}$.
- Eq. (9): The joint flow capacity of each edge is one. Notice that commodity flows can use each edge $(i,j) \in \mathcal{E}$ in both directions.

2.2 Validity of the ILP formulation for the EDP problem

The validity of formulation (1)-(10) for the EDP problem can be seen as follows. First, let us see that, given a source node $\bar{v}$ and an edge $(\bar{v}, j)$, if we have $x_{\bar{v}j\bar{v}} = 1$ then there exist one path connecting the pair of terminal nodes $[\bar{v}, K + \bar{v}]$. The details are as follows: The equality $x_{\bar{v}j\bar{v}} = 1$ means that the source node $\bar{v}$ sends a unit of commodity-$\bar{v}$ flow to node j. Equations (4), (5) and (8) enforce that all the nodes of the graph behave as transhipment nodes for commodity $\bar{v}$, except the source node $\bar{v}$ and the corresponding sink node $K + \bar{v}$. The number of edges is finite and each edge admits, at most, one unit of flow. Therefore, this flow transmission initiated at source node $\bar{v}$ will necessarily end up at sink node $K + \bar{v}$. As a consequence, we will have $x_{ivk} = 1$ for some $i \in \mathcal{V}$, for $v = K + \bar{v}$ and for $k = \bar{v}$. The commodity-$\bar{v}$ flow just described defines a path connecting the pair of terminal nodes $[\bar{v}, K + \bar{v}]$.

Second, let us see that, given two source nodes $\bar{u}$ and $\bar{v}$, if we have $x_{\bar{u}j\bar{u}} = x_{\bar{v}j\bar{v}} = 1$ then the paths connecting the corresponding pairs of terminal nodes are disjoint. This is directly enforced by Equation (9).

Let as define EDP^* as the maximum number of edge-disjoint paths associated to a pair $(\mathcal{G}, \mathcal{T})$ and let us consider z^*_{EDP}, the optimal value of the corresponding ILP formulation (1)-(10). So far we have shown that $z^*_{EDP} \leq EDP^*$.

Third, let us see now that $EDP^* \leq z^*_{EDP}$. This is clear, since given a pair of terminal nodes, say $[\bar{v}, K + \bar{v}]$ connected by an edge-disjoint path P, it is possible to send one unit of commodity-$\bar{v}$ flow from node $\bar{v}$ to node $K + \bar{v}$ through P. In the ILP formulation (1)-(10), this flow corresponds to set equal to one some variables, in particular $x_{\bar{v}i\bar{v}} = 1$ for some edge $(\bar{v}, i) \in \mathcal{E}$, which implies $EDP^* \leq z^*_{EDP}$ as we wanted to see.

All in all, we have seen seen that $EDP^* = z^*_{EDP}$. Since by construction, the ILP formulation (1)-(10) generates a set of edge-disjoint paths with cardinality z^*_{EDP}, we conclude that it is a valid formulation of the EDP problem.

2.3 Particle Swarm Optimization for the EDP problem

Particle Swarm Optimization (PSO) is a population-based stochastic optimization technique introduced in [10]. This procedure is inspired by the social behavior of bird flocking or fish schooling. From an algorithmic point of view, in the PSO framework there exists a set of particles or agents that search for good solutions to a given optimization problem. Each agent represents a candidate solution to the corresponding optimization problem and it tries to reach the best solution found so far. Additionally, each agent is also influenced by the best value obtained so far by any particle in its neighborhood. The main idea behind PSO is to change, at each iteration, the velocity of (accelerating) each particle toward its local and global best found solutions. In order to diversify the search, the acceleration is weighted by using a random term.

In this paper, we adapt this methodology to the EDP problem. The algorithm starts by constructing a predetermined number of shortest paths between each pair of terminals by using the algorithm described in [17]. More precisely, given

a pair of terminal nodes $[i_k, j_k] \in \mathcal{T}$ (with $1 \leq k \leq K$), we construct λ paths from i_k to j_k and store them in the set $SP(k, \lambda)$, where we k refers to the specific pair of terminals. We assume that the paths are sorted in ascending order (from the shortest to the largest). Once we have finished this preprocessing step, the algorithm explores the set of terminals at random. The first pair is connected with the shortest path in its corresponding set. The rest of pairs of terminals are connected as follows. We first try to join the pair with the shortest path. If we succeed, we move to the next path. If it is not possible (since there is, at least, one common edge previously selected), the method keeps selecting paths until it finds an disjoint path or the set of λ shortest paths has been completely explored. The method ends when all pair of terminals have been explored, returning the solution as a set of disjoint paths.

The aforementioned procedure constructs a single solution. Then, the proposed PSO method uses it to create a population P of different solutions. At the beginning of the optimization process, there is not information about previous iterations. Therefore, the best local solution is set to the current value of each particle. Similarly, the best global solution is selected between the best one in P and the solution obtained with the ILP.

Similarly to an evolutionary computation technique, PSO maintains a population of particles, where each particle stores a potential solution to an optimization problem. Therefore, each particle can be evaluated with the corresponding objective function. In the context of PSO the current value of the particle is called inertial term. In general, particles evolve during the search process by means of the modification of the solution store in each particle. In order to guide the search, each particle also stores the best local solution found by each particule (cognitive term), and the best global solution (social term) selected among all particles and iterations.

In the particular case of the EDP problem, we propose to modify the solution stored in each particle by adding/dropping paths. More precisely, given a particle in iteration t, PSO produces a new particle in iteration $t+1$ by connecting each pair of terminals using the following equations:

$$path_{iter} = \text{rand}\left(\varphi_{iner} \cdot path_{iner}, \varphi_{cogn} \cdot path_{cogn}, \varphi_{soc} \cdot path_{soc}\right) \tag{11}$$

$$\varphi_{iner} \in [0, 1] \qquad \text{inertial weight} \tag{12}$$

$$\varphi_{cogn} \in [0, 1] \qquad \text{cognitive weight} \tag{13}$$

$$\varphi_{soc} \in [0, 1] \qquad \text{social weight} \tag{14}$$

where φ_{iner}, φ_{cogn}, φ_{soc} are the weight associated to each component and $path_{iner}$, $path_{cogn}$, $path_{soc}$ represent, respectively, the value of the objective function of the current, best local and best global solutions stored in each particle. We us illustrate this random selection with an example. Let us assume that $\varphi_{iner} = \varphi_{cogn} = \varphi_{soc} = 1/3$; then, inertial, cognitive and social components would have the same weight. Similarly, let us assume that the value of the current particle is $path_{iner} = 10$, best local is $path_{cogn} = 12$, and best global is $path_{cogn} = 15$. In order to decide which path is selected we use a typical roulette

wheel strategy by computing the cumulative probability (i.e, 10+12+15 = 37) and generate a random number between 0 and 37. If this number is lower than 10, we keep the current path to joint the corresponding pair of terminals. If that number is larger than 10 but lower than 22, the pair of terminals is joint with the path stored in the best local solution. Finally, if the random number is larger than 22 but, we use the path considered in the best global solution.

The aforementioned procedure is maintained until it reaches a maximum computing time, returning the best solution found during the search.

3 Experimental Evaluation

This section reports the computational experiments that we have performed for testing the efficiency of the proposed matheuristic procedure for solving the EDP problem. The ILP was executed with CPLEX (version 12.2) while the PSO was implemented in Java JR6 6. All the the experiments were conducted on a Pentium(R) Dual-Core CPU at 3.20GHz, and 4GB of RAM memory. We have considered a set of instances previously used in this problem [6]. Specifically, this set contains the following families of graphs: `graph3` (164 vertices and 370 edges), `graph4` (434 vertices and 981 edges), `AS.BA.R-Wax.v100e217` (100 vertices and 217 edges), `bl-wr2-wht2.10-50.sdeg` (500 vertices and 1020 edges), and `mesh 25x25` (625 vertices and 1200 edges). Each family contains 20 graphs and the number of terminals ranges from $0.10 \cdot V$ to $0.40 \cdot V$ and the weight of each edge is 1.

We have divided this section into two different parts: preliminary experimentation and final experimentation. The preliminary experiments are performed to set the values of the key search parameters of the proposed method. We consider a representative subset of instances with different sizes and densities to perform these experiments. This strategy avoids over-fitting and allows us to extract conclusions about the robustness and effectiveness. For the sake of brevity, we only report the best values found for each parameter in Table 2.

Parameter	Value	Description
$t_{MP} = \rho \cdot V \cdot K$	$\rho = 0.0012$	Limit of execution time for the ILP
$t_{HA} = \rho \cdot V \cdot K$	$\rho = 0.0012$	Limit of execution time for the PSO
λ	100	Number of shortest path
$particles$	100	Number of particles
φ_{iner}	0.2	PSO inertial parameter
φ_{cogn}	0.4	PSO cognitive parameter
φ_{soc}	0.4	PSO social parameter

Table 2. Parameters Tuning

Once we have adjusted the key search parameters, we compare our procedure, PSO+ILP, with the best previous method identified in the state of the art [6].

In particular, an Extended Artificial Ant Colony (Ext. ACO) and a Multi-start Simple Greedy (MSGA). For the sake of completeness, we include the results obtained with the metaheuristic (PSO) and exact (ILP) approaches. Table 3 reports average results for the whole set of 100 instances. The columns of this table show: z_{EDP}, average quality over all instances; Dev. (%), average percent deviation with respect to the best solution found in the experiment; and Time, average computing time in seconds required by the procedure.

Table 3 shows that the proposed procedure clearly outperforms the other previous two methods (including the best algorithm found in the state of the art) in both, average objective function and average percentage deviation. Moreover, it only needs a fifth of the computing time used by the other methods to produce better results. Therefore, results reported in this table suggest the superiority of the matheuristic approach over the other two methods. It is worth mentioning that the results obtained by each method (either PSO or ILP) executed as an independent algorithm for the same computing time than the matheuristic approach does not present a remarkable performance. Indeed, PSO and ILP obtain even worse results than the best previous algorithm. In our opinion, this fact justifies the hybridization between both procedures.

Algorithm	z_{EDP}	Dev(%)	Time (s)
MSGA	34.73	15.27	544.47
Ext ACO	37.82	10.07	537.45
ILP	29.16	20.75	45.19
PSO	35.58	14.52	97.37
ILP+PSO	39.10	6.52	89.50

Table 3. Summary of Results

We applied the non-parametric Friedman test for multiple correlated samples to the best solutions obtained by the five compared methods. This test computes, for each instance, the rank value of each method according to solution quality (where rank 1 is assigned to the best method and rank 5 to the worst one). Then, it calculates the average rank values of each method across all the instances solved. If the averages differ greatly, the associated p-value or significance will be small. The resulting p-value of 0.0001 (considering a level of significance of 0.05) obtained in this experiment clearly indicates that there are statistically significant differences among the five methods tested. Specifically, the rank values produced by this test are 1.62 (ILP+PSP), 2.15 (Ext. ACO), 2.23 (ILP), 2.77 (PSO), and 3.23 (MSGA).

4 Conclusions

In this paper, we propose a matheuristic procedure based on the hybridization between an Integer Linear Programming (ILP) technique and a Particle Swarm

Optimization (PSO) algorithm to deal with the Edge-Disjoint Paths (EDP) problem. We provide an extensive experimental comparison among our procedure and the best previous methods for this problem in the state of the art. We performed an extensive computational testing over a set of 100 instances previously used. Experimental results show that the proposed algorithm outperforms the best methods identified. We also proved statistical tests to confirm the significance of the obtained results, thus emerging the matheuristic approach as the best algorithm in terms of quality and computing time.

Of particular interest in our work has been testing the combination of exact and approximate procedures. Through extensive experimentation, we have been able to determine the benefits of this combination. We believe that our findings can be translated to other combinatorial problems and it will help in the development of more elaborated matheuristics methods.

Acknowledgements

This research has been partially supported by the Spanish Ministry of "Economía y Competitividad", with grants ref. TIN2015-65460-C2-2-P, MTM2015-63710-P, TIN 2014-57458-R and "Comunidad de Madrid" with grant ref. S2013/ICE-2894.

References

1. Bang-Jensen, J. and Guting, G.: Digraphs: Theory, Algorithms and Applications. Springer Monographs in Mathematics. Springer. 2001.
2. Barabasi, A., Albert, R.: Emergence of scaling in random networks. Science 286, 509512 (1999)
3. Bazargan, M.: Airline Operations and Scheduling. Ashgate. 2007.
4. Baveja, A., Srinivasan, A.: Approximation algorithms for disjoint paths and related routing and packing problems. Math. Oper. Res. 25(2), 255280 (2000)
5. Blesa, M.J. and Blum, C.: Ant colony optimization for the maximum edge-disjoint paths problem. In 1st European Workshop on Evolutionary Computation in Communications, Networks, and Connected Systems (EvoCOMNET'04), vol. 3005 of Lecture Notes in Computer Science, pags 160-169. Springer-Verlag. 2004.
6. Blesa, M.J. and Blum, C.: Finding edge-disjoint paths in networks by means of artificial ant colonies. Journal of Mathematical Modeling and Algorithms. 2007.
7. Blum, C. and Roli, A.: Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison. ACM Computing Surveys, Vol 35, No 3, pp 260-308. 2003. Edge-disjoint paths and unsplittable flow. Handbook of Approximation Algortithms and Metaheuristics, 57–1. 2007.
8. Eberhart, R.C. and Kennedy, J.: A new optimizer using particle swarm theory. Proceedings of the Sixth International Symposium on Micromachine and Human Science, Nagoya, Japan. pp. 39-43, 1995.
9. Karp, R.: Complexity of Computer Computations. Miller R.E. and Thatcher J.W. Plenum Press, New York, 1972.
10. Kennedy, J. and Eberhart, R. C. A discrete binary version of the particle swarm algorithm. Proceedings of the World Multiconference on Systemics, Cybernetics and Informatics 1997, Piscataway, NJ. pp. 4104-4109, 1997

11. Kolliopoulos, S.G.: Edge-disjoint paths and unsplittable flow. Handbook of Approximation Algortithms and Metaheuristics, 57–1. 2007.
12. Kramer, M., and van Leeuwen, J.: Advances in computing research, volume 2. JAI Press, 1994.
13. Marx, D.: Eulerian disjoint paths problem in grid graphs is NP-complete. Discrete Applied Mathematics, 2004.
14. Middendorf, M. and Pfeiffer F.: On the complexity of the disjoint path problem. Combinatorica, 13(1), 1993.
15. Nishizeki, T., Vygen, J. and Zhou, X.: The edige disjoint paths problem is np-complete for series-paralell graphs. Discrete Applied Mathematics, 2001.
16. Vygen J.: NP-completeness of some edge-disjoint paths problems. Discrete Applied Mathematics, 61(1), 1995.
17. Yen, Jin Y.: An algorithm for finding shortest routes from all source nodes to a given destination in general networks, Quarterly of Applied Mathematics 27: 526530, 1970.

A Matheuristic for the VRP with Limited Traffic Zones

Simona Mancini

Dipartimento di Matematica e Informatica, Universitá di Cagliari, Cagliari, Italy

Abstract. This paper addresses the Vehicle Routing Problem with Mixed Fleet and Limited Traffic Zones (VRPLTZ), which is an extension of the standard Vehicle Routing Problem (VRP), in which the fleet is composed by both electric (EV) and traditional (TV), vehicles. The customers area is split in a Limited Traffic Zone (LTZ) and a Free Access Zone (FAZ). Customers within the FAZ may be visited by all the vehicles at any time. TV cannot access the LTZ within a given time window while EV can access it at anytime. Due to their limited autonomy, EV have a restriction on the maximum route length, while no length restriction are imposed to routes covered by TV. A maximum route duration is imposed for all the routes. A Mixed Integer Programming formulation and a Large Neighborhood Search Based Matheuristic are proposed. Computational results reported show the efficiency and effectiveness of the proposed approach.

1 Introduction

In the last decades, an increasing attention has been given to City Logistics aspects, [21]. Furthermore, the greenhouse effect has become a hot political topic throughout the world and laws and regulations have been adopted to reduce pollution emissions in all of the highly developed countries. Such political decisions have had an important effect on the logistics industry. Transportation companies are requested to provide more competitive services in a more sustainable way. This goal can be reached both through a better exploitation of available resources and through the introduction of new technologies. A better exploitation of currently available resource could be obtained by applying more efficient and sophisticated routing planning optimization methods and adopting smart distribution systems, as stated in [15], which would yield a decrease of traveled distance (that means a reduction of operational cost for the company) and, consequently, a reduction of emissions. However, this generally results in a decline of emissions of only few percentage points. To actually increase logistics operations sustainability, a more promising strategy is the use of vehicles with electric propulsion. Although this technology is very attractive from a sustainability point of view, the usage of Electric Vehicles (EV) in practical applications is very limited due to several restrictions they introduce. In fact, EVs have a very limited autonomy and battery recharging stations are not widespread along the road network. Therefore, visits to recharging points should be a priori planned during the route construction phase in order to avoid remaining stuck without

possibility to recharge the battery. Furthermore, recharging operations are not immediate as fuel replenishment, but can last for hours. Last but not least, due to their innovative technology, purchasing cost of EV are sensibly higher if compared with traditional fuel ones. For all the above reasons, the total replacement of the fleets with EVs by the transportation companies, is unrealistic. Nevertheless, a winning strategy to balance profit maximization and environmental issues, is to integrate the current fleet with few electric vehicles exploiting their advantages to construct smarter routing planning.Public administrations have adopted many measures to regulate the traffic within the urban area. One of the most used is the introduction of Limited Traffic Zones (LTZ) the access to which is allowed only during certain periods of the day. Furthermore, in order to promote the use of environmental friend vehicles, such the electric ones, LTZ limitations do not apply to this kind of vehicles, which can freely travel in this protected areas. Optimizing a routing plane with this kind of constraints is not a trivial issue and the standard VRP variants studied in the literature do not allow to handle these kinds of features. Therefore, new problems must be defined and new models must be developed. The paper is organized as follows. Section 2 is devoted to a literature review on related problems. In Section 3 a formal description of the problem addressed in this paper is given, while in Section 4 a Mixed Integer Programming formulation is reported. A Large Neighborhood Search Matheuristic is presented in Section 5 and computational results are reported in Section 6. Finally, Section 7 deals with conclusions and future developments.

2 Literature Review

Recently, an increasing attention has been paid to Green Logistics, which involves the integration of environmental aspects in logistics. Many papers concerning Operations Research applications to Green Logistics have been proposed in the literature. A wide set of issues has been addressed, such as intermodal transportation, mode, fleet and fuel choice, and smart distribution systems, such as the Multi-Echelon ones, [15]. For a complete survey on Green Logistics issues and challenges, the readers may refer to [6].

One emerging research area concerns pollution emission minimization. In [3], the authors introduced the Pollution-Routing Problem (PRP), an extension of the classical Vehicle Routing Problem with Time Windows, which consists in routing a number of vehicles to serve a set of customers, and determining their speed on each route segment in order to minimize a function that includes fuel, emission and driving costs. The same problem has been addressed in [7] where an Adaptive Large Neighborhood Search based heuristics approach is proposed. A time-dependent version of the PRP has been addressed in [11], while [8] introduced the bi-objective pollution routing problem, in which two conflicting objectives are addressed: driving time and fuel consumption. The usage of electric vehicles is, currently, the most sustainable solution to this issue. However, due their limited autonomy, EV are more popular for the movements of in-city

goods than for medium-long range freight transport. In order to compensate for their short range, a dense power re-supply network would need to be set-up, possibly in conjunction with the possibility of batteries swapping. Unfortunately, the present re-supply network is still very limited. Only a few papers have actually addressed VRPs with alternative fuel vehicles. In [13], the authors dealt with a VRP concerning Pickup and Delivery (VRPPD) with a mixed fleet composed by both electric and traditional vehicles. They considered time and capacity constraints but they simply added an extra time for recharging the electric vehicles batteries, when needed. In this extra time is implicitly included time to reach the nearest recharging station and recharging time. This is a too strong simplification of the problem, since they did not explicitly consider neither recharging stations location nor the possibility of remaining stuck in the network without autonomy to reach a refueling station. This type of model may be used under the hypothesis that In [5] the recharging vehicle routing problem (RVRP) was introduced, in which vehicles were allowed to recharge directly at customer locations, adding a time penalty to the route duration. Erdogan and Miller-Hooks, [9] proposed a VRP with the possibility of refueling a vehicle at a station along the route, named Green Vehicle Routing Problem (GVRP). They considered a limited number of refueling infrastructures located along the network and a fixed recharging time (independent of the remaining level of battery charge). They also impose a maximum route duration. Fuel is consumed with a given rate per traveled distance and they assume that the tank is totally replenished at recharging stations. In [19], the authors proposed the Electric Vehicle Routing Problem with Time Windows and Recharging Stations (E-VRPTW), which can be seen as an extension of the GVRP in which time windows are considered. A further extension of the E-VRPTW, where partial recharging are allowed, has been studied in [4], while multiple recharging technologies, characterized by different recharging times and costs, have been introduced in [10]. In [16], the Hybrid Vehicle Routing Problem has been proposed, an extension of the GVRP in which vehicles may switch from electric to traditional propulsion at anytime. Kilometers traveled in traditional mode have a cost much higher than those traveled in the electric one.

The Heterogeneous Fleet Vehicle Routing Problem (HVRP) is an extension of the VRP, broadly addressed in the literature, in which the fleet is composed by different types of vehicles. In the HVRP customers are served by a heterogeneous fleet of vehicles with various capacities, fixed usage costs, and variable cost per distance unit, as stated in [12]. An extensive literature review on HVRP is reported in [2]. A HVRP with electric vehicles has been introduced in [14]. In the HVRP defined in the literature, vehicles differ among each others only for cost and capacity but they have to respect all the same constraints (i.e. time windows restriction, maximum route length and duration). On the contrary, in the VRPLTZ, some constraints must be respected only by vehicles belonging to a given category. In fact, time windows constraints hold only for traditional vehicles, route length limitation only for electric ones and route duration limi-

tation for all the vehicles. This issue has never been addressed in the literature before.

3 Problem Definition

The VRPLTZ consists into visiting a set of customers I, starting from a single depot, with a fleet composed by two categories of vehicles, traditional (TV) and electric (EV) ones. A maximum number of vehicles for each category, N_t and N_e is imposed. A maximum route length, D_{max}, is imposed for EV, while no length restrictions are applied to routes performed by TV. A maximum route duration, T_{max} is imposed for every route, basing on driver daily workload limitations. A customer i may be visited at anytime by the EV, while it cannot be accessed by TV within the time window $[a_i, b_i]$. This representation allows also to describe Limited Traffic Zones (LTZ) situations in which customer within the LTZ have access restrictions while customers outside the LTZ have free access at anytime for all the vehicles. In fact, in this case, it is sufficient to impose $a_i = 0$ and $b_i = 0$, defining a dummy access restriction time window, for all the customers outside from the LTZ. For each customer i is known the service time p_i. The fleet is supposed to be available, therefore no purchasing costs are considered. EV and TV have different distance unitary cost, C_e and C_t, whit $C_t > C_e$. For each pair of customers i and j is known the distance d_{ij} and the time necessary to go from i to j, t_{ij}. For each customer i is also known the distance from the depot d_{0i} and the time necessary to be reached from the depot t_{0i} The problem is asymmetric, therefore d_{ij} may be different from d_{ji}, and consequently, t_{ij} may be different from t_{ji}. The goal of the problem is to minimize the total routing cost, while visiting all the customers and respecting all the constraints.

4 Mathematical Model

In this section is reported a Mixed Integer Programming Model for the VRPLTZ. A summary of the input data and the sets involved in the model is hereby reported:

- I is the set of customers
- Dep is the depot
- $I0 = I \cup Dep$
- N_t is the maximum number of traditional vehicles
- N_e is the maximum number of electric vehicles
- C_t is the kilometric cost for a traditional vehicle
- C_e is the kilometric cost for an electric vehicle
- T_{max} is the maximum duration for each route
- D_{max} is the maximum length for each route performed by an electric vehicle
- t_{ij} is the time necessary to go from node i to node j
- d_{ij} is the distance (expressed in Km) between node i and node j
- p_i is the service time for node i

- a_i and b_i represents, respectively, the beginning and the ending of the denied access time window for customer i
- M and ϵ are a very big and a very small constant, respectively

while the variables involved in the model are:

- X_{ij} is a binary variable that is equal to 1 if arc ij is covered by a traditional vehicle and equal to 0 otherwise
- Y_{ij} is a binary variable that is equal to 1 if arc ij is covered by an electric vehicle and equal to 0 otherwise
- T_i represents the arrival time at node i
- D_i represents the cumulated distance covered by the vehicle when it arrives at node i
- α_i and β_i are *service* variables which allows to describe the constraint imposing the respect of the access denied time window at node i

The mathematical model can be formulated as follows:

$$\min C_t \sum_{i \in I0} \sum_{jinI0} X_{ij} d_{ij} + C_e \sum_{i \in I0} \sum_{j \in I0} Y_{ij} d_{ij} \tag{1}$$

$$\sum_{j \in I} X_{ij} + \sum_{j \in I} Y_{ij} = 1 \quad \forall\, i \in I \tag{2}$$

$$\sum_{j \in I} X_{ji} + \sum_{j \in I} Y_{ji} = 1 \quad \forall\, i \in I \tag{3}$$

$$\sum_{j \in I0} X_{ij} = \sum_{j \in I0} X_{ji} \quad \forall\, i \in I0 \tag{4}$$

$$\sum_{j \in I0} Y_{ij} = \sum_{j \in I0} Y_{ji} \quad \forall\, i \in I0 \tag{5}$$

$$T_j \geq T_i + (t_{ij} + p_j)(X_{ij} + Y_{ij}) - Tmax(1 - (X_{ij} + Y_{ij})) \quad \forall\, j \in I\, \forall\, i \in I0 \tag{6}$$

$$T_j \leq T_{max} - t_{j0} \quad \forall\, j \in I0 \tag{7}$$

$$D_j \geq D_i + d_{ij}(X_{ij} + Y_{ij}) - D_{max}(1 - (X_{ij} + Y_{ij})) \quad \forall\, j \in I\, \forall\, i \in I0 \tag{8}$$

$$D_j \leq D_{max} - d_{j0} + M * \sum_{i \in I0} X_{ij} \quad \forall\, j \in I0 \tag{9}$$

$$\sum_{j \in I} X_{0j} \leq N_t \tag{10}$$

$$\sum_{j \in I} Y_{0j} \leq N_e \tag{11}$$

$$\alpha_j \geq \epsilon * (T_j - a_j) \quad \forall\, j \in I \tag{12}$$

$$\beta_j \geq \epsilon * (b_j - T_j) \quad \forall\, j \in I \tag{13}$$

$$\alpha_j + \beta_j \geq 2 \sum_{i \in I0} Y_{ij} + \sum_{i \in I0} X_{ij} \quad \forall\, j \in I \tag{14}$$

$$X_{ij} \in \{0, 1\} \quad \forall\, j \in I0 \tag{15}$$

$$Y_{ij} \in \{0, 1\} \quad \forall\, j \in I0 \tag{16}$$

$$\alpha_j \in \{0, 1\} \quad \forall\, j \in I \tag{17}$$

$$\beta_j \in \{0, 1\} \quad \forall\, j \in I \tag{18}$$

Constraints (2) and (3) ensure that each customer is visited exactly once. While (4) and (5) imply that if customer i is reached by a TV it cannot be left by an EV and vice versa. Constraint (6) allows to determine the minimum arrival time at node i and constraint (8) determines the cumulated distance covered by the vehicle once it reaches customer i. Constraints (7) and (9) impose restriction on the maximum route duration, and route length (active only for EV vehicles), respectively. A limit on the maximum number of TV and EV used is imposed in (10) and (11), respectively. Constraints (12), (13) and (14) ensure that access restrictions, for traditional vehicles, are respected. Finally, constraints (15)-(18) specify variables domain.

5 A Large Neighborhood Search Matheuristic for the VRPLTZ

The Large Neighborhood Search heuristic (LNS), belongs to the class of heuristics that is known as Very Large Scale Neighborhood search (VLSN) algorithms, as stated in [1]. All VLSN algorithms are based on the observation that searching a large neighborhood results in finding local optima of high quality, and hence a VLSN algorithm may return better solutions. However, searching a large neighborhood may be extremely time consuming, therefore various filtering techniques are used to limit the search. In VLSN algorithms, the search is usually restricted to a subset of the solutions belonging to the neighborhood that can be searched efficiently, or to a sample subsets of solutions. LNS is based on the concept of *ruin and recreate*, introduced in [20], in which at each iteration, the solution is partially destroyed and repaired aiming to obtain an improving solution. The

neighborhood in LNS is implicitly defined by the moves used to destroy and repair an incumbent solution. For a detailed survey on LNS the reader may refer to [18].

The destroy operators may be defined in different ways. For routing problems, a destroy operator could consist in removing k elements, (arcs, edges or nodes), from the incumbent solution. The rule according to which these elements are selected may be random, randomized, or deterministic. Deterministic destroy operators, select the currently worst elements in the solution, and therefore they have a greater chance to improve the solution, on the other hand, given their deterministic nature, if applied several times on the same solution, they produce the same results; this means that they have a greater chance to remain stuck in a local minimum. Random and randomized operators main advantage is that even if applied n times on the same solution, they could, potentially, produce n different results. This means that they are able to widely explore the solution space and therefore they have a greater chance to overcome local minima. Generally, a set of destroy operators, among which to chose the one to apply, are defined, including both deterministic and random (or randomized) ones.

Repair methods rebuild a feasible solution starting from the partially destroyed one. Generally, a greedy construction heuristic is used to rebuild the solution. This is very fast but not always very accurate, since only a sample solution is analyzed in the neighborhood. In fact, simple repair operators are fast but not very accurate, while more complex operators obtain better solutions, because they analyze a larger subset in the neighborhood, but they are sensibly slower. If the neighborhood are large, is impossible to explicitly analyze all the solutions within them in reasonable computational times. In the algorithm proposed in this paper, the Large Neighborhood Search is exploited directly by the model, following the idea presented in [17]. In this way, it is possible to obtain, in small computational times, the local minimum with respect to the considered neighborhood, and this makes the intensification phase of the algorithm more powerful and accurate.

5.1 Algorithm Description

The LNS proposed to address the VRPLTZ works as follows. The algorithm starts from an initial feasible solution obtained running the mathematical model and keeping the best solution obtained within a short time limit. At each iteration, the solution is partially destroyed removing k arcs from it. Being A the set of arcs in the current solution S and A^R the set of removed arcs, we can define $A' = A/A^R$ the set of arcs belonging to the partially destroyed solution S^P. A feasible solution S' is constructed running the model imposing that the new solution contains all the arcs belonging to A^P. This can be ensured adding to the model constraint 19, for each pair of nodes i and $j \in I0$ for which $X_{ij} = 1$ in S and constraint 20 for each pair of nodes i and $j \in I0$, for which $Y_{ij} = 1$ in S.

$$X_{ij} = 1 \tag{19}$$

$$Y_{ij} = 1 \tag{20}$$

This *overconstrained* version of the model is run with a short timelimit and the best obtained solution, S' is kept. If S' is better than S, than S' is taken as new current solution S. The whole procedure ends when the maximum number of iterations $MAXITER$ or the maximum number of iterations without any improvement $MAXNOIMPROVE$ are reached.

The pseudocode of the LNS is reported in Algorithm 1.

Algorithm 1 A LNS for the VRPLTZ

run the model with a time limit equal to TIMELIMIT and take the best found solution S^0 as the initial solution
set the current solution S equal to S^0
repeat
  select a destroy operator ω_*
  select k arcs to be removed from S using ω_*
  for all $i \in I0$ and $j \in I0$ — $X_{ij} = 1$ in S^P **do**
    add constraint $(X_{ij} = 1)$
  end for
  for all $i \in I0$ and $j \in I0$ — $Y_{ij} = 1$ in S^P **do**
    add constraint $(Y_{ij} = 1)$
  end for
  run the model again until the optimal solution is reached
  if the newly obtained solution S' is better than S **then**
    set the current solution S equal to S'
  end if
until maximum number of iterations, $MAXITER$, is reached or no improvement has been obtained for $MAXNOIMPROVE$ iterations

5.2 Destroy Operators

Three different destroy operators have been defined:

- **Random Removal(RR)**: randomly select k arcs to be removed
- **Node Removal(NR)**: randomly select $k/2$ nodes and for each selected node i remove both the ingoing and the outgoing arcs
- **Worst Removal (WR)**: remove the k worst arcs. The fitness of arc ij, F_ij is computed as:

$$F_{ij} = 1 - \frac{d_{ij}}{dmax_j}$$

where

$$dmax_j = \max_{k \in I0} d_{kj}$$

The worse arcs are those with the lowest value of F_{ij}.

The first two operators, RR and NR, are random and randomized, respectively, while the last one, WR is deterministic. Therofere, RR and NR may be used also as a stand alone operator, while WR will easily remain stuck into local minima if used as a stand alone operator, but it would be useful if used alternatively with the others two operators. A selection probability Π_ω is associated to each operator ω such that:

$$\sum_{\omega \in \Omega} \Pi_\omega = 1$$

being Ω the set of destroy operators.

6 Computational Results

The Model and the Matheuristic (MH) have been tested on a set of instances of various sizes and typologies. For all the instances, customers are located in a squared area of size 100x100. The Limited Traffic Zone (LTZ) is represented as a square of size 20x20 centered in the customer location area. The number of customers varies among instances in the range [20,50] and the number of customers within the LTZ varies in the range [5,40]. Customers within the LTZ have an access denied time windows in [0,150], which means that LTZ cannot be entered by TV between 8 a.m and 10:30 a.m, which is a common restriction in many large cities. The maximum length of EV routes, $Dmax$, is equal to 100 for all the instances (and represents standard EV autonomy), while the maximum route duration, $Tmax$ can be equal to 270 (half day plan) or to 540 (full day plan). For each instance is known the number of EV and TV available, N_e and N_t, respectively. The unitary distance cost for TV, C_v is three times greater than the EV one. In Tab. 1 instances features are resumed.

INSTANCE	CUSTOMERS	LTZ CUSTOMERS	Tmax	Dmax	Ne	Nt
ZTL20-5	20	5	270	100	3	3
ZTL20-10	20	10	270	100	3	3
ZTL30-10	30	10	540	100	3	3
ZTL30-15	30	15	540	100	3	3
ZTL30-20	30	20	540	100	3	3
ZTL50-10	50	10	270	100	6	6
ZTL50-20	50	20	540	100	6	6
ZTL50-25	50	25	540	100	6	6
ZTL50-30	50	30	540	100	6	6
ZTL50-40	50	40	540	100	6	6

Table 1. Instances features resume

Computational results obtained running the Model within a timelimit of 10000 seconds are reported in Tab. 2. Column 1 reports the name of the instances, while in columns 2 and 3 are reported the total number of customers and the number of customers within the LTZ zone. Upper bounds, i.e. best solutions

INSTANCE	CUSTOMERS	LB	UB	GAP
ZTL20-5	20	114.37	345.85	66.9%
ZTL20-10	20	105.73	130.24	18.8%
ZTL30-10	30	275.86	427.96	35.5%
ZTL30-15	30	239.7	372.13	35.6%
ZTL30-20	30	112.92	275.38	59.0%
ZTL50-10	50	331.25	616.21	46.2%
ZTL50-20	50	329.61	687.15	52.0%
ZTL50-25	50	319.42	570.59	44.0%
ZTL50-30	50	286.29	582.17	50.8%
ZTL50-40	50	283.85	474.04	40.1%
AVG		239.90	448.17	44.91%

Table 2. Upper and Lower Bounds obtained by the Model within 10000 seconds

	RR		NR		RR+NR		RR+NR+WR	
INSTANCE	BEST	AVG	BEST	AVG	BEST	AVG	BEST	AVG
ZTL20-5	313.19	330.42	313.19	337.03	313.19	324.884	313.19	324.884
ZTL20-10	120.95	129.42	129	136.15	120.95	128.923	120.95	128.923
ZTL30-10	362.15	390.17	362.15	393.67	362.15	393.668	362.15	393.668
ZTL30-15	271.26	284.05	271.26	295.02	271.26	295.021	271.26	295.021
ZTL30-20	221.91	239.46	221.91	242.49	221.91	242.489	221.91	242.489
ZTL50-10	579.59	604.11	579.59	608.86	579.59	601.508	579.59	590.549
ZTL50-20	562.07	585.79	562.07	591.89	506	583.659	506	557.943
ZTL50-25	455.27	491.97	455.27	510.89	417.82	502.014	417.82	486.847
ZTL50-30	582.17	594.00	584.39	602.81	582.17	587.57	576.5	586.942
ZTL50-40	404.39	420.41	404.39	451.12	404.39	418.024	396.66	409.815
AVG	387.30	406.98	388.32	416.99	377.94	407.78	376.60	401.71
AVG GAP	2.76%	7.46%	3.02%	9.69%	0.35%	7.64%	0.00%	6.25%

Table 3. MH versions comparison

INSTANCE	UB	IS	MH-BEST	MH-AVG
ZTL20-5	345.85	452.55	313.19	324.884
ZTL20-10	130.24	196.51	120.95	128.923
ZTL30-10	427.96	470.55	362.15	393.668
ZTL30-15	372.13	414.61	271.26	295.021
ZTL30-20	275.38	522.23	221.91	242.489
ZTL50-10	616.21	715.11	579.59	590.549
ZTL50-20	687.15	796.16	506	557.943
ZTL50-25	570.59	654.11	417.82	486.847
ZTL50-30	582.17	616.56	576.5	586.942
ZTL50-40	474.04	540.33	396.66	409.815
AVG	448.17	537.87	376.60	401.71
GAP with MODEL			-15.97%	-10.37%
GAP with IS			-29.98%	-25.32%

Table 4. MH and Model comparison

found by the model within the timelimit are reported in column 4, while the best lower bounds are reported in column 5. Column 6 shows gaps between upper and lower bounds. The last row reports averaged values. As shown in Tab. 2, the gaps between upper and lower bounds are very large. This justify the use of heuristic methods.

In order to analyze the performance of each component of the MH, different MH versions have been tested. In the first one only the Random Removal (RR) destroy operator has been used while in the second only the Node Removal (NR) one. In th third we have considered both RR and NR, each one with a selection probability equal to 0.5, while, in the last version, all the three defined operators are applied: RR and NR both with a probability equal to 0.4 while the Worst Removal one, (WR), with a probability of 0.2. The WR operator has not been tested alone because, given its deterministic nature, it will remain stuck in local minima. All the versions have been tested starting from an initial solution obtained running the Model with a timelimit of 20 seconds. Computational results are summarized in Tab. 3. Both best and average values obtained over 10 runs are reported. The last two rows reports averaged values and the gap with the best performing version. As expected, the RR+NR+WR obtains show the best performance both in terms of best solution and in terms of averaged solution. Combining only RR and NR we obtain results very near to the best ones, while larger (but still small) gaps are obtained using only one operator. We can conclude that a great advantage is reached combining RR and NR, while the addition of WR yield to a further marginal improvement. Averaged computational times are around 50 seconds for the RR and around 20 seconds for the other three versions. In fact, RR is more accurate than NR but has a lower convergence speed, i.e. at each iteration it provides small improvements, while NR provide a larger improvement even from the first iterations. As discussed before, WR is not able to enhance diversification, but it is a powerful intensification tool and, if applied in combination with the other operators, it is capable to reach local minima which cannot be reached using the other operators. Preliminary parameters tuning tests, not explicitly reported here, have shown that the best probabilities configuration is the one used for this experiments.

In Tab. 4 we compare results obtained with the best configuration of MH with those obtained by the Model within a timelimit of 10000 seconds and with the initial solution (IS) obtained by the Model within a timelimit of 20 seconds. As shown in the table, MH strongly outperform the model, (10.37% on average), within a computational time which is more than two orders of magnitude smaller. Furthermore the very large improvement respect to the initial solution (25.32% on average) shows the efficacy of the proposed approach.

7 Conclusions and Future Developments

In this paper the Vehicle Routing Problem with Mixed Fleet and Limited Traffic Zones (VRPLTZ) have been introduced. which can be seen as an extension of the well known Vehicle Routing Problem (VRP), in which the fleet is composed

both by electric (EV) and by traditional vehicles, (TV). The customers area is split in a Limited Traffic Zone (LTZ) and a Free Access Zone (FAZ). In the LTZ, an access restriction time window is imposed for TV while EV may enter it at anytime. EV must respect a limit on the maximum route length, while no length restriction are imposed to routes covered by TV. A maximum route duration is imposed for all the routes. This problem has a great practical relevance since it describes freight distribution within city centers, where is common to face the presence of LTZ for TV, in order to promote EV usage and sustainability. Despite literature on VRP with Heterogeneous fleet is wide, the features of the VRPLTZ have never been addressed before. In fact, in the problems treated in the literature, vehicles categories differs among each other by cost and capacity, but time windows and route length constraints hold for all the vehicles. The novel aspect of the VRPLTZ is that those constraints hold only for a subset of the vehicles. A Mixed Integer Programming formulation and a Large Neighborhood Search Based Matheuristic, (LNS), has been proposed. Computational results, carried out on realistic instances, showed the efficiency and effectiveness of the proposed approach. Future development in this field could address a further extension of the VRPLTZ in which additional features are considered, such as vehicles loading capacity, customers time windows, fixed vehicle purchasing costs, exc.. From a methodological point of view, it could be interesting to develop highly performing exact methods, such as column generation or branch and price, which generally are very effective but take very long computational times. In this way we will have a more challenging term of comparison for the LNS.

References

1. R.K. Ahuja, Ö. Ergun, J.B. Orlin, and A.P. Punnen. A survey of very largescale neighborhood search techniques. *Discrete Applied Mathematics*, 123:75–102, 2002.
2. R. Baldacci, M. Battarra, and D Vigo. *The Vehicle Routing Problem: Latest Advances and New Challenges*, chapter Routing a Heterogeneous Fleet of Vehicles, pages 3–27. Springer, Berlin, 2008.
3. T. Bektaş and G. Laporte. The pollution-routing problem. *Transportation Research Part B*, 45:1232–1250, 2011.
4. F. Bruglieri, M.and Pezzella, O. Pisacane, and S. Suraci. A variable neighborhood search branching for the electric vehicle routing problem with time windows. *Electronic Notes in Discrete Mathematics*, (7):221–228, 2015.
5. R.G. Conrad and M.A Figliozzi. The recharging vehicle routing problem. In *Proceedings of the 2011 Industrial Engineering Research Conference*, 2011.
6. R. Dekker, J. Bloemhof, and I. Mallidis. Operations research for green logistics an overview of aspects, issues, contributions and challenges. *European Journal of Operational Research*, 219:671–679, 2012.
7. E. Demir, T. Bektaş, and G. Laporte. An adaptive large neighborhood search heuristic for the pollution-routing problem. *European Journal of Operational Research*, 223:346–359, 2012.
8. E. Demir, T. Bektaş, and G. Laporte. The bi-objective pollution-routing problem. *European Journal of Op*, 232:464–478, 2014.
9. S. Erdogan and E. Miller-Hooks. A green vehicle routing problem. *Transportation Research Part E*, 48:100–114, 2012.

10. A. Felipe, M.T. Ortuo, G. Righini, and G. Tirado. A heuristic approach for the green vehicle routing problem with multiple technologies and partial recharges. *Transportation Research Part E: Logistic and Transportation Review*, 71:111–128, 2014.
11. A. Franceschetti, D. Honhon, T. Van Woensel, T. Bektaş, and G. Laporte. The time-dependent pollution-routing problem. *Transportation Research Part B*, 56:265–293, 2013.
12. B.L. Golden, A.A. Assad, L. Levy, and F.G. Gheysens. The fleet size and mix vehicle routing problem. *Computers & Operations Research*, 11:49–66, 1984.
13. F. Gonçalves, S.R. Cardoso, S. Relvas, and A.P.F.D. Barbosa-Povoa. Optimization of a distribution network using electric vehicles: A vrp problem. Technical report, CEG-IST, UTL, Lisboa, 2011.
14. G Hiermann, J. Puchinger, S. Ropke, and R.F. Hartl. The electric fleet size and mix vehicle routing problem with time windows and recharging stations. *European Journal of Operational Research*, 2016. available online.
15. S. Mancini. *Logistics: Perspectives, Approaches and Challenges*, chapter Multi-echelon freight distribution systems: a smart and innovative tool for increasing logistic operations efficiency, pages 171–182. Nova Publisher, New York, 2013.
16. S. Mancini. The hybrid vehicle routing problem. Technical report, Optimization-Online, 2015.
17. S. Mancini. A real-life multi depot multi period vehicle routing problem with a heterogeneous fleet: Formulation and adaptive large neighborhood search based matheuristic. *Transportation Research Part C*, doi:10.1016/j.trc.2015.06.016, 2015.
18. D. Pisinger and S. Ropke. *Handbook of Metaheuristics: International Series in Operations Research & Management Science*, chapter Large Neighborhood Search, pages 399–419. Springer, Berlin, 2010.
19. M. Schneider, A. Stenger, and D. Goeke. The electric vehicle-routing problem with time windows and recharging stations. *Transportation Science*, 48(4):500–520, 2014.
20. G. Schrimpf, J. Schneider, H. Stamm-Wilbrandt, and G. Dueck. Record breaking optimization results using the ruin and recreate principle. *Journal of Computational Physics*, 159:139–171, 2000.
21. E. Taniguchi, R.G. Thompson, T. Yamada, and R. van Duin. *City Logistics. Network Modelling and Intelligent Transport Systems.* Elsevier, 2001.

Automatic Configuration of MIP Solver: Case Study in Vertical Flight Planning

Zhi Yuan*

IRIDIA, CoDE, Université Libre de Bruxelles, Brussels, Belgium

Abstract. Mixed Integer Programming (MIP) solvers are highly parameterized and randomized matheuristic algorithms. How to properly design experiments for algorithm comparison using MIP solvers is an important topic. In this work, three classic MIP formulations for piecewise linear interpolation underlying the real-world vertical flight planning problem are compared. We analyze the performance variability of MIP solvers due to random seed settings and parameter configurations. An automatic configuration tool is used to fine-tune the algorithmic parameters of the MIP solver, which significantly improves its performance, and leads to more reliable algorithm comparison.

Keywords: Automatic configuration; mixed integer programming; performance variability; flight planning; piecewise linear interpolation

1 Introduction

Mixed Integer Programming (MIP) is a general way for modelling many real-world optimization problems. A general-purpose solver for MIP, such as Cplex, Gurobi, or SCIP, can be regarded as a matheuristic algorithm that integrates the exact branch-and-cut algorithm with various MIP-based heuristics such as local branching, feasibility pump, among others (cf. [7] for a survey). Such MIP solvers usually contain a large number of parameters, e.g., IBM ILOG Cplex 12.6 has a total of 159 parameters [15]. Most of these parameters are related to hardware or tolerance, e.g., available working memory, number of threads, random seed, time limit, optimum tolerance, constraint violation tolerance, etc. These should be fixed according to the hardware in use and the practicality of the problem. Besides, there are algorithmic parameters of different types, e.g., categorical parameters, such as options of branching strategies, LP method, whether to use certain heuristic; numerical parameters, such as how often a certain heuristic or perturbation is applied, cut limits, and so on; and conditional parameters, such as perturbation constant is only used when perturbation is switched on, or limit of strong candidate list or strong candidate iteration is only used when strong branching is selected. These algorithmic parameters could potentially be automatically configured for each particular class of problems. There exists work on automatic configuration of the MIP solvers [13] using a configuration software

* Part of the work was done when ZY was at Helmut Schmidt University Hamburg

ParamILS [14]. Other general-purpose configurators for such task also include iterated racing [3], GGA [1], SMAC [11], etc.

In this work, we empirically compare three classic piecewise linear formulations, as it arises from the real-world vertical flight planning problem [28, 26], by automatic configuration of MIP solver. We also investigate the performance variability of the MIP solver due to both random seed settings and parameter configurations.

2 Vertical Flight Planning

The vertical flight planning (VFP) problem concerns assigning optimal cruise altitude and speed to each trajectory-composing segment, such that the fuel consumption is minimized, and the arrival time constraints are satisfied. The original MIP models for VFP [28, 27] that assign continuous speed consist of second-order cone constraints, which lead to prohibitively long computation time. The speed discretization scheme proposed in [26] transforms the nonlinear model into linear, and significantly reduces the computation time to within seconds or minutes.

2.1 Mixed Integer Linear Programming model

The MIP model of VFP using discrete speed [26] is as follows.

$$\min \quad w_0 - w_n \tag{1}$$

$$\text{s.t.} \quad t_0 = 0, \quad \underline{T} \leq t_n \leq \overline{T} \tag{2}$$

$$\forall i \in S: \quad \Delta t_i = t_i - t_{i-1} \tag{3}$$

$$\forall i \in S: \quad \sum_{v \in V} \mu_{i,v} = 1 \tag{4}$$

$$\forall i \in S: \quad \Delta t = \sum_{v \in V} \mu_{i,v} \cdot \Delta T_{i,v} \tag{5}$$

$$w_n = W^{dry} \tag{6}$$

$$\forall i \in S: \quad w_{i-1} = w_i + f_i \tag{7}$$

$$\forall i \in S: \quad f_i = \sum_{v \in V} \mu_{i,v} \cdot \widehat{F}_{i,v}(w_i). \tag{8}$$

The total fuel consumption (1) measured by the difference of aircraft weight before and after the flight is minimized; (2) ensures the flight duration within a given time window; (3) preserves the time consistency; Only one speed is assigned to each segment by (4), and the travel time on each segment depends on the speed assignment by (5). (6) initializes the weight vector by assuming all trip fuel is burnt during the flight; weight consistency is ensured in (7) and the fuel consumption of each segment in (8) is calculated based on the speed selection μ and a piecewise linear function $\widehat{F} : [w_0, w_{max}] \rightarrow \mathbb{R}$ interpolating the fuel consumption F based on weight. The piecewise linear function can be modelled by three different formulations.

The Convex Combination (Lambda) Method. A variant of the *convex combination* (*Lambda*) method [5] can be formulated as follows. To interpolate F we introduce binary decision variables $\tau_k \in \{0,1\}$ for each $k \in K$, and continuous decision variables $\lambda_k^l, \lambda_k^r \in [0,1]$ for each $k \in K$.

$$\sum_{k \in K} \tau_k = 1 \tag{9a}$$

$$\forall\, k \in K: \quad \lambda_k^l + \lambda_k^r = \tau_k \tag{9b}$$

$$w = \sum_{k \in K} (w_{k-1} \cdot \lambda_k^l + w_k \cdot \lambda_k^r) \tag{9c}$$

$$\widehat{F}(w) = \sum_{k \in K} (F(w_{k-1}) \cdot \lambda_k^l + F(w_k) \cdot \lambda_k^r) \tag{9d}$$

The Incremental (Delta) Method. The *incremental (Delta) method* was introduced by Markowitz and Manne [19]. It uses binary decision variable $\tau_k \in \{0,1\}$ for $k \in K$ and continuous decision variables $\delta_k \in [0,1]$ for $k \in K$, and

$$\forall\, k \in K: \quad \tau_k \geq \delta_k \tag{10a}$$

$$\forall\, k \in K \setminus \{n\}: \quad \delta_k \geq \tau_{k+1} \tag{10b}$$

$$w = w_0 + \sum_{k \in K} (w_k - w_{k-1}) \cdot \delta_k \tag{10c}$$

$$\widehat{F}(w) = F(w_0) + \sum_{k \in K} (F(w_k) - F(w_{k-1})) \cdot \delta_k \tag{10d}$$

The Special Ordered Set of Type 2 (SOS) Method. Instead of introducing binary variables for the selection of a particular interval, we mark the lambda variables as belonging to a *special ordered set of type 2* (*SOS*). That is, at most two adjacent variables from an ordered set $(\lambda_0, \lambda_1, \ldots, \lambda_m)$ are positive. Such special ordered sets are treated by the solver with a special SOS branching [2]. We introduce continuous decision variables $0 \leq \lambda_k \leq 1$ for each $k \in K_0$, and:

$$\mathrm{SOS}(\lambda_0, \lambda_1, \ldots, \lambda_m) \tag{11a}$$

$$w = \sum_{k \in K} (w_{k-1} \cdot \lambda_{k-1} + w_k \cdot \lambda_k) \tag{11b}$$

$$\widehat{F}(w) = \sum_{k \in K} (F(w_{k-1}) \cdot \lambda_{k-1} + F(w_k) \cdot \lambda_k) \tag{11c}$$

The comparison of these classic piecewise linear formulations has been the topic in many scientific publications on various optimization problems, including gas network design [20], water network design [10], transportation [24], process engineering [9], flight planning [27], etc. SOS method was found the best in [20], while Delta method was best in [10], and mixed results among the three methods were presented in [24, 9, 27], despite the superior theoretical property of the Delta method over the Lambda method [23].

2.2 Problem Instance

Two of the most common aircraft types, Airbus 320 (A320) and Boeing 737 (B737), are used for our empirical study. The aircraft performance data are provided by Lufthansa Systems AG. We generated random instances including three flight ranges for A320: 1000, 2000, and 3000 nautical miles (NM), and one flight range of 1500 NM for B737. Two types of segment lengths are generated, the homogeneous instances include segment lengths uniformly randomly generated from 40 to 60 NM, while the heterogeneous instances include segment lengths generated from a uniform distribution from 10 to 90 NM. The expected numbers of segments are 20, 30, 40, and 60 for flight ranges of 1000, 1500, 2000, and 3000 NM, respectively. Three time constraints are considered which require the aircraft to accelerate over its unconstrained optimal speed by factors of 2%, 4%, and 6%. For each of the four aircraft ranges with two homogeneities and three acceleration factors, 10 random instances were generated, totalling 240 instances for testing purpose. Besides, another 240 instances were generated in the same way with different random seeds for training purpose. It is worth noting that these instances represent a broad range of the vertical flight planning problem.

3 Performance Variability of MIP solver

The MIP solvers were believed by many researchers and practitioners for a long time to be deterministic and perfect for benchmarking. The issue of *performance variability* in MIP solvers was first introduced to the MIP community by [4], where she reported a seemingly neutral change in the computing environment can result in a drastic change in solver performance. An experiment in [6] also reported a drastic performance variability of up to 25% by adding redundant constraints. A performance variability of up to a factor of 915 for the MIPLIB instances is also observed in [17] by randomly permuting rows or columns of a MIP model. The roots of such performance variability were first explained in [17] to be *imperfect tie-breaking*. There are many decisions to make in a branch-and-cut process, e.g., the cut separation, cut filtering, and the order of the variables to branch on, etc. Such decisions are made based on ordering the candidates by a score. However, it is impossible to have a perfect score that uniquely distinguishes all candidates at each step. When a tie in the score occurs, a deterministic choice was always made, e.g., by taking always the first candidate. Therefore, changing the order of the variables or constraints will lead to a change of path in the tree search, thus very different behavior and performance in MIP solver. It was further argued in [18] that even if a perfect score for tie-breaking exists, it may be too expensive to compute. Therefore, randomness is intrinsic in MIP solvers, and one can even exploit the randomness to improve performance as in [8], where a *bet-and-go* approach was proposed that tried out a number of different neutrally perturbed settings for a short runtime, and then pick the best setting and continue for a long run. However, performance variability must be taken into account in scalability study and benchmarking different formulations or

Table 1. The performance variability of Cplex with default setting under two random seeds for each testing instance: a fixed default seed and a randomly generated seed.

Method	1 thread					12 threads				
	#solved	var.coef.		$\frac{max}{min}$		#solved	var.coef.		$\frac{max}{min}$	
	def./ran.seed	avg.	max.	avg.	max.	def/ran.seed	avg.	max.	avg.	max.
Lambda	233/236	0.28	1.84	1.91	23.79	240/240	0.15	0.82	1.18	2.39
SOS	48/49	0.33	1.06	1.50	3.23	64/64	0.37	1.72	1.82	13.49
Delta	227/230	0.34	1.48	1.59	6.70	220/220	0.22	1.37	1.31	5.31

new algorithmic ideas to avoid misinterpretation. A more detailed discussion on performance variability can be found in [18].

In response to the issue of performance variability, MIP solvers start to break ties randomly, and allow users to specify a random seed to initialize the random number generator, e.g., Cplex since 12.5. Ready or not, it is time for us to accept the fact that MIP solvers are randomized algorithms. An experimental study of the MIP solvers should follow a proper experimental setup for randomized algorithms, cf. [22, 16]. An empirical study of a (randomized) MIP solver based on a fixed random seed (as done in e.g. [20, 24, 10, 9, 27]) will limit the conclusion to only a specific implementation of the MIP solver with a peculiar sequence of random numbers. A proper experimental setup for studying a specific problem instance should be performed by running MIP solver with multiple random seeds and collecting proper statistics; empirical study for a problem class should include a wide range of instances from the problem class, and assign to each instance a different random seed. Each instance can be paired with a unique random seed, such that algorithmic candidates are evaluated on each instance with the common random seed, so as to reduce evaluation variance [21].

All experiments ran on a computing node with 12-core Intel Xeon X5675 CPU at 3.07 GHz and 48 GB RAM. We use Cplex 12.6 with both single thread and 12 threads. We fixed the memory parameters such as working memory (`WorkMem`) to 40 GB, and the node storage file switch (`NodeFileInd`) to 3. We first ran Cplex with the default setting on the 240 testing instances. Two random seed settings were run, one with Cplex default fixed seed, and the other assigned a different random number to each instance as random seed. The goal is to evaluate the performance variability of Cplex due to different random seed settings. Two types of variability measure are used: (i) *variation coefficient* (termed variability score in [17]), which is a relative variability measure defined as the standard deviation divided by the mean, i.e., $\frac{2\cdot|t_1-t_2|}{t_1+t_2}$ where t_1 and t_2 are the computation time of the two random seed settings; (ii) the *ratio* of the maximum and minimum computation time between the two random seeds for each instance, which is also mentioned in [17]. Both the average and the maximum of both variability measures across all instances are presented in Table 1. The variability measure is calculated with only the instances that are solved within 300 seconds. The variability of Cplex due to random seed on vertical flight planning problem is certainly not negligible. The average variation coefficient is between 0.15 to 0.37,

while average ratio is between 1.18 to 1.91. In general, the variability is higher in single thread than parallel computation, especially in the Lambda method: both its average measures are 60% to 85% higher in single thread, and it has the highest max-min-ratio of 23.79, which is on an instance that is solved by a randomly generated seed in 12.6 seconds, but takes the Cplex with default seed 300 seconds and still leaves a 0.02% gap. The variability appears to increase for models that are harder to solve, i.e., the variability of the SOS is higher than the Delta and then higher than the Lambda method. Note that the variability measure for especially the SOS method is probably an underestimate, since it is only calculated on a small set of solved instances. Hence although the parallel SOS seems more variable than the sequential, the high variability scores are mainly contributed by the 15 additional instances that are solved in the parallel SOS but not in the sequential SOS. Although using the randomly generated seed solves a few more instances to optimality than using the default seed in 1-thread case, no statistical significant difference is found by Wilcoxon's signed rank test or binomial test.

4 Automatic configuration of MIP solver

4.1 Automatic solver configuration

The MIP solvers are highly parameterized matheuristic algorithms. Cplex 12.6 has a total of 159 parameters, where around 70 to 80 parameters can influence algorithmic behaviors. Although Cplex claimed that "A great deal of algorithmic development effort has been devoted to establishing default ILOG CPLEX parameter settings that achieve good performance on a wide variety of MIP models." [15, p. 222], this configuration needs not be a good choice for a specific problem class of interest. Experimental study using MIP solver with only the default configuration will limit the study to only a specific implementation of the MIP solver with a particular parameter setting rather than a general algorithm. Especially when empirically comparing algorithmic ideas, there will be high risk of misinterpretation due to the interaction of the algorithmic idea with certain algorithmic parameters of Cplex. Each algorithmic idea to be compared should be given a fair amount of configuration effort by the same procedure.

The automatic configuration experiments are performed on the 240 training instances. 11 copies of the training set of the instances are generated, each using a different random instance order and a different random seed for each instance. 10 copies are used for 10 independent *training* trials, respectively, and one copy is used for *validation* where the best configurations found in the 10 training trials are compared. The best configuration identified by the validation phase is applied to the *testing* set of instances to assess the quality of the automatically trained configuration. The automatic configuration of Cplex on MIPLIB instances done by Hutter et al. [13] has sped up over the use of default setting by a factor of 1.3 to 52. We followed the algorithmic parameter file for solving MILP by Cplex listed at [12], which has a total of 74 parameters. Two parameters are removed from the list: `lpmethod` since no pure LP exists in our problem (LP relaxation in MIP

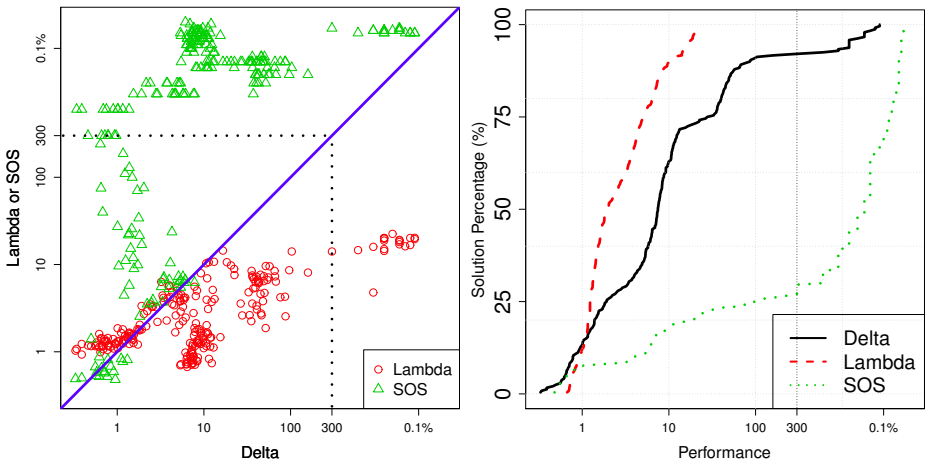


Fig. 1. The comparison of three piecewise linear formulations: Delta, Lambda, and SOS methods, solved by Cplex with default setting.

is controlled by `MIP startalgorithm` or `MIP subalgorithm`); `NodeFileInd` is fixed to 3, such that it allows Cplex to write the node files to hard disk rather than using a swap when the working memory (`WorkMem`) is exceeded.

There exists software for such automatic algorithm configuration tasks, e.g. ParamILS [14] is used in [13]. We have used JRace, which is a Java software for racing based configurators such as iterated racing [3] and post-selection [25, 29]. Each of the three piecewise linear formulations, Lambda, Delta, and SOS method, is trained separately. The cutoff time for the validation and testing phase is set to 300 seconds, and the cutoff time for the training phase is set to a much smaller value of 10 seconds. For the instances that are unsolved with a gap of $\delta\%$ after the cutoff time $\kappa = 10$ seconds, we modified the penalized average runtime (`PAR-10`) of $10 \cdot \kappa$ in [13] to `mPAR-10`: $10 \cdot \kappa \cdot (1 + (\delta - 0.01)/10)$, such that the unsolved runs during training can still compare with each other using their optimality gaps. A configuration budget of 3 000 evaluations is allowed for each training trial, which amounts to maximum around 8 hours per trial.

4.2 Formulation Comparison with Default Setting

Figure 1 compares the three piecewise linear formulations, Delta, Lambda, and SOS methods, using the Cplex default setting with 12 threads and randomly generated seed on the testing set. The performance distribution plot on the left shows the runtime or the optimality gap if unsolved after 300 seconds. The Delta method is clearly better performing than the SOS, while it is also clearly outperformed by the Lambda method except for a few smallest instances. The runtime development plot on the right shows the percentage of instances that are solved within a runtime or reach a gap after cutoff. It agrees with the clear trend that the Lambda method is the overall best performing formulation of the three.

Table 2. The comparison of the default configuration and the automatically tuned configuration of Cplex and the performance variability induced by using the two configurations. Number of solved instances, average runtime for solved instances, and the number of wins (out of 240) over the other configuration are shown in the comparison of the two configurations. The average speedup factor is also presented.

Method	default			tuned			avg.	var.coef.		$\frac{max}{min}$	
	#solved	avg.time	#win	#solved	avg.time	#win	spdup.	avg.	max.	avg.	max.
Lambda	240	4.24	84	240	3.49	151	1.21	0.20	0.83	1.25	2.42
SOS	64	38.72	39	67	18.87	170	2.05	0.61	1.81	2.83	20.62
Delta	220	38.43	14	240	2.49	226	15.37	1.19	1.92	9.24	51.99

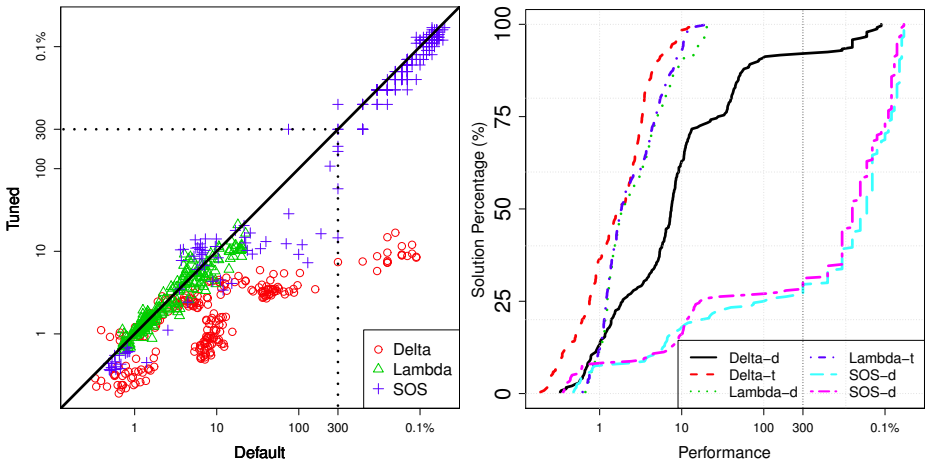


Fig. 2. The comparison of default and tuned parameter setting of Cplex on the three piecewise linear formulations: Delta, Lambda, and SOS methods.

In fact, the Lambda method is the only one that solves all the instances within 300 seconds. The Delta method solves 220 out of 240, and the SOS method solves only 64. The difference between the three is also statistically significant by the Wilcoxon's signed rank test or binomial test with $\alpha = 0.01$.

4.3 The Automatically Tuned Setting versus the Default Setting

The comparison of the default configuration and the automatically tuned configuration is listed in Table 2. The tuned configuration statistically significantly outperforms the Cplex default setting for each of the three formulations. The tuned setting of the Lambda method speeds up in average 21% over the default setting, and performs better in 151 instances out of 240 while worse in 84. The tuned setting of the SOS method solves 3 more instances to optimality, and is more than twice as fast as the default setting in the solved instances, It also wins in 170 instances while loses in 39. The tuned setting of the Delta method solves all the 20 remaining unsolved problems, improves the default Cplex setting by

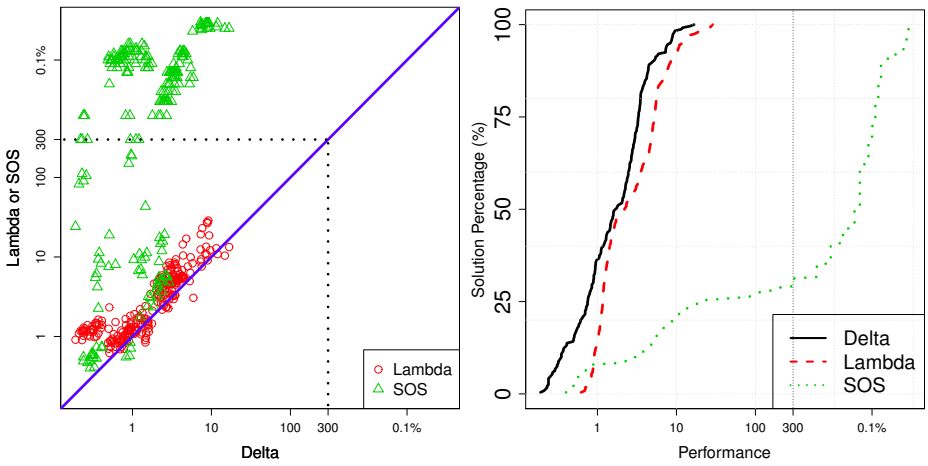


Fig. 3. The comparison of three piecewise linear formulations: Delta, Lambda, and SOS methods, solved by Cplex with tuned configuration.

an average speedup factor of over 15, and performs better in 226 out of 240 instances. Different formulations benefit from automatic configuration differently. This is best illustrated in Figure 2: the tuned configuration improves the Delta method much more than the SOS method which benefits more than the Lambda method. The performance variability measures due to the two different Cplex configurations shown in the last four columns of Table 2 is also drastically higher than the ones shown in Table 1. The highest ratio is ca. 52 times, with an instance solved in 5.8 seconds by the tuned configuration while taking Cplex with default setting 300 seconds with a gap of 0.03%.

4.4 Formulation Comparison with Tuned Setting

With the automatically tuned configuration, the ranking of the three piecewise linear formulations shown in Figure 3 looks quite different from Figure 1 with default setting. The Delta method clearly and statistically significantly outperforms Lambda method, and the average speedup is 40%. This shows that the comparison of different MIP formulations heavily depends on the setting of the MIP solver. Comparing them using only the default setting may limit the conclusion to only a particular implementation of the MIP solver. Generalizing such conclusion may lead to misinterpretation. As a more reliable experimental setup to benchmark different formulations, an automatic configuration of the MIP solver should always be performed.

4.5 Further Analysis on the Tuned Configuration

Since the tuned configuration drastically improves the Delta method over the default one for Cplex, a further analysis of the tuned Delta setting is conducted.

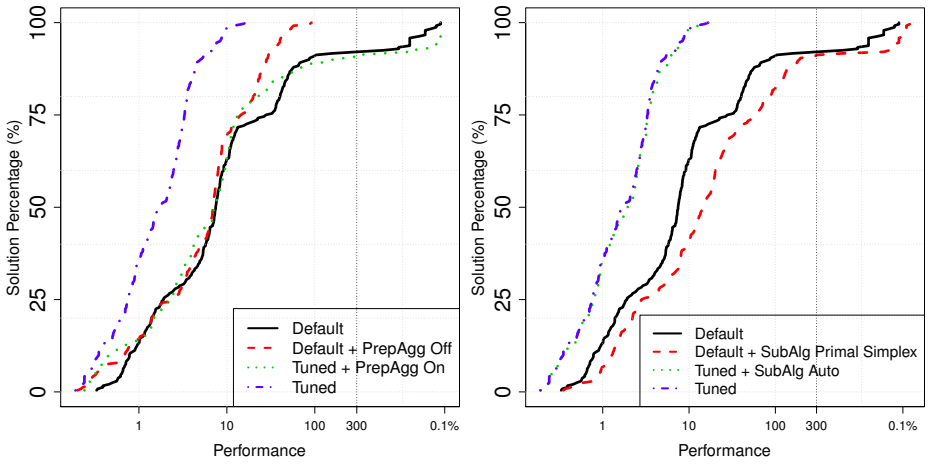


Fig. 4. Solver performance when changing one parameter `preprocessing aggregator` (left) or `MIP subalgorithm` (right) from either the default or the tuned configuration.

38 out of the 72 parameters have been varied by the automatic configuration. We tried to vary from the default (tuned) configuration one of these 38 parameters to its tuned (default) value, respectively, evaluated them on the testing instances to assess their influence. The list of changed parameters and their default and tuned value can be found in `http://iridia.ulb.ac.be/~zyuan/downloads/deltaParam.xls`. There are 15 parameter changes from the default setting that can lead to a significant performance improvement (by Wilcoxon's signed rank test with $\alpha = 0.01$). The most influential one is to turn off the `preprocessing aggregator` (`on` by default). Varying only this parameter already solves the 20 by default unsolved instances, and has an average runtime of 11.4 seconds, which speeds up the default setting by a factor of 3.4. However, 3 single parameter variations from the default setting leads to significant performance deterioration. The most worsening one is changing the `MIP subalgorithm` from default value of `auto` (i.e., always select `dual simplex`) to `primal simplex`, which takes 2.2 times longer runtime than the default setting. On the other hand, there are 7 parameters, changing which from the tuned configuration towards default leads to statistically significantly worse performance. Again, the most influential is to turn the `preprocessing aggregator on` from the tuned configuration, which performs even worse than the default configuration for the hardest instances, as shown in the left of Figure 4. An interesting observation is that the parameter `MIP subalgorithm` also belongs to one of the 7 most influential parameters, varying which from `primal simplex` to default `auto` statistically significantly worsens the performance (p-value 10^{-8}), and takes in average 5% more runtime, as shown in the right of Figure 4. This shows that it can be misleading to analyze the influence of a single parameter to a given problem, or to set parameters in a one-factor-at-a-time fashion, since it also depends on other parameters.

Such parameter correlation should be taken into account, and a sophisticated automatic configuration tool such as JRace should be a good choice.

5 Conclusions

MIP solvers are highly parameterized and randomized matheuristic algorithms. In this article, we analyze the performance variability of a MIP solver Cplex, and apply an automatic configuration to Cplex for comparing three classic piecewise linear formulations in the vertical flight planning problem. The performance variability in Cplex due to random seed setting is certainly not negligible, the average variation coefficient ranges from 0.15 to 0.37. The performance variability due to different Cplex parameter settings is even higher, and the formulation comparison depends heavily on the Cplex setting. The experiments are conducted using different random seed and automatic configuration tool JRace. The automatically tuned configuration significantly improves the Cplex default setting in all the three formulations by a factor of up to 15. Besides, the automatic configuration makes a theoretically superior but by default computationally inferior formulation (the Delta method) stand out as the best performing formulation.

Acknowledgments This work was partially funded by BMBF Verbundprojekt E-Motion. The author thanks Swen Schlobach and Anton Kaier from Lufthansa Systems for providing aircraft performance data, Ingmar Vierhaus, Armin Fügenschuh, and Alexander Martin for literature suggestions. Special thanks to Éric Taillard, Vittorio Maniezzo, and Stefan Voß for the inspiring talks on MIP solver variability in Hamburg, Dec. 2015.

References

1. Ansótegui, C., Sellmann, M., Tierney, K.: A gender-based genetic algorithm for the automatic configuration of solvers. In: CP, LNCS, vol. 5732, pp. 142–157. (2009)
2. Beale, E.L.M., Tomlin, J.A.: Global Optimization Using Special Ordered Sets. Mathematical Programming 10, 52–69 (1976)
3. Birattari, M., Yuan, Z., Balaprakash, P., Stützle, T.: F-Race and iterated F-Race: An overview. In: Bartz-Beielstein, T., et al. (eds.) Experimental Methods for the Analysis of Optimization Algorithms, pp. 311–336. Springer (2010)
4. Danna, E.: Performance variability in mixed integer programming. In: Presentation at Workshop on Mixed Integer Programming (2008)
5. Dantzig, G.B.: On the significance of solving linear programming problems with some integer variables. Econometrica 28(1), 30 – 44 (1960)
6. Fischetti, M., Monaci, M.: On the role of randomness in exact tree search methods. In: Presentation at Matheuristics (2012)
7. Fischetti, M., Lodi, A.: Heuristics in mixed integer programming. Wiley Encyclopedia of Operations Research and Management Science (2011)
8. Fischetti, M., Monaci, M.: Exploiting erraticism in search. Operations Research 62(1), 114–122 (2014)
9. Fügenschuh, A., Hayn, C., Michaels, D.: Mixed-Integer Linear Methods for Layout-Optimization of Screening Systems in Recovered Paper Production. Optimization and Engineering 15(2), 533–573 (2014)

10. Geißler, B., Martin, A., Morsi, A., Schewe, L.: Using piecewise linear functions for solving MINLPs. In: Mixed Integer Nonlinear Programming, pp. 287–314. (2012)
11. Hutter, F., Hoos, H.H., Leyton-Brown, K.: Sequential model-based optimization for general algorithm configuration. In: Proc. of LION-5. pp. 507–523 (2011)
12. Hutter, F.: Automated Configuration of MIP solvers. `http://www.cs.ubc.ca/labs/beta/Projects/MIP-Config`, last accessed: 2016-05-04
13. Hutter, F., Hoos, H.H., Leyton-Brown, K.: Automated configuration of mixed integer programming solvers. In: CPAIOR. LNCS, vol. 9335, pp. 186–202. (2010)
14. Hutter, F., Hoos, H.H., Leyton-Brown, K., Stützle, T.: ParamILS: An automatic algorithm configuration framework. Journal of Artificial Intelligence Research 36, 267–306 (2009)
15. IBM ILOG CPLEX: 12.6 user's manual. 2014
16. Johnson, D.S.: A theoreticians guide to the experimental analysis of algorithms. Data structures, near neighbor searches, and methodology: fifth and sixth DIMACS implementation challenges pp. 215–250 (2002)
17. Koch, T., Achterberg, T., Andersen, E., Bastert, O., Berthold, T., Bixby, R.E., Danna, E., Gamrath, G., Gleixner, A.M., Heinz, S., et al.: MIPLIB 2010. Mathematical Programming Computation 3(2), 103–163 (2011)
18. Lodi, A., Tramontani, A.: Performance variability in mixed-integer programming. TutORials in Operations Research pp. 1–12 (2013)
19. Markowitz, H.M., Manne, A.S.: On the solution of discrete programming problems. Econometrica 25(1), 84 – 110 (1957)
20. Martin, A., Möller, M., Moritz, S.: Mixed integer models for the stationary case of gas network optimization. Mathematical programming 105(2-3), 563–582 (2006)
21. McGeoch, C.: Analyzing algorithms by simulation: variance reduction techniques and simulation speedups. ACM Computing Surveys 24(2), 195–212 (1992)
22. McGeoch, C.C.: Feature article-toward an experimental method for algorithm simulation. INFORMS Journal on Computing 8(1), 1–15 (1996)
23. Padberg, M.: Approximating separable nonlinear functions via mixed zero-one programs. Operations Research Letters 27(1), 1–5 (2000)
24. Vielma, J.P., Ahmed, S., Nemhauser, G.: Mixed-integer models for nonseparable piecewise-linear optimization: unifying framework and extensions. Operations research 58(2), 303–315 (2010)
25. Yuan, Z., de Oca, M.M., Birattari, M., Stützle, T.: Continuous optimization algorithms for tuning real and integer parameters of swarm intelligence algorithms. Swarm Intelligence 6(1), 49–75 (2012)
26. Yuan, Z., Amaya Moreno, L., Fügenschuh, A., Kaier, A., Schlobach, S.: Discrete speed in vertical flight planning. In: Corman, F., et al. (eds.) Proceedings of International Conference on Computational Logistics. Lecture Notes in Computer Science, vol. 9335, pp. 734–749. Springer (2015)
27. Yuan, Z., Amaya Moreno, L., Maolaaisha, A., Fügenschuh, A., Kaier, A., Schlobach, S.: Mixed integer second-order cone programming for the horizontal and vertical free-flight planning problem. Tech. rep., AMOS#21, Applied Mathematical Optimization Series, Helmut Schmidt University, Hamburg, Germany (2015)
28. Yuan, Z., Fügenschuh, A., Kaier, A., Schlobach, S.: Variable speed in vertical flight planning. In: Operations Research Proceedings. pp. 635–641. Springer (2014)
29. Yuan, Z., Stützle, T., Montes de Oca, M.A., Lau, H.C., Birattari, M.: An analysis of post-selection in automatic configuration. In: Proceedings of GECCO. pp. 1557–1564. ACM (2013)

Dual Heuristics and New Lower Bounds for the Challenge EURO/ROADEF 2010

Nicolas Dupin, El-Ghazali Talbi

Univ. Lille, UMR 9189 - CRIStAL - Centre de Recherche en Informatique Signal et Automatique de Lille, F-59000 Lille, France

Abstract. This paper addresses the scheduling problem of nuclear power plant outages for maintenance and refuelling, defined for the Challenge EURO/ROADEF 2010. Dual matheuristics compute dual bounds using Mixed Integer Programming (MIP) formulations. A first step designs a MIP formulation for the problem relaxing only 2 types of constraints, declaring binary variables only for the decisions of outage dates. To have tractable formulations, dual bounds are proven with simplified formulations with time step aggregations and reductions on deterministic scenarios, and also with some parametric constraint simplifications to remove variables. Several sets of dual bounds are computable, improving significantly the former best dual bounds of the literature.

Keywords: matheuristics, stochastic programming, mixed integer programming, dual bounds, EURO/ROADEF 2010 Challenge, maintenance scheduling.

1 Introduction

The 2010 ROADEF/EURO Challenge was specified by the French utility company (EDF), to address the large-scale scheduling problem of nuclear power plant outages for maintenance and refuelling. The optimisation problem was formulated using 2-stage stochastic programming for the challenge with uncertainty on power demands, production capacities and cost. The first stage optimisation problem concerns the weeks of outages on nuclear power plants and the refuelling quantities. The second stage optimisation problem computes production plans implied by the first stage decisions for all stochastic scenarios, to minimise the average production cost over all the scenarios.

This article computes tractable lower bounds for the 2010 ROADEF/EURO Challenge using Mixed Integer Programming (MIP) formulations. Our approach can be seen as an illustration of dual heuristics for MIP emphasised in [11] : heuristics are useful for MIPs not only to exhibit primal solutions, but also to improve the quality of dual bounds. Branch-and-bound algorithms relied solely on cutting planes, dual heuristic alternatives are to incorporate heuristic relaxations to improve dual bounds. Key points are to prove that the heuristic relaxations lead to dual bounds for the original problem.

State of the art of the 2010 ROADEF/EURO Challenge One major difficulty of the challenge is to handle the size of the problem. The best results were mainly obtained with a frontal local search like in [6]. Matheuristics were also successfully and widely implemented for the challenge, taking advantage of the modelling facilities of mathematical approaches, to design a heuristic tackling big size instances. Several approaches used MIP or LP inside a heuristic algorithm iterating following the 2-stage structure, like [1, 2, 7–9]. MIP is a useful framework for the first level scheduling problem as implemented in [9], a variant is to use Constraint Programming like in [2, 7, 8]. LP is useful to optimise separately production or refuelling problems when outage decisions are fixed, like in [1, 9, 13]. ([12, 13]) approaches derived a heuristic from an exact method with a MIP formulation on simplified problems before a reparation phase. A common simplification was to aggregate the production time steps to weeks. [12] did not aggregate the stochastic scenarios, solving a stochastic MIP by Bender's decomposition. However, this approach was not efficient to tackle the real size instances. [13] fixed outage decisions on the average scenario with a column generation algorithm dualising demand coupling constraints. Another efficient MIP matheuristic is provided in [3], using a MIP deterministic formulation to solve large scale instances on a average scenario with a VNS.

Dual heuristics and lower bounds for the challenge It was an open question after the Challenge to have dual bounds for this large scale problem. The only work which published dual bounds was [2] with two dual heuristics: a greedy approach for a simple relaxation and a flow network relaxation solving a minimum cost flow problem. Amongst the exact methods, none could derive dual bounds for the whole ROADEF problem. The big size of the instances was a bottleneck to compute efficiently dual bounds for the whole problem of the ROADEF challenge. The Bender's decomposition [12] could theoretically furnish dual bounds, but it is too limited in its resolution capacity. Other exact approaches cannot derive dual bounds because of some restrictive hypotheses. Especially, [13] discretises the production domain in the column generation subproblems, which strengthen heuristically the feasible domain and thus cannot provide dual bounds for the original problem.

Paper outline New dual bounds for the ROADEF Challenge are provided with restricted MIP computations, proving that the different restrictions imply dual bounds for the whole problem. Section 2 gives an overview of the problem constraints, while section 3 presents a MIP model for the problem, relaxing only two sets of constraints. Section 4 provides a parametric family of dual bounds by other constraints aggregations. To deal with smaller problems, sections 5 and 6 prove that dual bounds can be computed with the aggregation of production time steps and the restriction of deterministic computations. The computational results are reported in section 7 and the conclusions are drawn in section 8.

2 Problem statement

We summarise here the problem description, we refer to [14] for more details.

Set and index Two kinds of power plants are modelled. On one hand, Type-2 (shortly T2) power plants indexed with $i \in \mathcal{I}$, correspond to nuclear power plants. T2 power plants have to be shut down for refuelling and maintenance regularly. On the other hand, Type-1 (shortly T1) power plants are indexed with $j \in \mathcal{J}$, model other power plants with more flexibility in the production.

*O*utages and *p*roduction campaigns are indexed with the cycles $k \in \mathcal{K}$ for all T2 plant. By convention, a cycle begins with the outage period for maintenance and refuelling, before the *p*roduction campaign.

The time horizon is discretised with two kind of homogeneous time steps. The outage decisions are discretised weekly and indexed with $w \in \mathcal{W} = [\![1; W]\!]$, whereas $t \in \mathcal{T}$ denotes production times steps from 8h to 24h. $\mathbf{D}_t$ denotes the duration of the production time steps. t_w denotes the first production period of the week $w \in W$, each production time step t is associated to its week w_t.

Stochastic scenarios are denoted with $s \in \mathcal{S}$, discretising the uncertainty on demands, power productions and costs, each scenario s having probability π_s.

Constraint	Description
CT1	For all s, t, the total production is the demand $\mathbf{Dem}_t^s$
CT2	For all s, t, the production of T1 plant $j \in \mathcal{J}$ is in $[\mathbf{Pmin}_{jt}^s, \mathbf{Pmax}_{jt}^s]$
CT3	T2 plants productions are null during an outage
CT4-5	For all $t \in \mathcal{T}$, the production of T2 plant i is in $[0, \mathbf{Pmax}_{it}]$
CT6	Decreasing power profile at the end of nuclear cycles (relaxed here)
CT12	Modulation constraints (relaxed here)
CT7	The refuelling of outage k of T2 plant i is in $[\mathbf{Rmin}_{i,k}, \mathbf{Rmax}_{i,k}]$
CT8	Initial fuel stock for T2 unit i is $\mathbf{Xi_i}$
CT9	Dynamic evolution fuel stocks/production for T2 units.
CT10	Refuelling losses when refuelling outage k of T2 plant i
CT11	The fuel level is in $[0, \mathbf{S}_{i,k}]$ for cycle k of T2 unit i.
	The fuel level must be lower than $\mathbf{A}_{i,k+1}$ to process outage $k+1$
CT13	Time windows constraints for the beginning dates of outages
CT14-18	Minimal spacing/ maximal overlapping constraints amongst outages
CT19	maximal overlapping constraints among outages with resource constraints
CT20	Maximal number of simultaneous outages
CT21	Maximal power off-line due to outages

Table 1. Definition of the constraints of the challenge ROADEF

Objective function The objective function minimises the expected cost of production while satisfying customer load for all time steps and all production scenarios. Production costs of T1 units j are $\mathbf{C}_{jt}^s$ proportional to the production levels for all scenarios s at time step t. Production cost of T2 units are calculated

proportionally to the fuel consumption: proportional refuelling costs $\mathbf{C}^r_{i,k}$ are considered, and reduced with the proportional cost of the remaining fuel with a proportional factor $\mathbf{C}^f_{i,s}$ to avoid end-of-side effects.

Constraints description Table 1 defines the constraints with their nomenclature from CT1 to CT21 in the challenge specification [14]. CT1 are the demand constraints to equalise productions and demands for each time step and each scenario. Constraints CT2 to CT6 and CT12 model production constraints: production bounds for T1 and T2 power plants and specific technical constraints of nuclear power plants CT6 and CT12 which are relaxed in this study. Constraints CT7 to CT11 model stock level constraints: bounds on stock levels and refuelling and CT9 is the equation linking T2 production and fuel consumption. The remaining constraints (CT13 to CT21) are specific to T2 plants outage scheduling: time window constraints to begin outages, minimal spacing/ maximal overlapping constraints amongst outages, and limitations of simultaneous outages.

3 MIP formulation relaxing CT6 and CT12

In this section, we provide a MIP formulation for the problem, relaxing only constraints CT6 and CT12 similarly as [12]. It leads to a MIP formulation where the only binary variables are the outages weeks decisions. A major modelling difference with [12] is in the binary variable definitions of $d_{i,k,w}$: we define $d_{i,k,w} = 1$ if and only if the outage beginning week for unit i's cycle k is before week w.

Other continuous variables to have a linear formulation are refuelling quantities $r_{i,k}$ for each outage (i,k), T2 power productions $p_{i,k,t,s}$ at cycle k, fuel stocks at the beginning of campaign (i,k) (resp at the end) $x^{init}_{i,k,s}, x^{fin}_{i,k,s}$, T1 power productions $p_{j,t,s}$, and fuel stock $x^f_{i,s}$ at the end of the optimizing horizon.

It gives rise to the MIP formulation below. (2) is required with the definition of variables d. (3) and (4) model CT13 time windows constraints: outage (i,k) is operated between weeks $\mathbf{To}_{i,k}$ and $\mathbf{Ta}_{i,k}$. (5) models CT1 demand constraints. (6) models CT2 bounds on T1 production. (7) models CT3, CT4 and CT5 bounds on T2 production. (8) models CT7 refuelling bounds, with a null refuelling when outage i,k is not operated, ie $d_{i,k,W} = 0$. (9) writes CT8 initial fuel stock. (10) writes CT9 fuel consumption constraints on stock variables of cycles k $x^{init}_{i,k,s}, x^{fin}_{i,k,s}$. (11) models CT10 fuel losses at refuelling. (12) writes CT11 bounds on fuel stock levels only on variables $x^{init}_{i,k,s}$ which are the maximal stocks level over cycles k. thanks to (10). (13) models CT11 minimum fuel stock before refuelling, these constraints are active for a cycle k only if the cycle is finished at the end of the optimizing horizon, ie if $d_{i,k+1,W} = 1$, which enforces to have disjunctive constraints where case $d_{i,k+1,W} = 0$ implies a trivial constraints thanks to (12). (14) is a linearising constraints to enforce $x^f_{i,s}$ to be the fuel stock at the end of the time horizon. $x^f_{i,s}$ is indeed the $x^{fin}_{i,k,s}$ such that $d_{i,k,W} = 1$ and $d_{i,k+1,W} = 0$, for the disjunctive constraints (14) that write a trivial constraints

in the other cases thanks to (12), we define $\overline{S}_i = \max_k \mathbf{S}_{i,k}$. (15) is a common framework for scheduling constraints from CT14 to CT21, which was noticed independently in [9, 12].

$$v_0 = \min \sum_{i,k} \mathbf{C}^r_{i,k} r_{i,k} + \sum_{j,s,t} \pi_s \mathbf{C}^p_{j,s,t} \mathbf{D}^t \, p_{j,s,t} - \sum_{i,s} \pi_s \mathbf{C}^f_{i,s} x^f_{i,s} \tag{1}$$

$$\forall i,k,w, \quad d_{i,k,w-1} \leqslant d_{i,k,w} \tag{2}$$

$$\forall i,k, \quad d_{i,k,\mathbf{To}_{i,k}-1} \leqslant 0 \tag{3}$$

$$\forall i,k, \quad d_{i,k,\mathbf{Ta}_{i,k}} \geqslant 1 \tag{4}$$

$$\forall s,t, \quad \textstyle\sum_{i,k} p_{i,k,s,t} + \sum_j p_{j,s,t} = \mathbf{Dem}^{t,s} \tag{5}$$

$$\forall j,s,t, \quad \mathbf{Pmin}^s_{j,t} \leqslant p_{j,s,t} \leqslant \mathbf{Pmax}^s_{j,t} \tag{6}$$

$$\forall i,k,s,t, \quad p_{i,k,s,t} \leqslant \mathbf{Pmax}_{i,t}(d_{i,k,w_t-\mathbf{Da}_{i,k}} - d_{i,k+1,w_t}) \tag{7}$$

$$\forall i,k, \quad \mathbf{Rmin}_{i,k} \; d_{i,k,W} \leqslant r_{i,k} \leqslant \mathbf{Rmax}_{i,k} \; d_{i,k,W} \tag{8}$$

$$\forall i,s, \quad x^{init}_{i,0,s} = \mathbf{Xi}_i \tag{9}$$

$$\forall i,k,s, \quad x^{fin}_{i,k,s} = x^{init}_{i,k,s} - \textstyle\sum_t \mathbf{D}^t \, p_{i,k,s,t} \tag{10}$$

$$\forall i,k,s, \quad x^{init}_{i,k,s} - \mathbf{Bo}_{i,k} = r_{i,k} + \frac{\mathbf{Q}_{i,k-1}}{\mathbf{Q}_{i,k}}(x^{fin}_{i,k-1} - \mathbf{Bo}_{i,k-1}) \tag{11}$$

$$\forall i,k,s, \quad x^{init}_{i,k,s} \leqslant \mathbf{S}_{i,k} \tag{12}$$

$$\forall i,k,s, \quad x^{fin}_{i,k,s} \leqslant \mathbf{A}_{i,k+1} + (\mathbf{S}_{i,k} - \mathbf{A}_{i,k+1})(1 - d_{i,k+1,W}) \tag{13}$$

$$\forall i,k,s, \quad x^f_{i,s} \leqslant x^{fin}_{i,k,s} + \overline{S}_i(1 - d_{i,k,W} + d_{i,k+1,W}) \tag{14}$$

$$\forall c,w, \quad \textstyle\sum_{(i,k)\in\mathbf{A}^c}(\alpha_{i,k,w} d_{i,k,w}) \leqslant \beta^c_w \tag{15}$$

$$d \in \{0,1\}^N, r, p, x \geqslant 0 \tag{16}$$

v_0 is a dual bound for the whole problem, relaxing only constraints CT6 and CT12. To face the resolution limits to calculate v_0, more relaxations are considered in the following.

4 Parametric lower bounds with outage relaxations

In this section, we underestimate v^{det}_s to have easier computations of dual bounds. The idea is to relax the constraints CT3 that T2 production is null during all the outages $k > k^0$. For $k > k^0$, it allows to remove binary variables $d_{i,k,w}$, having only binaries $o_{i,k} = d_{i,k,W}$ to compute refuelling costs for T2 units. T2 productions are aggregated in one global production for cycles $k \geqslant k^0$ thanks to the outages relaxations. This aggregation of nuclear productions requires to aggregate the refuelling constraints and variables for cycles $k \geqslant k^0$ to ensure that the global nuclear production is feasible. Constraints $M^{ordo}_{k^0} d \geqslant b^{ordo}_{k^0}$ are the restriction of constraints $M^{ordo} d \geqslant b^{ordo}$ with only variables $d_{i,k,w}$ with $k \leqslant k^0$.

$$v^{rlx}_{k^0} = \min \sum_{i,k} \mathbf{C}^r_{i,k} r_{i,k} + \sum_{j,s,t} \pi_s \mathbf{C}^p_{j,s,t} \mathbf{D}^t \, p_{j,s,t} - \sum_{i,s} \pi_s \mathbf{C}^f_{i,s} x^f_{i,s} \tag{17}$$

$$\forall i,k, \quad o_{i,k+1} \leqslant o_{i,k} \tag{18}$$

$$\forall i, k \leqslant k^0, \quad o_{i,k} = d_{i,k,W} \tag{19}$$

$$M^{ordo}_{k^0} d \geqslant b^{ordo}_{k^0} \tag{20}$$

$$\forall i,k, \quad \mathbf{Rmin}_{i,k} \; o_{i,k} \leqslant r_{i,k} \leqslant \mathbf{Rmax}_{i,k} \; o_{i,k} \tag{21}$$

$$\forall j,s,t, \quad \mathbf{Pmin}^s_{j,t} \leqslant p_{j,s,t} \leqslant \mathbf{Pmax}^s_{j,t} \tag{22}$$

$$\forall i, k < k^0, s, t, \quad 0 \leqslant p_{i,k,s,t} \leqslant \mathbf{Pmax}_{j,t}(d_{i,k,w_t - \mathbf{Da}_{i,k}} - d_{i,k+1,w}) \tag{23}$$

$$\forall i,s,t, \quad p_{i,k^0,s,t} \leqslant \mathbf{Pmax}^s_{j,t} d_{i,k^0,w_t - \mathbf{Da}_{i,k^0}} \tag{24}$$

$$\forall s,t, \quad \textstyle\sum_{i,k \leqslant k^0} p_{i,k,s,t} + \sum_j p_{j,s,t} = \mathbf{Dem}^{t,s} \tag{25}$$

$$\forall i, \quad x^{init}_{i,0} = \mathbf{Xi}_i \tag{26}$$

$$\forall i, k \leqslant k^0, \quad 0 \leqslant x^{init}_{i,k} \leqslant \mathbf{S}_{i,k} \tag{27}$$

$$\forall i,s,t,k < k^0, \quad 0 \leqslant x^{fin}_{i,s,k} = x^{init}_{i,s,k} - \textstyle\sum_t \mathbf{D}^t \, p_{i,k,s,t} \tag{28}$$

$$\forall i,s,k < k^0, \quad x^{fin}_{i,s,k} \leqslant \mathbf{A}_{i,k+1} + (\mathbf{S}_{i,k} - \mathbf{A}_{i,k+1}) \, (1 - o_{i,k+1}) \tag{29}$$

$$\forall i, k \leqslant k^0, \quad x^{init}_{i,k} - \mathbf{Bo}_{i,k} = r_{i,k} + \tfrac{\mathbf{Q}_{i,k-1}}{\mathbf{Q}_{i,k}} (x^{fin}_{i,k-1} - \mathbf{Bo}_{i,k-1}) \tag{30}$$

$$\forall i, k \leqslant k^0, \quad x^f_i \leqslant x^{fin}_{i,k} + \overline{S}_i (1 + o_{i,k+1} - o_{i,k}) \tag{31}$$

$$\forall i, \quad 0 \leqslant x^f_i \leqslant \overline{S}_i \tag{32}$$

$$d \in \{0,1\}^N, r, p, x \geqslant 0 \tag{33}$$

This formulation provides dual bounds for v^{det}_s, and thus allows to compute dual bounds for the whole ROADEF problem with proposition 6 using less continuous and binary variables:

Proposition 1. *For all $s \in \mathcal{S}$ and $k^0 \in \mathcal{K}$, $v^{rlx}_{s,k^0} \leqslant v^{det}_s$.*

Proof: Let $(d^*_{i,k,w}, r^*_{i,k}, p^*_{i,k,w}, p^*_{j,w}, x^*)$ be an optimal solution of the MIP defining v^{det}_s. To prove that $v^{rlx}_{s,k^0} \leqslant v^{det}_s$, we prove that we have a feasible solution of the MIP defining v^{rlx}_{s,k^0} with $o_{i,k} = d^*_{i,k,W}$, $d_{i,k,w} = d^*_{i,k,w}$ for $k \leqslant k^0$, $p_{j,w} = p^*_{j,w}$, $p_{i,k,w} = p^*_{i,k,w}$ for $k < k^0$ $p_{i,k^0,w} = \sum_{k \geqslant k^0} p^*_{i,k,w}$ and $x = x^*$. The feasibility of the constraints defining v^{rlx}_{s,k^0} with the same objective function will prove $v^{rlx}_{s,k^0} \leqslant v^s_2$.

(24) are the only constraints that are not trivially verified.

$0 \leqslant p^*_{i,k,w} \leqslant \mathbf{Pmax}^s_{j,w}(d_{i,k,w-\mathbf{Da}_{i,k}} - d_{i,k+1,w})$. It implies
$p_{i,k^0,w} = \sum_{k \geqslant k^0} p^*_{i,k,w} \leqslant \mathbf{Pmax}^s_{j,w} \sum_{k \geqslant k^0} (d_{i,k,w-\mathbf{Da}_{i,k}} - d_{i,k+1,w-\mathbf{Da}_{i,k+1}})$, as $d_{i,k+1,w-\mathbf{Da}_{i,k+1}} \leqslant d_{i,k+1,w}$.

The telescopic sum and $d_{i,K+1,w} = 0$ imply (24). □

5 Dual bounds by time-step aggregation

In this section, we prove that time step aggregation for the nuclear power plants allows to compute dual bounds under some assumptions. Whatever the MIP formulation chosen after section 3 or 4, we have a MIP giving dual bounds for the challenge EURO/ROADEF 2010 of the form where $y_t \geqslant 0$ consider only the production variables of T1 and T2 power plants, whereas $x \in X$ denotes the other variables which are not indexed with $t \in \mathcal{T}$:

$$v = \min_{x \in X, y \geqslant 0} \quad c^1 x + \sum_t \mathbf{D}^t c_t^2 y_t \tag{34}$$

$$s.t : T^1 x + \textstyle\sum_t \mathbf{D}^t W_s^1 y_t \geqslant h^1 \tag{35}$$

$$\forall t, \quad T^2 x + W^2 y_t \geqslant h_t^2 \tag{36}$$

An important point for the following results is that T^1, T^2, W^1, W^2 do not depend on t. Furthermore, $\mathbf{D}^t$ is constant in the dataset of the challenge, and we take the hypothesis that the T1 production costs c_t^2 are only depending on the weeks, so that we can define c_w^2 for all weeks with $c_{w_t}^2 = c_t^2$ for all t.

We define the MIP $\overline{v}$ as the aggregation of v where $\mathbf{D}^t$ and $\mathbf{D}^w$ denotes the constant durations of time steps and weeks, $\alpha = \frac{\mathbf{D}^t}{\mathbf{D}^w}$, and $\overline{h}_w^2 = \alpha \sum_{t, w_t = w} h_t^2$.

$$\overline{v} = \min_{x \in X, y \geqslant 0} \quad c^1 x + \sum_w \mathbf{D}^w \overline{c}_w^2 y_w \tag{37}$$

$$s.t : T^1 x + \textstyle\sum_w \mathbf{D}^w W^1 y_w \geqslant h^1 \tag{38}$$

$$\forall w, \quad T^2 x + W^2 y_w \geqslant \overline{h}_w^2 \tag{39}$$

Proposition 2. *Aggregating production time steps to weeks provide a dual bound for the disaggregated problem:* $\overline{v} \leqslant v$.

Proof: Let (x^*, y_t^*) be an optimal solution of the MIP (34-36). Let $\overline{y}_w^* = \alpha \sum_{t, w_t = w} y_t^*$, we have just to prove that $(x^*, \overline{y}_w^*)$ is a feasible solution for the MIP defining $\overline{v}$. Indeed, the objective cost associated with $(x^*, \overline{y}_w^*)$ would thus be greater than the optimum $\overline{v}$: $\overline{v} \leqslant c^1 x^* + \sum_w \overline{D}^w \overline{c}_w^2 \overline{y}_w^* = v$, noticing that $\sum_w \mathbf{D}^w \overline{c}_w^2 \overline{y}_w^* = \sum_w \frac{\mathbf{D}^t}{\alpha} \sum_{t, w_t = w} \overline{c}_{w_t}^2 \alpha y_t^* = \sum_t \mathbf{D}^t c_t^2 y_t^*$.

Constraints (38) are proven for $(x^*, \overline{y}_w^*)$ from (35) using:
$\sum_w \mathbf{D}^w W^1 y_w^* = W^1 \mathbf{D}^w \alpha \sum_w \sum_{t, w_t = w} y_t^* = W^1 \mathbf{D}^t \sum_t y_t^*$.
Thus $T^1 x^* + \sum_w \mathbf{D}^w W^1 y_w^* = T^1 x^* + \sum_t \mathbf{D}^t W^1 y_t^* \geqslant h^1$.

Constraints (39) are proven for all week w aggregating constraints related to time steps t with $w_t = w$ with weight $\mathbf{D}^t$:

$$\sum_{t, w_t = w} \mathbf{D}^t T^2 x^* + \sum_{t, w_t = w} \mathbf{D}^t W^2 y_t^* = \mathbf{D}_w T^2 x^* + \left(\sum_{t, w_t = w} \mathbf{D}^t W^2 y_t^* \right) \geqslant \sum_{t, w_t = w} \mathbf{D}^t h_t^2.$$

Hence, $\mathbf{D}_w T^2 x^* + \alpha \mathbf{D}^t W^2 y_w^* \geqslant \mathbf{D}_w \overline{h}_w^2$ which proves (39). □

6 Dual bounds by scenario decomposition

In this section, it is proven how to calculate dual bounds with deterministic computations. Therefore, we define the following MIP as a general expression for the MIP giving bound after section 4, 5 or 6, with exact preprocessing and possibly the lastly aggregation.

$$v^{sto} = \min_{x \in X, y \geqslant 0} c_x x + \sum_s \pi_s c_s y_s \tag{40}$$

$$s.t: \quad Ax \geqslant a \tag{41}$$

$$\forall s, T_s x + W y_s \geqslant h_s \tag{42}$$

In this denomination, x denotes the the first stage variables $d_{i,k,w}, r_{i,k}$, $y_s \geqslant 0$ gather the other continuous variables duplicated for all scenario s. We denote v_s^{det} the following deterministic MIP for all scenarios $s \in \mathcal{S}$:

$$v_s^{det} = \min_{x \in X, y \geqslant 0} c_x x + c_s y \tag{43}$$

$$s.t: \quad Ax \geqslant a \tag{44}$$

$$T_s x + W y \geqslant h_s^1 \tag{45}$$

Proposition 3. *We have $\sum_s \pi_s v_s^{det} \leqslant v^{sto} \leqslant v_0$. In other words, dual bounds for the whole ROADEF problem can be calculated with $|\mathcal{S}|$ independent parallel computations of dual bounds on reduced problem with single scenarios.*

Proof: We reformulate the problem duplicating first stage variables for all scenarios $x_s = x$ and using relation $\sum_{s \in \mathcal{S}} \pi_s = 1$:

$$\begin{aligned} \overline{v}_k = \min_{x_s \in X, y \geqslant 0} & \sum_s \pi_s c_x x_s + \sum_s \pi_s c_s y_s \\ s.t: & \quad Ax \geqslant a \\ \forall s, & \quad x^s = x \\ \forall s, & \quad Ax_s \geqslant a \\ \forall s, & \quad Tx + W y_s \geqslant h_s^1 \end{aligned}$$

Relaxing constraints $x^s = x$, we get a dual bound for v_1. This relaxation implies independent sub-problems for all scenarios:

$$\begin{aligned} \overline{v}_k \geqslant \min & \sum_s \pi_s c_x x^s + \sum_s c_s y_s \\ s.t: \forall s, & \quad Ax_s \geqslant a \\ \forall s, & \; Tx + \textstyle\sum_w W y_s \geqslant h_s^1 \end{aligned}$$

The sub-problem related to s has value $\pi_s v_s^{det}$, it proves $\sum_s \pi_s v_s^{det} \leqslant v^{sto}$. □

7 Computational results

Our implementation used OPL and Cplex version 12.5 to solve MIP and OPL script to compute iteratively successive MIP. Our experimentations were computed with a laptop running Linux Ubuntu 12.04 with an Intel Core2 Duo processor, 2.80GHz. MIP computations were limited to 1h. We used the dataset from the EURO/ROADEF 2010 challenge which is now completely public, we refer to [14]. Table 2 indicates the instances characteristics, instances B and X are representative of real-world size instances, whereas A was the qualification dataset of the challenge.

	I	J	K	S	T	W	varBin	LP	time	inf0	sup0	time	inf	sup	time
A1	10	11	6	10	1750	250	3892	0,17%	0,09	0,00%	0,00%	1,1	0,00%	0,00%	1,1
A2	18	21	6	20	1750	250	78896	0,22%	0,4	0,02%	0,00%	3,8	0,00%	0,00%	4,9
A3	18	21	6	20	1750	250	8162	0,36%	0,11	0,05%	0,00%	4,4	0,00%	0,00%	3
A4	30	31	6	30	1750	250	17465	0,95%	3,1	0,42%	0,09%	21,4	0,00%	0,00%	334
A5	28	31	6	30	1750	250	15357	0,90%	7,5	0,53%	0,44%	55	0,15%	0,00%	3600
B6	50	25	6	50	5817	277	24563	2,23%	11,2	0,61%	0,51%	64	0,13%	0,00%	3600
B7	48	27	6	50	5565	265	35768	2,90%	66,7	0,78%	1,99%	346	0,52%	0,98%	3600
B8	56	19	6	121	5817	277	69653	9,47%	244	7,95%	NS	3320	7,95%	NS	3600
B9	56	19	6	121	5817	277	69306	8,51%	350	6,69%	NS	3600	6,69%	NS	3600
B10	56	19	6	121	5565	265	29948	3,45%	14,7	0,66%	0,27%	77	0,07%	0,00%	3600
X11	50	25	6	50	5817	277	20081	1,65%	12,1	0,58%	0,46%	124	0,37%	0,01%	3600
X12	48	27	6	50	5523	263	27111	2,98%	21,6	0,58%	0,35%	145	0,20%	0,00%	3600
X13	56	19	6	121	5817	277	30154	2,61%	17	1,30%	1,56%	411	1,09%	0,22%	3600
X14	56	19	6	121	5817	277	30691	3,09%	25	1,22%	0,81%	267	0,89%	0,04%	3600
X15	56	19	6	121	5523	263	27233	3,60%	13,4	0,47%	0,45%	100	0,10%	0,00%	3600

Table 2. Characteristics of the instances and MIP convergences for one scenario and weekly time steps: cardinality of sets $\mathcal{I}, \mathcal{J}, \mathcal{K}, \mathcal{S}, \mathcal{T}$, number of binaries, comparison of the gap of dual and primal bounds to the best primal solution known after LP relaxation, after Cplex cuts at the root node, and in one hour MIP resolution

MIP resolution for single scenarios instances With Cplex 12.5, dual bounds could be computed for all instance with a single scenarios, as shown in Table 2. Hence, we had tractable dual bounds of the whole ROADEF problem computing $\sum_s \pi_s v_s^4$ where v_s^4 are lower bounds of v_s^{det} or v_{s,k^0}^{rlx} that are obtained in a LP or a MIP truncated resolution in 1 hour. Furthermore, we observed that the variable definition is crucial in the efficiency of the branching quality: our level variables $d_{i,k,w} \leqslant d_{i,k,w+1}$ imply better branching and MIP convergence than using binaries $x_{i,k,w} = d_{i,k,w} - d_{i,k,w-1}$ as in [12] with GUB constraints $\sum_w x_{i,k,w} \leqslant 1$. These results are coherent with [4, 15].

We observed significant improvement in the MIP convergence in truncated resolution time with a preprocessing to remove variables. A common preprocessing for the Challenge EURO/ROADEF was to reduce the time windows

and to remove T1 production variables. Exact and heuristic preprocessing are especially proposed in [12]. To compute dual bounds, data preprocessing must induce optimal variable fixing which restrict the preprocessing possibilities. The propagation of calculated minimal durations for production cycles (i, k) thanks to CT11 and max stock levels and fuel consumption allows to tighten the time windows. Furthermore, T1 production variables can be fixed bounding the T2 production and noticing that once the T2 production fixed, the optimisation of T1 production is equivalent to the resolution of independent knapsack problems for all scenarios s and time steps t.

Lower bounds for the dataset A None of the dataset A instances fulfils the hypothesis to apply the time step aggregation of section 5. Single scenario computations with disaggregated time steps were tractable to provide dual bounds. Table 3 compares the dual bounds for the ROADEF challenge computing dual bounds of $\sum_s \pi_s v_s^{det}$ with LP relaxation, and MIP computations truncated in 30 minutes, to the parametric bounds $\sum_s \pi_s v_{s,k^0}^{rlx}$ with $k^0 = 4$ and $k^0 = 5$ and MIP computations truncated in 30 minutes. Our dual heuristics improve significantly the former best lower bounds in [2]. The exact formulation gives better results than the parametric formulations of section 4, the difference between $v_{s,5}^{rlx}$ and v_s^{det} is mainly explained by the approximation of the remaining fuel cost. Comparing the dual bounds got computing lower bounds of v_s^{det} for all $s \in \mathcal{S}$ emphasises the impact of the cuts and branching of Cplex already shown with table 1.

	Primal Best	best [2]	$v_{s,4}^{rlx}$ MIP	$v_{s,5}^{rlx}$ MIP	v_s^{det} PL	v_s^{det} +cuts	v_s^{det} MIP	Dual Best	Gap
A1	169538M	2,35%	0,58%	0,12%	0,33%	0,13%	**0,08%**	169403M	**0,08%**
A2	146048M	4,15%	1,48%	0,45%	0,70%	0,41%	**0,31%**	145593M	**0,31%**
A3	154429M	3,87%	1,93%	0,81%	1,15%	0,58%	**0,47%**	153704M	**0,47%**
A4	111591M	8,30%	3,44%	2,46%	2,90%	2,05%	**1,60%**	109801M	**1,60%**
A5	125822M	10,6%	5,15%	4,14%	4,36%	3,77%	**3,54%**	121366M	**3,54%**
Set A	707428M	5,4%	2,32%	1,42%	1,71%	1,23%	**1,07%**	699869M	**1,07%**

Table 3. Dual bounds on the dataset A

Lower bounds for the dataset B and X For all the instances of the dataset B and W, the hypothesis holds to apply the time step aggregation of section 5. Lower bounds were computable in practice with the scenario decomposition with aggregated production time steps. Table 4 compares the dual bounds for the ROADEF challenge computing dual bounds of $\overline{v}_s^{det}$ with LP relaxation, and MIP computations truncated in 1 hour, to the bounds obtained with parametric simplified formulation of section 6 with $k^0 = 4$ and $k^0 = 5$, to the former best bounds of the state of the art in [2]. Our dual bounds outclass significantly the former best dual bounds of the literature. We note that the best bounds

of [2] were slightly better than the first parametric dual bounds with $k^0 = 0$, parameter $k^0 = 1$ already improved the bounds of [2].

With $1h$ limit for MIP computations, parametric formulations with $k^0 = 5$ gives better dual bounds than the exact MIP formulation for the most difficult instances. This seems paradoxical as sub-problems $\overline{v}^{rlx}_{s,5}$ converge to optimality to a worse value than $\overline{v}^{det}_{s}$. The approximation done with $k^0 = 5$ is slight, as economically interesting solutions avoid to schedule 6 outages. whereas the variables corresponding to $k^0 = 6$ are numerous, and handicap the MIP resolution. Computing $\overline{v}^{rlx}_{s,5}$ instead of $\overline{v}^{det}_{s}$ allows to accelerate significantly the MIP resolution, it reaches more advanced phases of the Branch&Bound convergence and thus better dual bounds in 1h.

	Primal Best	best [2]	$\overline{v}^{rlx}_{s,4}$ MIP	$\overline{v}^{rlx}_{s,5}$ MIP	$\overline{v}^{det}_{s}$ PL	$\overline{v}^{det}_{s}$ +cuts	$\overline{v}^{det}_{s}$ MIP	Dual Best	Gap
B6	83424M	16,5%	6,85%	5,36%	7,02%	5,72%	**5,30%**	79003M	**5,30%**
B7	81174M	15,5%	7,23%	**6,31%**	9,60%	8,94%	7,64%	76053M	**6,31%**
B8	81926M	23,6%	14,52%	**12,39%**	17,06%	15,51%	15,51%	71772M	**12,39%**
B9	81750M	21,7%	12,71%	**11,65%**	16,43%	15,21%	15,21%	72224M	**11,65%**
B10	77767M	18,0%	8,33%	**7,32%**	10,01%	8,27%	7,7%1	72075M	**7,32%**
Set B	406041M	19,1%	9,94%	8,61%	12,03%	10,74%	10,28%	372424M	**8,60%**
X11	79116M	15,4%	6,45%	6,01%	6,73%	5,86%	**5,58%**	74698M	**5,58%**
X12	77589M	14,2%	7,42%	**6,68%**	9,05%	7,49%	7,06%	72405M	**6,68%**
X13	76449M	18,7%	10,17%	8,09%	8,92%	8,12%	**7,78%**	70502M	**7,78%**
X14	76172M	17,2%	9,64%	**7,85%**	9,70%	8,79%	8,36%	70196M	**7,85%**
X15	75101M	17,6%	10,02%	**8,98%**	11,90%	9,67%	9,18%	68354M	**8,98%**
Set X	384427M	16,6%	8,72%	7,50%	9,23%	7,96%	7,57%	357396M	**7,35%**

Table 4. Dual bounds on the datasets B and X

8 Conclusions and perspectives

Conclusions New dual bounds for the 2010 EURO/ROADEF Challenge are computed with the restriction of the MIP problem sizes. The relaxation of CT6 and CT12 allows to have a MIP formulation declaring binary variables only for the decisions of outage dates. A parametric aggregation of outages leads to a parametric family of dual bounds. To deal with smaller problems, we proved that dual bounds can be computed with restrictions to deterministic computations and aggregated production time steps for the datasets B and X. This leads to tractable computations, outclassing significantly the former best dual bounds of the literature, justifying also the quality of the primal solutions of [6].

Perspectives This work offers new perspectives to improve the dual bounds using lighter partial relaxations. Other perspectives are to derive primal matheuristics from our dual bounds computations, using the partial solutions with time

step aggregation and scenario decomposition to build a primal solutions. This work offers also special perspectives to improve the approach developed in [12]. First, our variable definition improve the MIP branching which would be interesting for their MIP computations. The parametric formulation of section 4 would be interesting for the generation of Bender's cuts: having less continuous variables and less constraints in the Bender's sub-problem, the cut generation would be more stable. Low parameters k^0 furnish a very stable cut generation while $k^0 = 5$ would furnish good quality approximated cuts.

References

1. D. Anghinolfi, L. Gambardella, R. Montemanni et al. A matheuristic algorithm for a large-scale energy management problem. *LNCS* , 7116:173–181, 2012.
2. F Brandt, R Bauer, M Völker, and A Cardeneo. A constraint programming-based approach to a large-scale energy management problem with varied constraints. *Journal of Scheduling*, 16(6):629–648, 2013.
3. N. Dupin. *Modélisation et résolution de grands problèmes stochastiques combinatoires: application à la gestion de production d'électricité.* PhD thesis, 2015.
4. N. Dupin. Tighter MIP formulations for the discretised unit commitment problem with min-stop ramping constraints. *EURO J Comput Optim*, to appear.
5. A. Froger, M. Gendreau, J. Mendoza, E. Pinson, L-M. Rousseau. Maintenance scheduling in the electricity industry: A literature review. *European Journal of Operational Research* , 2015.
6. F. Gardi and K. Nouioua. Local Search for Mixed-Integer Nonlinear Optimization: a Methodology and an Application. *LNCS*, 6622:167–178, 2011.
7. H. Gavranović and M. Buljubasić. A hybrid approach combining local search and constraint programming for a large scale energy management problem. *RAIRO Operations Research*, 47(4):481–500, 2013.
8. S. Godskesen, T. Jensen, N. Kjeldsen, and R. Larsen. Solving a real-life, large-scale energy management problem. *Journal of Scheduling*, 16(6):567–583, 2013.
9. V. Jost and D. Savourey. A 0-1 integer linear programming approach to schedule outages of nuclear power plants. *Journal of Scheduling*, 16(6):551–566, 2013.
10. J. Lazic, S. Hanafi, N. Mladenovic, and D. Urozevic. Variable neighbourhood decomposition search for 0-1 mixed integer programs. *Computers&Operations Research*, 37(6):1055–1067, 2010.
11. Y. Li, O. Ergun, G.L. Nemhauser. A dual heuristic for mixed integer programming. *Operations Research Letters* 43(4): 411–417, 2015.
12. R. Lusby, L. Muller, and B. Petersen. A solution approach based on Benders decomposition for the preventive maintenance scheduling problem of a stochastic large-scale energy system. *Journal of Scheduling*, 16(6):605–628, 2013.
13. A. Rozenknopf, R. Wolfler Calvo, et al. Solving the electricity production planning problem by a column generation based heuristic. *Journal of Scheduling*, 2012.
14. T. Simovic et al. Challenge ROADEF/EURO 2010: a large-scale energy management problem with varied constraints. Technical report, EDF R&D, 2010.
15. S. Yıldız and J. P. Vielma. Incremental and encoding formulations for mixed integer programming. *Operations Research Letters*, 41(6):654–658, 2013.

Matheuristics for the Discrete Unit Commitment Problem with Min-Stop Ramping Constraints

Nicolas Dupin, El-Ghazali Talbi

Univ. Lille, UMR 9189 - CRIStAL - Centre de Recherche en Informatique Signal et Automatique de Lille, F-59000 Lille, France

Abstract. The discrete unit commitment problem with min-stop ramping constraints applies for thermal units. To face the operational time limit for resolution, this article derives efficient matheuristics from MIP formulations previously obtained. Constructive matheuristics defines variable fixing and relax-and-fix strategies. To improve known feasible solutions, a Variable Neighbourhood Search scheme is implemented using MIP neighbourhoods with variable fixing strategies. The resulting matheuristic outclasses significantly the MIP frontal resolution in the truncated time of the industrial application.

Keywords: Matheuristic, mixed integer programming, unit commitment problem, relax-and-fix heuristic, variable neighbourhood search.

1 Introduction

Unit commitment problem Energy management induces several level of optimisation problems from strategic decisions in an uncertain environment to daily production decisions, we refer to [20]. Electricity production problems are gathered in the Unit Commitment (UC) problems category, dispatching at lesser-cost the generated productions to match supply and demand. A large taxonomy of UC problems exists, depending on the time horizon and on power generation types. Short term UC problems provide production decisions to be operated, which requires to have a fine description of technical operating constraints like [2], whereas long term UC problems provide strategic decisions in an uncertain environment, with an aggregated description of operating constraints, like [14].

Industrial motivations This article was motivated by daily UC problem considerations, to optimize real-time adjustments to match production and demand. Hydraulic units are appropriate for real-time adjustments of production thanks to their high modulation capacities. The motivating study concerns the ability of the French thermal fleet to participate in real-time adjustments. A strong limiting factor comes from the *minimum stop* and *ramping* constraints illustrated in Figure 2. Furthermore, operational production plans of thermal units follow discrete values in the French Utility company. The applicative context required

thus to study the original problem UCPd, a discretized UC problem with minimum stop and ramping constraints for thermal units. A specific MIP formulation has been developed in [9], with a polyhedron formulation work. This paper seeks to find the best solutions in the short truncated resolution time required by the operational application.

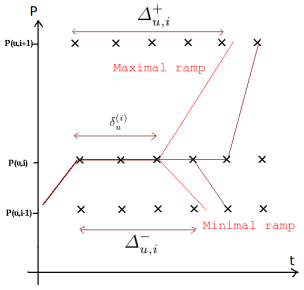


Fig. 1. Stop constraints

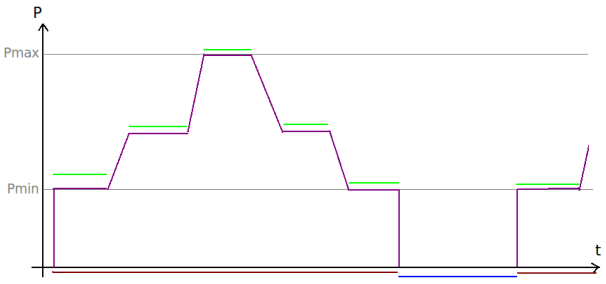


Fig. 2. Minimum stops on operating power points

Resolution of thermal UC MIP is an appropriate framework to handle operational constraints for UC. Formerly, MIP models of UC problems were commonly solved with Lagrangian decomposition, giving rise to Lagrangian heuristics like in [7]. Recent advances in MIP increased significantly the frontal resolution limits which benefits UCP, we refer to [2, 4]. This induced significant academic work on MIP formulations and polyhedral studies, we refer to [2, 5, 18]. UCPd is very close to the operational resolution that is operated in the French Utility company for thermal units, we refer to [7, 20]. The resolution method on the whole French fleet (nuclear, thermal and hydro power plants mainly) is presented in [7],it is a Lagrangian approach dualizing demand constraints. Each thermal unit induces a decoupled sub-problem after the dualization, for a dynamic programming resolution with discretized power levels and ramping constraints.

MIP primal heuristics Heuristic ideas implemented in MIP solvers led to significant improvement for the practical efficiency of the Branch& Bound (B& B) algorithm (we refer to [3] for a survey). Constructive heuristics are based on the continuous solutions given with Linear Programming (LP) relaxation, with rounding operations, Feasibility Pump (FP) ([10]) or with diving strategies in the branching operations (we refer to [1]). Once a feasible solution is found, B&B primal heuristics inspired from local search algorithms led to significant improvement, considering a small optimization problem around the incumbent. RINS heuristic (Relaxation Induced Neighbourhood Search, see [6]) fixes the integer variables which are common in the incumbent and in the continuous solution. Local branching strategies - introduced in [11] - implements a traditional local search using the incumbent, defining neighbourhoods with a maximal number of modifications in the latter.

Variable Neighbourhood Search In this paper, we focus on Variable Neighbourhood Search (VNS) methodology to escape local extrema. VNS was first proposed in [15]. Since, it has rapidly developed both in its variants and applications, we refer to [12, 16]. VNS considers different types of neighbourhoods and changes systematically the neighbourhood within the local search. The way to escape from a local optimum relative to a neighbourhood is to consider another neighbourhood where the current solution can be improved. VNS ideas can improve Local Branching, as developed in [17]. Furthermore, a variant of VNS - Variable Neighbourhood Decomposition Search (VNDS) introduced in [13] - allowed to improve MIP generic primal heuristics incorporating VNS ideas. An other way to couple VNS and MIP is to define large MIP neighbourhoods in a VNS scheme, like in [8, 21]. We note that VNS was already successfully applied to UC problems in [8, 22].

Paper outline This paper is organized as follows. In section 2, we introduce the problem UCPd. In section 3, we present the MIP compact formulations for UCPd used for the math-heuristics. In section 4, constructive matheuristics are obtained with variable fixing and Relax-and-Fix strategies. In section 5, a Variable Neighbourhood Search scheme is presented using MIP neighbourhoods. In section 6, the computational results are discussed. In section 7, our contributions are summarized, discussing also future directions of research. We refer to Table 1 for the notations.

$u \in \mathcal{U}$	index and set to designate generating units.
$t \in \mathcal{T}$	index and set for optimization time steps, $t = 0$ is initial period.
$i \in \mathcal{I}_u$	index and set for operating points of unit u, $i = 0$ is null power.
i_u^0	Initial point at $t = 0$ for unit u.
N_u	Number of operating points for unit u.
$P_{u,i}$	Power generated by unit u at point i.
$R^1_{u,i}$	Maximal capacity in primary reserve for unit u at point i.
$R^2_{u,i}$	Maximal capacity in secondary reserve for unit u at point i.
Δ_u^{off}	Minimum down time for unit u.
Δ_u^{on}	Minimum up time for unit u.
$\Delta^+_{u,i}$	Minimum stop time at i for unit u before ramping up to $i+1$.
$\Delta^-_{u,i}$	Minimum stop at i for unit u before ramping down to $i-1$.
D_t^P	(Forecast) demand in power for period t.
D_t^{R1}	Demand in primary reserve for period t.
D_t^{R2}	Demand in secondary reserve for period t.
C_u^S	Start-up cost for unit u.
C_u^F	Fixed cost whenever unit u is up.
C_u^P	Proportional cost to the power generated by unit u.

Table 1. Notations for the problem UCPd

2 UCPd, a discrete Unit Commitment Problem

The originality of the UC problem UCPd is that the production domain is discrete, production points are called *operating points*. Ramping constraints and minimum stop constraints illustrated in Figure 2 are aggregated in the operating points, as illustrated in Figure 1. UCPd focuses on a short term UC problem describing precisely operating constraints of thermal units. We consider that units $u \in \mathcal{U}$ are generating independently. This hypothesis holds for thermal units, not for hydraulic units. The objective function of thermal units gathers start-up costs, online set-up costs, and proportional costs to the generated power.

Production possibilities are discrete, whereas demands remain continuous. To face such difficulties, we write demand constraints as inequalities rather than equalities, producing at least the demands at any time. Minimizing production costs dissuades to over-generate power, which justifies such hypotheses. We consider also two types of reserves, namely primary and secondary reserves, to equalize real-time production and demands. Primary and secondary reserves differ in the operating delays. Reserve constraints are modelled similarly to power demands: every unit has limited capacities in each reserve in its operating points. The planning must also fulfil at least imposed reserve demands at any time. It implies a multi-knapsack structure for all time step t. To model possible power variations, three types of dynamic constraints are considered for UCPd:

- *Min up - min down* constraints: every unit u has a minimum up time Δ_u^{on} and a minimum down time Δ_u^{off}, exactly the same constraints in [19].
- *Dynamic transitions*: When a unit generates at an operating point i at period t, the allowable transitions for period $t+1$ are either to keep operating at point i or to shift to a neighbour point $j \in \{i-1, i+1\}$.
- *Min stops on operating points*: Once unit u is on point i, the power must be stabilized during $\Delta_{u,i}^{+}$ (resp. $\Delta_{u,i}^{-}$) time steps, before reaching point $i+1$ (resp. $i-1$). These minimal durations on operating points depend on the next move, combining a min-up time for all operation points and the required time to reach the other operating point as illustrated in Figure 1. We note that at point 0, we generalize min-down constraints, $\Delta_u^{off} = \Delta_{u,0}^{+}$.

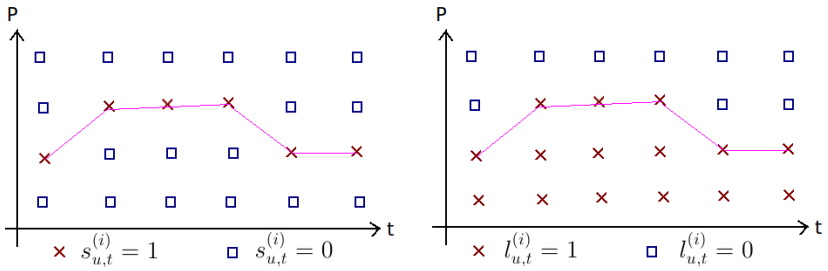


Fig. 3. State variables vs Level variables

3 MIP formulation

In [9], several tight and polyhedrally equivalent MIP formulations were provided with two variants of variable definition illustrated in Figure 3. For efficient MIP heuristics and branching as stated in [9], we use in this article variables $x_{u,t}^{(i)} \in \{0,1\}$, referred to as *level variables*, $x_{u,t}^{(i)} = 1$ indicating that unit u generates power at operating point i or upper. We extend with $x_{u,t}^{(0)} = 1$, $x_{u,t}^{(N_u+1)} = 0$. The alternative in [9] was to define *state variables*, $s_{u,t}^{(i)}$, with $s_{u,t}^{(i)} = 1$ if unit u generates exactly at operating point i. Start-up variables $y_{u,t}^{(i)-}$ and $y_{u,t}^{(i)+}$ are also introduced to have tight formulations, to have following MIP formulation for the problem UCPd:

$$\min_{x,y} \quad \sum_{u,t} C_u^F x_{u,t}^{(1)} + C_u^S y_{u,t}^{(1)+} + \sum_{u,t,i} C_u^P \qquad (P_{u,i} - P_{u,i-1}) x_{u,t}^{(i)} \quad (1)$$

$$\forall u, i, \quad x_{u,0}^{(i)} = \mathbb{1}_{i_u^0 \geqslant i} \quad (2)$$

$$\forall u, t, i, \quad x_{u,t}^{(i)} \leqslant x_{u,t+1}^{(i-1)} \quad (3)$$

$$\forall u, t, i, \quad x_{u,t}^{(i)} \geqslant x_{u,t+1}^{(i+1)} \quad (4)$$

$$\forall u, t, i, \quad x_{u,t}^{(i)} - x_{u,t-1}^{(i)} + y_{u,t}^{(i-1)-} = y_{u,t}^{(i)+} \quad (5)$$

$$\forall u, t, \quad \sum_{t'=t-\Delta_u^{on}+1}^{t} y_{u,t'}^{(1)+} \leqslant x_{u,t}^{(1)} \quad (6)$$

$$\forall u, t, \quad \sum_{t'=t-\Delta_u^{off}+1}^{t} y_{u,t'}^{(1)+} \leqslant 1 - x_{u,t-l_u}^{(1)} \quad (7)$$

$$\forall u, t, i, \quad \sum_{t'=t+1}^{t+\Delta_{u,i}^+} y_{u,t'}^{(i+1)+} + \sum_{t'=t+1}^{t+\Delta_{u,i}^-} y_{u,t'}^{(i-1)-} + x_{u,t}^{(i+1)} \leqslant x_{u,t}^{(i)} \quad (8)$$

$$\forall t, \quad \sum_{u,i} (P_{u,i} - P_{u,i-1})\ x_{u,t}^{(i)} \geqslant D_t^P \quad (9)$$

$$\forall t, \quad \sum_{u,i} (R_{u,i}^1 - R_{u,i-1}^1)\ x_{u,t}^{(i)} \geqslant D_t^{R1} \quad (10)$$

$$\forall t, \quad \sum_{u,i} (R_{u,i}^2 - R_{u,i-1}^2)\ x_{u,t}^{(i)} \geqslant D_t^{R2} \quad (11)$$

$$\forall u, t, i, \quad y_{u,t}^{(i)+}, y_{u,t}^{(i)-}, x_{u,t}^{(i)} \in \{0,1\} \quad (12)$$

The objective to minimize (1) is a linear combination of fixed costs, start-up costs and proportional costs to the power generated. Constraints (2) are implied by the level variables definitions. *Min up - min-down* constraints are (6),(7), using formulation in [19] with set-up $x_{u,t} = x_{u,t}^{(1)}$. The *transitions* constraints are coded with (3-4). (8) express *minimal stops on operating points*. A start-up $y_{u,t}^{(i)+} = 1$ implies on the next periods $t \leqslant t' \leqslant t + \Delta_{u,i}^+$, $x_{u,t'}^{(i)} = 1$ and $x_{u,t'}^{(i+1)} = 0$, giving rise to constraints (8). (9), (10) and (11) are demand constraints in generated power and reserves.

4 Constructive primal matheuristics

The LP relaxation has high quality as shown in [9]. It suggests to use continuous relaxations to build constructive matheuristics, with variable fixing or Relax-and-Fix heuristic strategies.

Variable fixing heuristics Following fixing strategies using the LP relaxed solutions $\tilde{x}_{u,t}^{(i)}$ were implemented:

- `hFix01`: For all variable such that $\tilde{x}_{u,t}^{(i)} \in \{0,1\}$, we fix $x_{u,t}^{(i)} = \tilde{x}_{u,t}^{(i)}$.
- `hFix0` : For all variable such that $\tilde{x}_{u,t}^{(i)} = 0$, we fix $x_{u,t}^{(i)} = 0$.
- `hFct01` : Same strategy as `hFix1`, applying only for $i = 1$.
- `hFct0` : Same strategy as `hFix0`, applying only for $i = 1$.
- `hSelect` : we remove the power plants u in which production is systematically null in the continuous relaxation, ie such as for all t, $\tilde{x}_{u,t}^{(i)} = 0$.
- `hTube` : power plants are first removed using `hSelect`. For the remaining units, if $\tilde{x}_{u,t}^{(i+1)} = 1$ (resp $\tilde{x}_{u,t}^{(i-1)} = 0$), we fix $x_{u,t}^{(i)} = 1$ (resp 0).
- `hTube3` : we consider three scenario for the demands :the real one, an increased demand and an underestimated demand. For these three scenario we calculate the continuous relaxation, and we fix the common variables of the `hTube` strategy fixation.

Noting $a > b$ that fixations of a contains fixations of b, `hFix01` > `hFct01` > `hFct0` and `hFix01` > `hFix0` > `hTube` > `hTube3` > `hSelect`.

Relax and Fix heuristics Decomposition approaches were also implemented, to optimize successively easier MIPs. Decompositions truncate the problem and/or relax continuously a subset of variables to have easier resolutions as follows:

- The `dcpDay` strategy fixes the first day production, relaxing continuously the variables relative to the second day. Then we resolve the MIP on the second day with the remaining variables.
- The `dcpLev` strategy builds a solution level by level following index i. Once variables with index $i' < i$ are fixed, variables $x_{u,t}^{(i)}$ are calculated using the MIP with variables $x_{u,t}^{(i')}$ integer for $i' \leqslant i$ and fixed to their previous value for $i' < i$. Variables $x_{u,t}^{(i')}$ for $i' > i$ are continuously relaxed.
- `dcpMid` strategy also proceeds level by level, following index i. First iteration consider only the variables of the middle operating points $Mid(u) = \lfloor \frac{N_{fct}(u)}{2} \rfloor$, with other variables continuously relaxed. A middle variable fixed to 0 (resp 1) involves indeed that the variables of the upper levels (lower respectively) are also fixed. with constraints $x_{u,t}^{(i)} \geqslant x_{u,t}^{(i+1)}$.

Adding more flexibility with slight modifications of the previously fixed decisions improves the solution quality and avoids infeasibilities. With `dcpMid` or `dcpLev` strategies, only *1*-stable variables are fixed, as defined below. With `dcpDay`, the first step considers binary variables for $t \in [1, 54]$ and fixes the variables for $t \in [1, 42]$ (assuming $T = 96$).

Definition 1 (k-stable variable) *k-stable variables* $x_{u,t}^{(i)}$ *have incumbent values* $x_{u,t-k}^{(i)} = \cdots = x_{u,t-1}^{(i)} = x_{u,t}^{(i)} = x_{u,t+1}^{(i)} = \cdots = x_{u,t+k}^{(i)}$.

A parallelisable matheuristics `dcpMixt` strategy combines `hSelect`, `dcpDay` and `dcpMid`, in 4 steps computing independent MIP, as described in Algorithm 1.

Algorithm 1: Parallelisable matheuristic decomposition

First step: Variable elimination `hSelect` unit selection using the LP relaxation.
Second step : Two independent MIP are computed:
MIP 2.1: only variables $x_{u,t}^{Mid(u)}$ are integer for $t \in [\![1, 48]\!]$.
MIP 2.2: only variables $x_{u,t}^{Mid(u)}$ are integer for $t \in [\![49, 96]\!]$.
Third step : Three independent MIP are computed:
MIP 3.1: only variables $x_{u,t}^{(i)}$ are integer for $t \in [\![1, 32]\!]$.
Integer 1-stable variables of MIP 2.1 are fixed to their optimal value in MIP 2.1.
MIP 3.2: only variables $x_{u,t}^{(i)}$ are integer for $t \in [\![65, 96]\!]$.
Integer 1-stable variables of MIP 2.2 are fixed to their optimal value in MIP 2.2.
MIP 3.3: only variables $x_{u,t}^{Mid(u)}$ are integer.
The optimal values in MIP 2.1 (resp 2.2) are fixed for $t \in [\![1, 36]\!]$ (resp $t \in [\![65, 96]\!]$).
Final step: the partial solutions are merged in one computation fixing:
• $x_{u,t}^{(i)}$ for $t \in [\![1, 24]\!]$ (resp $t \in [\![79, 96]\!]$) are fixed to its value in MIP 3.1 (resp 3.2).
• $x_{u,t}^{(Mid(u))}$ for $t \in [\![25, 32]\!]$ (resp $t \in [\![65, 78]\!]$) are fixed if they are 1-stable and equal in the optimal values in MIP 3.3 and MIP 3.1 (resp MIP 3.2).
• $x_{u,t}^{(Mid(u))}$ for $t \in [\![33, 64]\!]$ are fixed only if their optimal value in MIP 3.3 are 1-stable.

5 VND local search with MIP neighbourhoods

Once a feasible solution is built with previous constructive matheuristics, a Variable Neighbourhood Descent (VND) can improve primal solutions with local iterations computing sub MIPs.

General ideas VND iterates in a local search computing iterations with B&B resolution with MIP neighbourhoods. The current solution is the primal solution given by the last B&B resolution and it is also defined as warmstart for the next B&B resolution to improve the efficiency of B&B primal heuristics, enabling RINS or Local Branching heuristics from the beginning. This ensures that the solution given by the MIP resolution is at least as good as the current solution, this algorithm is thus a steepest descent algorithm. The stopping criterion could be a maximal time limit or a maximal number of iterations, or being in a local extremum for all neighbourhoods. The key point is the neighbourhoods definition and description. Neighbourhoods are defined with three characteristics:

– The **restriction of search space**: Extra constraints that the current solution satisfies to have an easier B&B resolution. A canonical example is to fix some variables to their value in the incumbent.

- a **B&B stopping criterion**: it must defined so that the B&B resolution is efficient in a short resolution time, for instance limitations on resolution time, but also in number of nodes or absolute or relative gaps between primal and dual bounds.
- a **specific parametrization of the MIP resolution**: for an efficient B&B resolution in the defined time limit. Easy neighbourhoods need no specific MIP parametrization, for a better efficiency in short resolution time for difficult sub MIPs, emphasizing the heuristic search with `mipemphasis` parameter, disabling or limiting cutting plane passes with `cutpass` parameter (in the terminology of Cplex).

Algorithm 2: Variable Neighbourhood Descent with MIP neighbourhoods

Input: an initial solutions, a set and order of neighbourhoods to explore
Initialisation: `currentSolution` = initSolution, $\mathcal{N}$ =initial neighbourhood.
while the stopping criterion is not met
    define the MIP with incumbent `currentSol` and the neighbourhood $\mathcal{N}$)
    define `currentSol` as warmstart
    `currentSol` = solveMIP(MIP,timeLimit($\mathcal{N}$))
    $\mathcal{N}$ = `nextNeighborhood`($\mathcal{N}$)
end while
return CurrentSolution

MIP neighbourhoods Multiple types of large and variable neighbourhoods can be defined for UCPd. Generic neighbourhoods are first derived from [6, 11]:

- $\mathcal{N}^{rins}$: Similarly to RINS heuristic [6], variables are fixed if they are a common integer value in the LP relaxation and in the current solution.
- $\mathcal{N}_k^{LB}$: for k an integer, k-Local Branching neighbourhood allows only k modifications to the incumbent, adding constraints: $\sum_{i:x_i^0=1} x_i + \sum_{i:x_i^0=0} (1-x_i) \leqslant k$

The multi-index structure allows to define partitioning neighbourhoods:

- $\mathcal{N}_U^{units}$ unit selection: only units $U \in \mathcal{U}$ are reoptimised. `hSelect` strategy can be used in the incumbent solution.
- $\mathcal{N}_{[\underline{t},\overline{t}]}^{time}$: all units are reoptimised in the time window $[\underline{t},\overline{t}]$. Peaks periods seem interesting choices of reoptimising time windows.
- $\mathcal{N}_i^{level}$: variables relative to the power level i are fixed. fixing $x_{u,t}^{(i)} = 0$ implies $x_{u,t}^{(i')} = 0$ for all $i' > i$ and fixing $x_{u,t}^{(i)} = 1$ implies $x_{u,t}^{(i')} = 1$ for all $i' < i$.

Some dynamic structures allows to define specific neighbourhoods:

- $\mathcal{N}_k^{shift}$: a variable $x_{u,t}^{(i)}$ is fixed if it is k-stable in the incumbent.

– $\mathcal{N}^{tube}$: *Tube neighbourhoods*: similarly with `hTube`, a variable $x_{u,t}^{(i)}$ is fixed if in the incumbent $x_{u,t}^{(i)} = x_{u,t}^{(i-1)} = x_{u,t+1}^{(i+1)}$. We note that (3),(4) imply $\mathcal{N}_1^{shift} \subset \mathcal{N}^{tube}$.

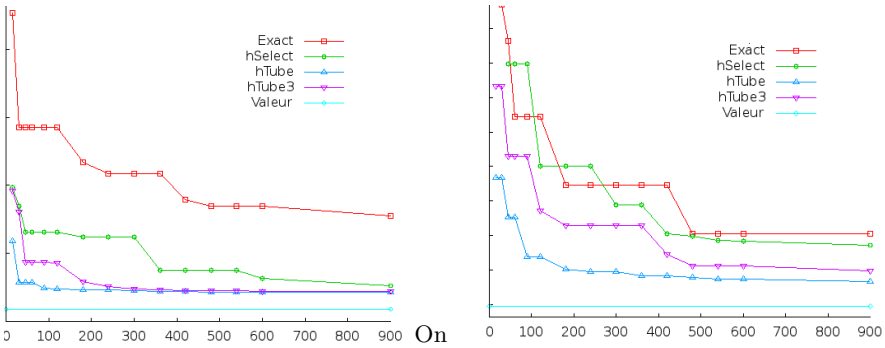
We note that a multiplicity of neighbourhood can be derived taking the intersection or the union of several previous neighbourhoods.

Neighbourhood selection and order The iterative order of neighbourhood is generally a crucial parametrization for VNS algorithms. Here, multiple types of Neighbourhood with search space restrictions are defined. Note that a single search space restriction can define several neighbourhoods, using different B&B resolution times: for a difficult MIP, one can seek to explore quickly a large spacial neighbourhood with MIP primal heuristics of the solver in a short resolution time, or wait to find better solutions after some branching.

The choice of neighbourhood order depends on the stopping criterion of the VNS. Having a infinite resolution time and seeking to find local minima of the VND (local minima for all the used neighbourhoods), the implementation selects dominant neighbourhoods and compute them all alternatively while no neighbourhood can improve the current solution. For a resolution in short time limits, the multiplicity of neighbourhoods incites to select the neighbourhoods for their efficiency cost improvement/resolution time. Parallelisation is a key point for the best efficiency of the VND exploring simultaneously different neighbourhoods.

6 Computational results

The matheuristics were implemented using the OPL interface to Cplex 12.5 to solve MIP and OPL script to iterate successive MIP problems. Tests were computed with a laptop Intel Core2 Duo processor, 2.80GHz, running Linux Ubuntu, using the dataset from [9].



On instance D80a

On instance D82b

Fig. 4. Convergence comparison variable fixing heuristics/exact resolution

Variable fixing results The results of variable fixing heuristics illustrate the compromise between calculation time and solution quality. `hFix01` leads often to infeasibilities. `hFix0` lessen infeasibilities, computing very quickly solutions with significant over-costs. `hFct01`, `hFct0` improves the convergence values inducing still a significant over-cost to the best solution known,but with a long convergence. `hSelect`, `hTube3` and `hTube` were very interesting, outmatching frontal convergence in time limits: the tiny approximation on the convergence value is balanced with the resolution acceleration, reaching more advanced phases of the B&B convergence in defined time limits up to 15 minutes as shown in Table 5 and Figure 4. the good quality of `hSelect` variable fixing relies on the fact that power plants with a prohibitive production cost are never activated in the LP relaxation, preferring cheaper production means to fulfil the demands. In 15 minutes, `hTube` gives the best solutions. Even after several days of calculation, variable fixing heuristics often outmatches the exact B&B resolution.

Relax-and-fix results Decomposition heuristics converge faster than variable fixing heuristics, but induce worse solutions. `dcpDay` leads sometimes to infeasibilities or prohibitive costs. In 5 minutes, `dcpMid` and `dcpMixt` are competitive, `dcpLev` results have a variable quality. In average, `dcpMixt` and `hTube` are the constructive heuristics giving the best results. Using several heuristics strategies in parallel taking the best solution is an implementation conclusion of this study, no approach outclasses universally the others.

Data	Frontal	hSelect	hTube	hTube3	dcpMid	dcpLev	dcpMix	VND
D67	2,24%	1,81%	**1,59%**	1,60%	1,99%	1,87%	1,97%	**1,57%**
D57	3,18%	1,76%	1,73%	**1,64%**	1,84%	1,60%	1,86%	1,75%
D30	0,42%	0,31%	0,06%	**0,03%**	0,66%	0,09%	0,62%	**0,03%**
D32a	1,02%	1,31%	0,65%	**0,36%**	1,24%	0,74%	1,15%	**0,21%**
D32b	0,56%	0,55%	0,30%	**0,16%**	0,46%	0,34%	0,50%	**0,00%**
D80a	1,20%	0,57%	0,25%	0,28%	0,42%	**0,16%**	0,42%	0,24%
D80b	0,70%	0,66%	0,69%	**0,52%**	0,93%	0,78%	0,97%	0,76%
D82a	1,39%	0,76%	0,33%	0,27%	0,64%	3,05%	**0,21%**	0,29%
D82b	0,89%	0,87%	0,16%	0,89%	0,38%	0,20%	**0,16%**	0,19%
Total	1,36%	0,98%	0,58%	0,69%	0,86%	1,26%	0,70%	**0,52%**

Fig. 5. Comparison of matheuristic solutions in 5 minutes to the best solution known

VND results With the diversity of MIP neighbourhoods, global industrial resolution time is to short to consider all neighbourhoods in the VND scheme, to compute a local extrema for all neighbourhoods. Neighbourhoods must be chosen empirically. Unit selection similar to `hSelect` was therefore always activated after the analyse of variable fixing results. $\mathcal{N}_1^{shift}$ and $\mathcal{N}^{rins}$ were very useful with a very fast convergence, and allowing to decrease very significantly the objective. *i-level neighbourhoods* on middle operating points provided significant improvements, requiring a little more computation time. The other neighbourhoods can

improve to the local extrema obtained after previously cited neighbourhoods, with a lower efficiency improvement/computation time. for a quick computation, it incites to choose neighbourhoods deterministically, using first efficient neighbourhoods, $\mathcal{N}_1^{shift}$ and $\mathcal{N}^{rins}$, *i-level neighbourhoods* on middle operating points adding flexibility combined with $\mathcal{N}_1^{shift}$ and $\mathcal{N}^{rins}$ while having a localal extrema for these neighbourhoods before trying the other neighbourhoods. The alternation of RINS with other neighbourhoods was very efficient.

The resulting VND was efficient in short resolution time, as we can see in Table 5, outmatching performance of frontal resolution and `hSelect` strategies, with similar performance with `hTube`. In some cases, it improved our former best solution, after long calculation time with variable fixing heuristics.

7 Conclusion and perspectives

Conclusions To improve the primal solutions of UCPd based on the exact MIP formulations, three approaches were successfully implemented. Variable fixing using strategies `hSelect`, `hTube3` or `hTube` imply few over-costs for optimal values in infinite resolution time, for a significant acceleration of the Branch&Bound convergence in truncated resolution time. Heuristic MIP decompositions increase significantly the Branch&Bound convergence, but with more approximations and over-costs. Once feasible solutions are known, VND local searches iterating with MIP fixations of variables present some advantages: the implementation is easy and generic, relying on the MIP formulation work, using MIP to ensure feasibility and cost, and allowing to define multiple types of large and variable neighbourhoods to imply high quality local minimums.

Perspectives This work opens new perspectives. The easy resolution (especially with decomposition heuristics and VND) allows to extend the size of the instances considering more units than just the thermal ones. An extension to hydraulic units is a challenging extension inducing a more difficult MIP resolution. Parallelisation is a crucial implementation point to improve the performances of the VND and to combine the advantages of the different constructive heuristics.

Acknowledgements

This paper was written in a PhD Thesis financed by the French Procurement Agency (DGA) of the French ministry of Defence. The authors address special thanks to Pierre-Edouard Adenot and Pierre Bazot for their support.

References

1. Achterberg, T., Koch, T., Martin, A.: Branching rules revisited. Operations Research Letters **33**(1), 42–54 (2005)
2. Arroyo, J., Carrion, M.: A computationally efficient mixed-integer linear formulation for the thermal unit commitment problem. IEEE Trans. on power systems **21**(3) (2006)
3. Berthold, T.: Primal heuristics for mixed integer programs. Master's thesis (2006)
4. Cong, G., Meyers, C., Rajan, D., Parriani, T.: Parallel Strategies for Solving Large Unit Commitment Problems in the California ISO Planning Model. In: IEEE Parallel and Distributed Processing Symposium (IPDPS), pp. 710–719 (2015)
5. Damcı-Kurt, P., Küçükyavuz, S., Rajan, D., Atamtürk, A.: A polyhedral study of production ramping. Mathematical Programming pp. 1–31 (2013)
6. Danna, E., Rothberg, E., Pape, C.L.: Exploring relaxation induced neighborhoods to improve MIP solutions. Math Programming **102**, 71–90 (2005)
7. Dubost, L., Gonzalez, R., Lemaréchal, C.: A primal-proximal heuristic applied to the French Unit-commitment problem. Math Programming **104**(1), 129–151 (2005)
8. Dupin, N.: Modélisation et résolution de grands problèmes stochastiques combinatoires: application à la gestion de production d'électricité. Ph.D. thesis, Lille 1 (2015)
9. Dupin, N.: Tighter MIP formulations for the discretized unit commitment problem with min-stop ramping constraints. EURO Journal of Computational Optimization (To appear)
10. Fischetti, M., Glover, F., Lodi, A.: The feasibility pump. Math Programming **104**(1), 91–104 (2005)
11. Fischetti, M., Lodi, A.: Local branching. Math Programming **98**(1-3), 23–47 (2003)
12. Hansen, P., Mladenović, N., Perez, J.M.: Variable neighbourhood search: Principles and applications. Annals of Operations Research **175**(1), 367–407 (2010)
13. Lazic, J., Hanafi, S., et al.: Variable neighbourhood decomposition search for 0-1 mixed integer programs. Computers & OR **37**(6), 1055–1067 (2010)
14. Lusby, R., Muller, L., Petersen, B.: A solution approach based on Benders decomposition for the preventive maintenance scheduling problem of a stochastic large-scale energy system. Journal of Scheduling **16**(6), 605–628 (2013)
15. Mladenović, N., Hansen, P.: Variable neighborhood search. Computers & OR **24**(11), 1097–1100 (1997)
16. Mladenović, N., Hansen, P.: Variable neighborhood search: Principles and applications. European Journal of Operational Research **130**, 449–467 (2001)
17. Mladenović, N., Hansen, P., Urozević, D.: Variable neighborhood search and local branching. Computers and Operations Research **33**(10), 3034–3045 (2006)
18. Morales-España, G., et al.: Tight and compact MILP formulation for the thermal unit commitment problem. IEEE Trans. on Pow. Syst. **28**(4), 4897–4908 (2013)
19. Rajan, D., Takriti, S.: Min-Up/Down Polytopes of the Unit Commitment Problem with Start-Up Costs. Tech. rep., IBM Research Report (2005)
20. Renaud, A.: Daily generation management at Electricité de France: from planning towards real time. IEEE Trans. on Automatic Control **38**(7), 1080–1093 (1993)
21. Schmid, V., Doerner, K., et al.: Hybridization of very large neighborhood search for ready-mixed concrete delivery problems. Computers & OR **37**(3), 559–574 (2010)
22. Todosijević, R., Mladenović, M., et al.: VNS based heuristic for solving the Unit Commitment problem. Elec. Notes in Discrete Mathematics **39**, 153–160 (2012)

A Large Neighbourhood Search Principle for Column-Oriented Models: Theoretical Derivation and Example Applications

Yixin Zhao, Torbjörn Larsson, and Elina Rönnberg

Department of Mathematics, Linköping University, Sweden
{yixin.zhao,torbjorn.larsson,elina.ronnberg}@liu.se

Both column generation methods and heuristic solution methods are frequently used approaches within the field of discrete optimisation. There is an increased interest in combining these approaches to efficiently solve large scale discrete optimisation problems; see [1] for an overview of proposed integrations of column generation and heuristics. This paper presents a large neighbourhood search principle that is adapted to discrete column-oriented models. The theoretical foundation for the method is a primal-dual optimality condition for such models. From this condition we derive an auxiliary problem that is solved by a large neighbourhood search strategy, where the repair method is to solve a column generation type subproblem.

An important characteristic of the proposed search method is that we are dealing directly with the integer program and that our column generation type subproblem takes both feasibility and near-optimality in the original discrete problem into account. We are relying on a fixed dual solution and are not repeatedly solving a linear programming master problem, which is otherwise typical for a column generation context.

1 Brief description of the method

We here give the theoretical background to, and the derivation of, the large neighbourhood search principle for discrete column-oriented models.

The primal-dual optimality condition presented is used to derive an auxiliary problem which is solved by a destroy-repair mechanism, where a specially designed column generation subproblem is used to repair a solution created by the destroy method, which removes a column from a current solution or reduces the number of times it is used.

We start with the following general column-oriented Integer Program (IP), where J is the index set of all columns, which is assumed to be very large.

$$z^* = \min \quad \sum_{j \in J} c_j x_j \tag{1a}$$

$$\text{s.t.} \quad \sum_{j \in J} A_j x_j \geq b \tag{1b}$$

$$x_j \geq 0 \text{ and integer}, \qquad j \in J \tag{1c}$$

Let $u \geq 0$ be a vector of Lagrangian dual variables associated with constraint (1b). When applying traditional column generation to the linear programming relaxation of

IP, one alternates between solving a restricted linear programming master problem and a column generation subproblem on the form

$$\min_{j \in J} \ (c_j - u^T A_j). \tag{2}$$

To begin the derivation of our column generation subproblem, we perform a Lagrangian relaxation of the constraint (1b) and define the Lagrangian dual function

$$h(u) = \min \quad \sum_{j \in J} c_j x_j + u^T \left(b - \sum_{j \in J} A_j x_j \right) \tag{3a}$$

$$\text{s.t.} \quad x_j \geq 0 \ \text{ and integer}, \ \ j \in J \tag{3b}$$

$$= b^T u + \begin{array}{l} \min \sum_{j \in J} \left(c_j - u^T A_j \right) x_j \\ \text{s.t.} \ \ x_j \geq 0 \ \text{ and integer}, \ \ j \in J. \end{array} \tag{3c}$$

For an integral $\widetilde{x} = \left(\widetilde{x}_j \right)_{j \in J} \geq 0$ and a $\widetilde{u} \geq 0$ we define

$$\varepsilon(\widetilde{x}, \widetilde{u}) = b^T \widetilde{u} + \sum_{j \in J} \left(c_j - \widetilde{u}^T A_j \right) \widetilde{x}_j - h(\widetilde{u}) \tag{4}$$

and

$$\delta(\widetilde{x}, \widetilde{u}) = \widetilde{u}^T \left(\sum_{j \in J} A_j \widetilde{x}_j - b \right). \tag{5}$$

The value of $\varepsilon(\widetilde{x}, \widetilde{u})$ can be interpreted as the degree of near-optimality of $\widetilde{x}$ in the Lagrangian relaxation for the dual solution $\widetilde{u}$, while the value of $\delta(\widetilde{x}, \widetilde{u})$ can be interpreted as the degree of near-complementarity of $\widetilde{x}$, with respect to $\widetilde{u}$.

The following theorem, which we provide a proof for in the full paper, is instrumental for deriving our column generation subproblem.

Theorem 1. *Let $\widetilde{u} \geq 0$ and assume it holds that $c_j - \widetilde{u}^T A_j \geq 0, \ j \in J$. Let the solution $\widetilde{x}$ be feasible in IP and let $\beta \geq 0$. Then $\widetilde{x}$ is β-optimal in IP if and only if*

$$\varepsilon(\widetilde{x}, \widetilde{u}) + \delta(\widetilde{x}, \widetilde{u}) \leq z^* - h(\widetilde{u}) + \beta.$$

An interpretation of this inequality is that the gap between the primal objective value $z^* + \beta$ and the lower bound $h(\widetilde{u})$ can be split into the two nonnegative terms. If $\widetilde{u}$ is an optimal Lagrangian dual solution and $\beta = 0$, then the right hand side of the inequality is the duality gap of IP. This primal-dual optimality condition is inspired by the results presented in [2], where the roles of the quantities corresponding to $\varepsilon(\widetilde{x}, \widetilde{u})$ and $\delta(\widetilde{x}, \widetilde{u})$ are central.

A conclusion that can be drawn is that in order to find a near-optimal solution to IP, three requirements must be fulfilled: (i) near-optimality in the Lagrangian relaxation, (ii) near-complementarity with respect to the relaxed constraint, and (iii) feasibility in IP. By taking a convex combination of the near-optimality in the Lagrangian relaxation

and the near-complementarity, dropping the constant terms, and penalising infeasibility, the following auxiliary problem for finding near-optimal solutions to IP is obtained.

$$\min \quad \alpha \sum_{j \in J} \left(c_j - \widetilde{u}^T A_j \right) x_j + (1-\alpha) \widetilde{u}^T \sum_{j \in J} A_j x_j + M e^T a \tag{6a}$$

$$\text{s.t.} \quad \sum_{j \in J} A_j x_j + a \geq b \tag{6b}$$

$$x_j \geq 0 \text{ and integer}, \qquad j \in J \tag{6c}$$

$$a \geq 0 \tag{6d}$$

Here, $\alpha \in (0, 1]$, e is a vector of ones, a is a vector of artificial variables, and M is a large positive number. By dividing the objective function by α, defining $\gamma = (2\alpha - 1)/\alpha$, redefining the value of M, and eliminating the artificial variables, the above problem is equivalent to

$$\min \quad \sum_{j \in J} \left(c_j - \gamma \widetilde{u}^T A_j \right) x_j + M e^T \max \left\{ 0, b - \sum_{j \in J} A_j x_j \right\} \tag{7a}$$

$$\text{s.t.} \quad x_j \geq 0 \text{ and integer}, \qquad j \in J, \tag{7b}$$

where the maximum operator is component-wise.

Now, consider a destroyed solution $\widetilde{x}$, which is nonnegative and integral, but not necesarily feasible with respect to constraint (1b). Let $J^R = \{j \in J \mid \widetilde{x}_j > 0\}$. In order to improve this destroyed solution, we wish to increase the value of one variable by one, and a best incremental change with respect to the problem (7a)–(7b) is then found by solving the problem

$$\min \quad \sum_{j \in J} \left(c_j - \gamma \widetilde{u}^T A_j \right) \left(\widetilde{x}_j + \delta_j \right) + M e^T \max \left\{ 0, b - \sum_{j \in J^R} A_j \left(\widetilde{x}_j + \delta_j \right) \right\} \tag{8a}$$

$$\text{s.t.} \quad \sum_{j \in J} \delta_j = 1 \tag{8b}$$

$$\delta_j \in \{0, 1\}, \qquad j \in J, \tag{8c}$$

that can be reduced to

$$\min_{j \in J} \left(c_j - \gamma \widetilde{u}^T A_j + M e^T \max \left\{ 0, b - \sum_{k \in J^R} A_k \widetilde{x}_k - A_j \right\} \right), \tag{9}$$

which is our column generation subproblem.

The subproblem will give us a column (c_l, A_l). If $l \notin J^R$, that is, if the column (c_l, A_l) was not previously used in the destroyed solution, then $J^R := J^R \cup \{l\}$ and $\widetilde{x}_l := 1$; otherwise, that is, if (c_l, A_l) is a copy of a column already used in the destroyed solution, then we only update the corresponding variable by setting $\widetilde{x}_l := \widetilde{x}_l + 1$.

In general, our large neighbourhood search principle for column-oriented models is composed of the following key elements. A *destroy method*, which can be randomized,

is designed to reduce the value of one or more variables $\widetilde{x}_j$, $j \in J^R$. Different destroy methods can be applied in different phases of the algorithm. A *repair method* is applied by solving a column generation type subproblem one or several times, where the exact design of the subproblem depend on the application at hand. Typically, restrictions can be imposed on the subproblem to escape local optima and diversify the search.

The near-optimal dual solution used in the algorithm can be found by applying for example linear programming column generation, Lagrangian relaxation and subgradient optimization, or some other appropriate dual method. The set of columns that define an initial solution can be found by, for example, applying the presented column generation principle from an empty solution, or by some other means.

2 To be presented in the full paper

The full paper begins with comparing the characteristics of our method with other principles for integrating column generation and heuristics and we especially comment on the similarities with diving heuristics. Thereafter follows a complete derivation of the method, including a proof for the theorem. This abstract considers only a general column-oriented formulation, but in the full paper we also address the special case of such a model resulting from a Dantzig-Wolfe reformulation of a block-angular structure.

As for any column generation scheme, our proposed large neighbourhood search principle needs to be adapted to the structure of the problem at hand. In the paper we study two applications: a bin packing problem (which has the structure described in this abstract), and a resource allocation problem (which has a block-angular structure). For both applications we derive a complete scheme and illustrate the computational properties on some instances.

It remains for future work to investigate how well this large neighbourhood search principle performs for various applications, and how details of the algorithm are best designed to achieve computational efficiency.

References

1. Raidl G.R.: Decomposition based hybrid metaheuristics. European Journal of Operational Research 244, 66–76 (2015).
2. Larsson T. and Patriksson M.: Global optimality conditions for discrete and nonconvex optimization—With applications to Lagrangian heuristics and column generation. Operations Research, 54, 436–453 (2006).

A Matheuristic Approach for Solving the 2-Connected Dominating Set Problem

Raka Jovanovic[1] and Stefan Voß[2,3]

[1] Qatar Environment and Energy Research Institute, Hamad bin Khalifa University, PO Box 5825, Doha, Qatar
[2] Institute of Information Systems, University of Hamburg, Von-Melle-Park 5, 20146 Hamburg, Germany
[3] Escuela de Ingenieria Industrial, Pontificia Universidad Católica de Valparaíso, Chile

A dominating set for a graph $G(V, E)$ is a subset of vertices $D \subseteq V$ that has the property that every vertex in G either belongs to D or is adjacent to a vertex in D. Finding the dominating set with the smallest possible cardinality among all dominating sets, for a graph, is one of the standard NP-complete problems [2]. In this work we focus on the problem of the 2-connected dominating set which has the additional property that there are two vertex disjoint paths between any two nodes $n, m \in D$. These paths consist only of nodes that that are elements of D.

To be more precise our goal is to develop a heuristic based method that manages to find near optimal solutions for the minimal 2-connected dominating set problem (2-CDSP). An example of a problem instance and solution for the 2-CDSP can be seen in Fig 1.

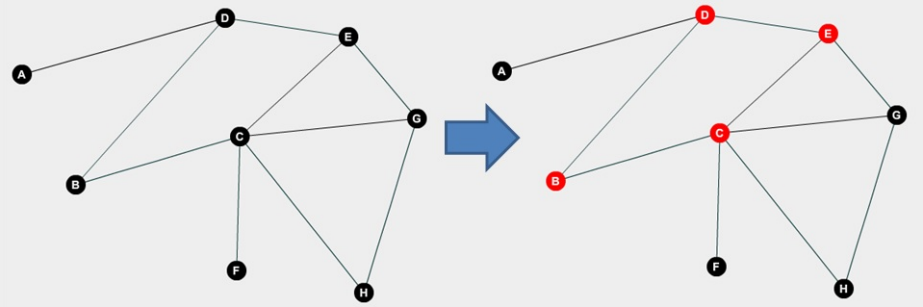


Fig. 1. Example of a problem instance (left) and solution (right) for the 2-CDS. In the solution the red nodes represent a minimal 2-connected dominating set.

The special focus is on finding such solution for large graphs. The dominating set problem (DSP) has been widely explored due to its close connection with wireless networks[6]. Several variations have been considered like the weighted [4] and connected version [2] of the problem. When modeling wireless and sensor networks, using graphs, the connected DSP (CDSP) gives us a significantly more

realistic representation of such systems when compared with the basic DSP. For the CDSP several different approaches have been used for finding either exact or approximate solutions [2].

In the recent years there has been an increased interest in the 2-connected version of the problem since it provides a higher level of failure resistance of the modeled wireless networks. A significant part of the published research has been dedicated to finding optimal solutions using mixed integer programming (MIP). The use of such models has a drawback that it only manages to find such solutions for relatively small graphs containing up to 200 nodes [1, 3]. On the other hand, relatively few works have been published on finding approximate solutions for the 2-CDSP. One example is the work by Shi et al. for application in wireless networks. [8]

In this paper we propose a matheuristic approach for solving this problem, but the general concept can be extended to many graph problems containing 2-connectivity constraints. The first reason for developing such an approach is the lack of an efficient algorithm for solving large scale problem instances. Secondly, MIP models have been extensively researched and have proven to be very suitable for solving the 2-CDSP this knowledge can be exploited in developing such a method. The general idea is to generate an initial solution using a randomized greedy approach and use a MIP model for local improvements. This concept has previously been successfully applied to the capacitated facility location problem [5]. In our method, a second heuristic function is used to select subsections of the existing solution for which there is a potential for improvement. The final step is solving such subsections to optimality using MIP. This type of approach has several advantages. First the greedy method manages to find initial solutions with low computational cost. Secondly, the subsections that are solved using MIP are selected in a way that the corresponding sub-problem is simpler than the 2-CDSP.

The heuristic method used for generating initial solutions is based on the existence of an open ear decomposition of a 2-connected graph [7]. To be more precise, we iteratively grow a bi-connected subgraph S by extending it with an open ear P. The extension of S is conducted in a greedy way, by selecting an ear P from a suitably defined list of candidates. Our goal is to make a minimal dominating set, so there is a preference for expanding S with the smallest ear that dominates the most nodes. Let us define the function $d(P)$ as the number of newly dominated nodes in G (nodes that are not adjacent to any $u \in S$) by P. The heuristic function used in the greedy algorithm is equal to $d(p)/|P|$, where the notation $|P|$ is used for the number of nodes in the ear P.

The local improvement of the generated dominating set D is done in the following way. A set of nodes L is selected in a way that $D \setminus L$ stays bi-connected. Let us define D_L as the set of nodes that have only been dominated by nodes in L. The improvement is performed on a set of nodes K containing nodes in D_L and all dominated nodes that are connected to D_L. Our goal is to select the minimal subset set of $I \subset K$ such that it dominates all nodes in D_L. The set I must also satisfy the constraint that and that there are two disjoint paths

containing only elements in I connecting it to $D \setminus L$. A MIP model based on multi-commodity flow is developed to calculate the optimal set I. In a preprocessing stage, additional constraints are calculated based on the relation between D and L to improve the efficiency of the model. Further, the sub-problem uses L as an initial feasible solution. The proposed matheuristic approach is compared to other methods based on MIP models and metaheuristic approaches on data sets previously used in literature and on large scale unit disk graphs.

References

[1] Buchanan, A., Sung, J.S., Butenko, S., Pasiliao, E.L.: An integer programming approach for fault-tolerant connected dominating sets. INFORMS Journal on Computing 27(1), 178–188 (2015)

[2] Du, D.Z., Wan, P.J.: Connected dominating set: theory and applications, vol. 77. Springer Science & Business Media (2012)

[3] do Forte, V.L., Lucena, A., Maculan, N.: Formulations for the minimum 2-connected dominating set problem. Electronic Notes in Discrete Mathematics 41, 415 – 422 (2013)

[4] Jovanovic, R., Tuba, M., Simian, D.: Ant colony optimization applied to minimum weight dominating set problem. In: Proceedings of the 12th WSEAS international conference on Automatic control, modelling & simulation. pp. 322–326. World Scientific and Engineering Academy and Society (WSEAS) (2010)

[5] Lagos, C., Guerrero, G., Cabrera, E., Niklander, S., Johnson, F., Paredes, F., Vega, J.: A matheuristic approach combining local search and mathematical programming. Scientific Programming (2016), Article ID 1506084, 7 pages, doi:10.1155/2016/1506084

[6] Li, Y., Wu, Y., Ai, C., Beyah, R.: On the construction of k-connected m-dominating sets in wireless networks. Journal of combinatorial optimization 23(1), 118–139 (2012)

[7] Robbins, H.E.: A theorem on graphs, with an application to a problem of traffic control. The American Mathematical Monthly 46(5), 281–283 (1939)

[8] Shi, Y., Zhang, Y., Zhang, Z., Wu, W.: A greedy algorithm for the minimum 2-connected m-fold dominating set problem. Journal of Combinatorial Optimization 31(1), 136–151 (2016)

A Variable Neighborhood Decomposition Search Applied to the Hierarchical Hub Location and Routing Problem

Fábio Francisco da Costa Fontes[1,2] and Gilles Goncalves[1]

[1] Univ. Artois, EA 3926, Laboratoire de Génie Informatique et d'Automatique de l'Artois (LGI2A), Béthune, F-62400, France
gilles.goncalves@univ-artois.fr
[2] Universidade Federal Rural do Semi-Árido Departamento de Ciências Exatas e Naturais CEP: 59625-900, Mossoró RN, Brazil
fabio_fontes@ufersa.edu.br

Abstract. In this work a Variable Neighborhood Decomposition Search Metaheuristic using exact methods is showed to solve a hub location and routing problem. Network is composed by a hierarchical structure with hub, sub-hub and spoke nodes representing liner shipping operations.

Keywords: VNDS metaheuristic, exact methods, hub location and routing problem, sub-hub.

1 Introduction

Network design is regarded as an important problem for development of an efficient and sustainable supply network.

Network design problem being solved optimally is achieved: to deploy, more easily, a supply service; to decrease the expenses with fuel, quantity of vehicles and crews; to meet the customers at a shorter time, allowing a greater customer satisfaction; and to decrease the CO_2 emissions.

Hub location and routing problem were studied by [1]. In that work a mathematical model for hub and spoke network with a new concept is developed by creating a hierarchical structure with sub-hubs. According to [2], such hub network design formulations are very difficult to solve.

Examples of analyzed networks are presented (Figure 1 and 2), where hub ports are represented by squares, spokes are the circle nodes, triangles represent sub-hub, dashed lines are the deep sea services and continuous lines represent short sea services.

Complex problems are presented by two models because they are also constituted by flow of goods at the network, where each node sends products to all others nodes. Model with sub-hubs increases the number of alternative paths to transport of goods, expanding also the complexity of problem.

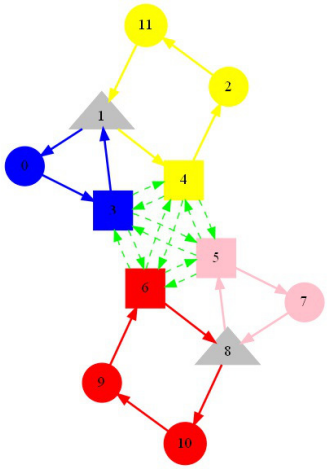


Fig. 1. Network with sub-hub

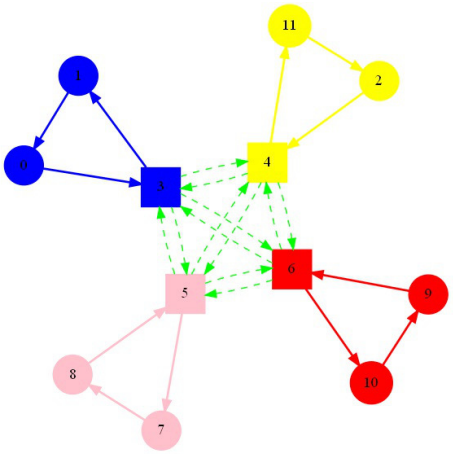


Fig. 2. Network without sub-hub

Due the complexity of studied problem, optimum results for instances with more than 12 nodes were not obtained at the realized tests with CPLEX. Therefore, a variant of Variable Neighborhood Search (VNS) metaheuristic known as Variable Neighborhood Decomposition Search (VNDS) [3] was implemented to analyze the two models with large instances.

2 Variable Neighborhood Decomposition Search

In this study, the problem is consisted by 4 sub-problems: hub and sub-hub location, spoke allocation, service design and network routing.

Operators of a Variable Neighborhood Decomposition Search (VNDS) were also implemented considering the characteristics those sub-problems. Moving between operators is guided by *neighborhood change sequential* procedure.

One operator was developed for location sub-problem . Concerning the sub-problem of spoke nodes allocation, two operators are proposed. For service design sub-problem two operators were developed. Finally, about the routing subproblem was used *Dijkstra algorithm*.

A Cyclic Variable Neighborhood Descendent (C-VND) procedure is used by *improve hub and improve sub-hub* operator. This procedure is executed in the step 2 (*Improvement Procedure*).

The set of neighborhood structures of Cyclic VND is composed by two neighborhood structures (*improve-hub* and *improve-sub-hub*).

At the *improve-hub* structure and for each cluster, the switch between each spoke node with its previous (original) hub (new hub is allocated) is tested. For each switch hub-spoke, hub route is built by the *Nearest Neighbor Algorithm*, allowing the sub-problem cost evaluation.

Algorithm 1 Variable Neighborhood Decomposition Search

Initialization
  Find a Initial solution $x \in S$ by a constructive heuristic
  Define the order of operators in the set $N = N_1, ..., N_{k_{max}}$
  $x_{actual} \leftarrow x$
repeat
  $k \leftarrow 1$
  while $k \leq k_{max}$ **do**
    step(1) - Getting subproblem(N_k):
      select exactly p solution attributes from previous solution x_{actual};
      denote this subset by y;
    step(2) - Solving subproblem:
      $y' \leftarrow$ Improvement procedure(y); //y' improved solution
    step(3) - Substituting the solution (it creates original problem solution):
      $x_{best} = (x_{actual} \setminus y) \bigcup y'$;
    step(4) - Making the problem cost evaluation:
      objective function $f(x_{best}) = f_A(x_{best}) + f_T(x_{best})$;
    step(5) - Moving between operators problem:
      Neighborhood change sequential $(x_{actual}, x_{best}, k)$;
  end while
  $x_{actual} \leftarrow Shake(x_{best}, N_{special})$
until the stopping condition is met

At *improve-sub-hub* structure, for each close clusters, the distance cost between each candidate sub-hub (spoke node from close clusters) and the hubs from close clusters is evaluated. Each spoke node is tested, trying to find a new sub-hub with distance cost less than actual sub-hub.

Change Neighborhood procedure used at the Cyclic VND is the Cyclic neighborhood change step.

Two route improve operator, at the step 2 (*Improvement Procedure*), two random and different routes are selected and, for each one, the *2-opt Algorithm* is applied. In this operator, the cost evaluated by the sub-problem is total cost of selected route.

2-opt Algorithm was implemented to select the neighbor solution of y with the smallest cost, independently if the global cost is better than the actual solution.

In *One route improve* operator, one route was created with the same steps as before. Just that, one random route is selected. Using this operator, it is allowed to escape local optimum achieved by the previous one.

Allocation sub-problem has been realized by an exact method: a p-median problem using CPLEX is solved in the proposed constructive heuristic. At the problem studied level, the allocation will be dependent not only of distance between hub and spoke, but also of the demands at network nodes.

Allocation spoke nodes operator was developed to improve the allocation through the whole network and not only inside each local cluster.

Using this operator, the transport cost is evaluated after select one random spoke node and to try insert it at a different cluster of its original cluster. After

testing each cluster, the best improvement will be accepted only if verified that it is better than the actual solution.

Allocation spoke node operator is closed after the first accepted best improvement or after an specific number of tests without any improvement.

Remember that the objective function in this problem is composed by minimize two parts, allocation cost and transport cost (line 16 in the algorithm).

Note that both transport cost and allocation cost are calculated at the step (4), but at the step (5), *Neighborhood change sequential* procedure, only the transport cost is considered. This was chosen because was observed that transport cost is more representative than allocation cost in the objective function.

For the step (4) of VNDS, a smallest path problem is necessary to be solved (routing subproblem) using *Dijkstra algorithm* for the transport cost evaluation. An exception is when the *allocation spoke node* operator is selected in the step (1), because the subproblem cost evaluation is exactly the transport cost.

Note that our proposed heuristic can be classified as a *matheuristic* because exact methods are used inside some procedures of the algorithm.

Shake procedure is also used in the VNS to diversify the search, getting away from a local optimum, and maybe, allowing to find the global optimum.

Operator used at shake procedure has the same structure of *allocation spoke node* operator, except that the first best improvement solution is accepted.

3 Conclusions

In this study, we want to check if the implemented matheuristic solves the analyzed problem in a efficient way in term of solution quality. Analysis were realized comparing the tests results found by the implemented matheuristic with results obtained by the CPLEX solver.

Instances given in [1] are also used in this study. Implemented matheuristic found good results for the problem.

Acknowledgement

The authors would like to thank the Coordenação de Aperfeiçoamento de Pessoal de Nivel Superior (CAPES), a foundation of the Ministry of Education of Brazil that finances a PhD scholarship for the development these studies.

References

1. Fabio Francisco da Costa Fontes and Gilles Goncalves. Hub location and routing problem with alternative paths. In *Advanced Logistics and Transport (ICALT), 2015 4th International Conference on*, pages 317–322. IEEE, 2015.
2. Teodor Gabriel Crainic, Kap Hwan Kim, et al. Intermodal transportation. *Transportation*, 14:467–537, 2006.
3. Pierre Hansen, Nenad Meladenovic, Raca Todosijevic, and Said Hanafi. Variable neighborhood search, basics and variantes. *Cahiers du GERAD*, 2016.

Combining Metaheuristics with Column Generation: Successful Approaches to Enhance Column Generation Algorithms Performance

Fabián Castaño[1], Marc Sevaux[2]

[1] Facultad de Ingeniería Industrial, Universidad Pontificia Bolivariana, Bucaramanga, Colombia
fabian.castano@upb.edu.co
[2] Lab-STICC, Université de Bretagne-Sud, Lorient, France
marc.evaux@univ-ubs.fr

Abstract. Column generation algorithms are typically adopted to address mathematical programming problems defined over a huge number of variables. This approach suffers, however, from several problems that might limit its usability. In this work some of these problems are discussed along with several strategies that take advantage of the use of (meta-)heuristics to help improve the methods performance and reduce the computational effort required to compute an optimal solution. In this work the benefits of using metaheuristic strategies within CG are discussed from the viewpoint of the way they, indirectly, address some of the causes leading to a poor performance. These different methods are tested by solving the maximum network lifetime problem in wireless sensor networks for which a model that naturally leads to column generation is considered. Experimental results show how the use of metaheuristics can generate large improvements on the performance of the basic column generation framework. cite **Keywords:** Column generation; Metaheuristics; Matheuristics;

1 Introduction

Column generation (CG) is an efficient method used to solve large scale linear programming problems that has been largely exploited to reduce the computational effort required to solve them [1, 3, 10]. CG essentially divides a problem in two namely a restricted master problem (RMP), and a pricing subproblem (PS). RMP represents the original problem formulated over a reduced part of the solution space expressed by a subset of columns (variables). Then, the PS is used to identify additional columns that, when added to RMP, can help to further improve the objective function of RMP.

CG is an iterative method that gradually enlarges the search space by adding new variables, not considered before, that contribute to improve the objective function for RMP. RMP is initialized with a reduced set of variables that generate an initial solution for the problem and sequentially adds new variables based on the reduced cost criterion. These new unknown columns are generated at every

new iteration by using any possible method useful to solve PS, which considers the constraints that are not directly considered when solving RMP. RMP is first solved, and the dual variable values associated to the optimal solutions are used to build the objective function for PS (the reduced cost criterion). PS is then solved to check whether or not it exists a new profitable column useful to improve the objective function. If a new column is available, it is added to RMP and a new iteration is carried out; otherwise, the algorithm finishes. When the method used to solve PS can guarantee that not additional interesting columns exists, the solution to RMP is guaranteed to be optimal (at least for the linear relaxation).

Nowadays, it is widely accepted that by combining CG with metaheuristic approaches, the former method might be seriously boosted. As it could be expected, this is partially a consequence of the efficiency achieved by metaheuristics while solving the difficult optimization problems corresponding to the PS. Nonetheless, as it is shown in this work, it is also a consequence of collateral effects associated to the way metaheuristics are implemented and embedded within CG that directly tackles the causes of slow convergence. In this work, the design of metaheuristics to be embedded within CG is analyzed , and several and well known approaches used to help accelerate CG are discussed to demonstrate how CG can benefit from the characteristics and flexibility offered by metaheuristic solution approaches. Several simple ideas that can be considered when designing a CG framework for solving computationally difficult problems are evaluated through extensive computational tests that show the interest of using such approaches.

2 Metaheuristics and Column Generation

Pure CG approaches might suffer from several pathological issues that affect its performance (see for example: [3, 14]). Consequently, the use of strategies to cope with such problems is not only desirable but might be necessary. Several causes have been identified. In first place, CG demands to solve a PS at every single iteration what might cause troubles when it corresponds to a difficult optimization problem. Additionally, CG may present some issues that are inherent to the approach. One of the most typical problems corresponds to convergence, meaning that while in the first iterations the evolution of the objective function is fast, in the latter iterations these improvements may be marginal and may be reflected in a large number of iterations required to fully solve the problem (*tail-off effect*) [7, 8]. A different problem corresponds to the *heading-in effect*, which can be explained as the successive enumeration of irrelevant columns, that are unlikely to be part of the optimal solution. This phenomenon often appears throughout the first iterations of CG, while not enough information is available to produce interesting columns, and can affect heavily its performance [15].

To cope with the aforementioned problems, several strategies have been successfully applied that mostly attack the problem from the viewpoint of the dual problem [2, 5, 11, 13]. Nonetheless, it is remarkable to see that CG obtain major benefits when it is combined with metaheuristic approaches. In the latter case, it

is typical to see that the basic structure of metaheuristics helps to improve CG performance compared to other exact approaches. Although, nowadays, MIP solvers are in a development state in which they are competitive against fast (meta-)heuristics in terms of the computational time required to solve difficult problems, the use of metaheuristics continues to be profitable as it seems they bring more reductions on the time required for solving problems when embedded in CG than its exact counterpart.

When using metaheuristics within a CG framework, three main approaches can be used to help accelerate the convergence towards the optimal solution:

- **Intensification strategies** An intensification strategy consists on returning to the RMP several interesting columns found through PS to the RMP at each iteration of CG [4, 9]. This strategy usually leads to a reduction in the number of iterations required for solving CG to achieve the optimal solution[12]. If a population based or trajectory based metaheuristic approach is used to solve PS, all the columns found during the optimization process might be saved to be added to the columns pool. The number of added columns might be limited to only κ columns such that the problem size is kept under control without losing the benefits offered by the intensification strategy.
- **Diversification strategies** Diversification strategies expand the idea of intensification. The purpose in this case is to compute at each iteration of CG an interesting set of columns contributing to different constraints [12]. This strategy might be powered if a metaheuristic approach used to solve PS is available that computes several columns simultaneously and offers naturally methods to increase diversity, *e.g* genetic algorithms, GRASP, etc.
- **Sequential application of metaheuristics and exact approaches** In this latter case, the idea is to exploit the easiness of the process of finding new profitable columns during the first iterations of CG. In this way, it is possible to solve the PS and compute columns that are easy to find with not-very-strong metaheuristics that can be sequentially replaced with more sophisticated metaheuristics or exact approaches when it becomes necessary while the CG solution process evolves [3]. Finally, once the solution process evolves enough, metaheuristics might be even replaced for exact approaches so as to confirm whether or not the current solution for RMP is optimal.

3 Results and discussion

Building an efficient column generation approach to solve a difficult optimization problem is a hard task. Furthermore, solving the pricing subproblem remains a complex task that often requires developing efficient specialized approaches to face it. The methods previously mentioned are tested for solving the Maximum Network Lifetime Problem in Wireless Sensor Networks with Coverage Constraints (α-MLP) [6]. For this problem, a pretty simple model based on an exponential number of variables is available. To solve the problem, an approach based on CG is proposed for which the three aforementioned approaches are

adopted through the use of diverse metaheuristic approaches, *e.g.* Evolutionary algorithms, GRASP, VNS. As shown in Figure 1 and 2 by applying the mentioned approaches, CG can be boosted compared to its basic implementation (CG-Exact). The experimental results show the benefits obtained for each of the presented methods and seem to indicate that, although not directly intended, the previously discussed strategies help to address the causes of slow convergence, *e.g.* the unstable behavior of dual variables values [11], diminishing the tail-off and heading-in effects.

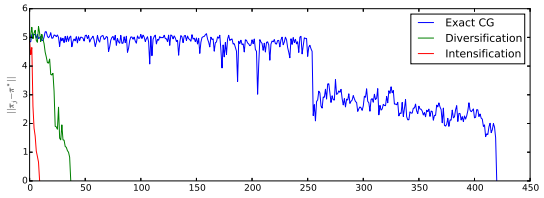


Fig. 1. Euclidean distance of dual variables to their optimal values along CG iterations

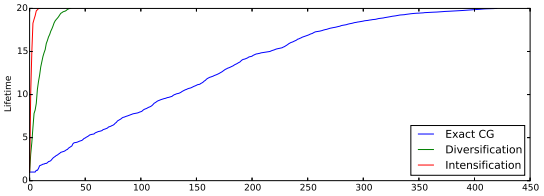


Fig. 2. Evolution of objective function along CG iterations

To help accelerate CG, it can be possible to combine CG with solution approaches that exploit several metaheuristics offering different performances both, in terms of the quality of the solution and in terms of the computational effort required for each. Furthermore, when these are used to return multiple and diverse columns, the number of iterations required to solve the problem might be heavily reduced while keeping CG being an exact method.

Bibliography

[1] D. Adelman. Dynamic bid prices in revenue management. *Operations Research*, 55(4):647–661, 2007.

[2] P. Benchimol, G. Desaulniers, and J. Desrosiers. Stabilized dynamic constraint aggregation for solving set partitioning problems. *European Journal of Operational Research*, 223(2):360–371, 2012.

[3] F. Castaño, A. Rossi, M. Sevaux, and N. Velasco. A column generation approach to extend lifetime in wireless sensor networks with coverage and connectivity constraints. *Computers & Operations Research*, 52:220–230, 2014.

[4] M. Desrochers, J. Desrosiers, and M.M. Solomon. A new optimization algorithm for the vehicle routing problem with time windows. *Operations research*, 40(2):342–354, 1992.

[5] O. Du Merle, D. Villeneuve, J. Desrosiers, and P. Hansen. Stabilized column generation. *Discrete Mathematics*, 194:229–237, 1999.

[6] Monica Gentili and Andrea Raiconi. α-coverage to extend network lifetime on wireless sensor networks. *Optimization Letters*, 7(1):157–172, 2013.

[7] P.C. Gilmore and R.E. Gomory. A linear programming approach to the cutting-stock problem. *Operations research*, 9(6):849–859, 1961.

[8] P.C. Gilmore and R.E. Gomory. A linear programming approach to the cutting stock problem-part ii. *Operations research*, 11(6):863–888, 1963.

[9] E. Macambira, N. Maculan, and C. de Souza. Integer programming models for the sonet ring assignment problem. Technical report, Technical Report IC-05-026, Universidade Federal do Rio de Janeiro, 2005.

[10] S.J. Maher. Solving the integrated airline recovery problem using column-and-row generation. *Transportation Science*, 2015.

[11] R.E. Marsten, W.W. Hogan, and J.W. Blankenship. The boxstep method for large-scale optimization. *Operations Research*, 23(3):389–405, 1975.

[12] N.T. Moungla, L. Létocart, and A. Nagih. Solutions diversification in a column generation algorithm. *Algorithmic Operations Research*, 5(2):86–95, 2010.

[13] P.J. Neame. Nonsmooth dual methods in integer programming phd thesis-department of mathematics and statistics. *The University of Melbourne March*, 1999.

[14] A. Rossi, A. Singh, and M. Sevaux. Column generation algorithm for sensor coverage scheduling under bandwidth constraints. *Networks*, 60(3):141–154, 2012.

[15] F. Vanderbeck. Implementing mixed integer column generation. In G. Desaulniers, J. Desrosiers, and M.M. Solomon, editors, *Column Generation*, pages 331–358. Springer US, 2005. 10.1007/0-387-25486-2_12.

Stochastic real world warehouse premarshalling

Vittorio Maniezzo[1], Marco Boschetti[2], and Walter Gutjahr[3]

[1] Department of Mathematics, University of Bologna, Italy
[2] Department of Computer Science, University of Bologna, Italy
[3] Department of Statistics and Operations Research, University of Vienna, Austria

1 Introduction

Warehouse premarshalling (or remarshalling) is the activity of reordering items in a storage location so that subsequent retrieval orders can be fulfilled with little or no need for further relocations. It has deep impact on operational costs, and it has been extensively studied with reference to containers in port yards or to boxes in warehouses. We are interested in this second application area, in the case when pickup orders become known only at the moment when they are to be performed.

We consider stacking organized warehouses, where homogeneous boxes are piled up one on each other in stacks. This is the most widespread warehouse storage strategy. Each level of the stack is called a tier, the storage location of a box is therefore given by its area (block), stack, and tier. An essential feature is that each stack can only be accessed following a LIFO policy. Operating a stack arranged warehouse involves several different activities, the most sensitive of which is order picking, reputedly the most labor-intensive and time-consuming process in warehouse logistics. Unfortunately, the LIFO strategy imposed by stacking, possibly implies the need to relocate unrequested boxes before getting access to the one targeted by the picking. Planning has the objective of storing the boxes in locations where they will be readily accessible for their future retrieval.

2 Premarshalling, deterministic setting

Customer orders are received daily as a picking list L of boxes to be retrieved. The precise knowledge of the picking list L is what differentiates the deterministic setting from the stochastic version.

A picking list L induces an ordering among boxes. Boxes to be retrieved first will have a higher *priority* than those to be retrieved later. After the premarshalling, the boxes with higher priority have to be on the top of the boxes with lower priority in their stack.

Boxes can be *moved* among the stacks. A move can either shift one item from the top of a stack to the top of another one (*relocation*) or pick an item from the top of a stack and bring it outside the warehouse (*removal*), if it is in a picking list. The problem asks to find a move sequence of minimum length, i.e. with the smallest number of moves, such that in the final layout no box is stored in a stack, having a lower priority box over it.

3 Premarshalling, stochastic setting

We do not assume the prior availability of the customer orders picking list L. A precise definition of the priorities is thus not possible. However, extensive historic data are usually available on the company ERP, including past picking lists, and these can be used to determine *expected priorities*.

Upon designing a suitable statistical model of the picking process, it is in fact possible to identify the best fitting distribution for random variables associated to each SKU in the ERP, which tells the expected number of boxes of each particular SKU that will be requested on the day after.

The problem can now be modeled as a two-stage stochastic programming problem, where *first stage decisions* are made on data known in advance, in our case the remarshalling acting on the known initial layout, a *random event*, in our case the arrival of a specific picking list L imposes a second decision and the *recourse action* copes with the actual case by handling its orders.

As in all two stage processes, the cost is given by the sum of the costs induced by the two successive decisions. However, in our case, the first-stage decision, whose cost function is associated with the number of relocation moves, dictates activities to underwork staff. The first-stage cost term is therefore 0, subject to a constraint that the total number of relocation moves must not exceed a threshold. This limit is determined by the length of the underwork interval.

The actual cost is incurred in the second stage, where the recourse function is defined as the expected value of the handling cost, i.e., the expected number of relocation moves needed to pick all the boxes in L.

We present a dynamic programming formulation of the problem under study, but it is impractical to directly use the DP formulation to solve the equivalent, as the DP algorithm quickly generates an exponential number of states even for small-sized instances. Therefore, to scale to real-world size, we need to resort to heuristics.

We adapted to our case two methods from the literature, namely the corridor method by Caserta and Voß [1] and the multiheuristic by Jovanovic et al. [2]. The idea behind the corridor approach is to reduce the size of the search space by iteratively allowing the exploration only of a part of the search space. At each iteration of the algorithm, one block to relocate is selected and the necessary moves to take it to a better position are scheduled. Differently from [1], we do not choose randomly the box to relocate, but we follow the priority indices in reversed order. After that, a subset of feasible relocation stacks is identified and one is chosen in probability.

We generated a set of artificial instances, in order to assess the scalability and the robustness of the approach. The instances were generated so to show descriptive statistics similar to those of the real world ones that originally motivated our study.

In order to determine the length of the lists, we assumed that the pickup requests in each day arrived following a Poisson distribution, since we clearly deal with a discrete distribution, where the requests occur independently, with the probability of getting a request in a time interval proportional to the length

of the interval, and where the rate at which requests arrive reasonably constant along the workshift. We therefore tried to fit Poisson distributions on our real world data in order to derive a relation that links the Poisson λ parameter with instance descriptors. One such loose relation was identified with respect to the number of boxes in the instance, which led to generation of instances with characteristics reported in Table 1.

Table 1. Artificial instances

name	stacks	tiers	nbox	nsku	λ
test10x5	10	5	40	6	3.2
test20x8	20	8	120	15	4.9
test30x8	30	10	250	30	7.0
test40x8	40	12	400	60	9.0
test50x8	50	15	700	100	11.0

Further validation of the proposed methodology was carried out on real world data. We were provided with extracts of WMS data relative to 1 year operation of 8 warehouses. The general descriptive statistics of these data were used as seeds for the generation of the instances in such a way that the biggest of the artificial instances was comparable with the smallest of the real world instances made available to us. The characteristics of these instances are listed in table 2. In this case the picking lists were the actual ones as derived from the WMS. We constructed two instance sets, one making use of the whole year's lists and one where we randomly extracted 50 lists for each warehouse, roughly corresponding to two months activities, which was the horizon of interest for the analysis.

Table 2. Real world instances

name	stacks	tiers	nbox	nsku	λ
instance1	71	18-30	1161	184	31
instance2	95	30	2330	362	64
instance3	92	18-24	1998	376	30
instance4	90	18-24	1852	371	28
instance5	88	24-30	1831	358	31
instance6	85	18-30	1735	304	30
instance7	69	12-18	999	113	15
instance8	51	15-24	737	102	11

Table 3 shows the comparative results obtained in the 2 stage setting by the proposed matheuristics versus manual operations. The results report the number of manual rehandlings needed for servicing identical, a priori unknown picking lists, in the case of untouched versus premarshalled warehouses. The number of

box movements decreases from 25% to 41%. Similar results are obtained in the one year horizon case.

Table 3. Real world instances, 50 scenarios

		Manual		MatHeu		
name	n.pallet	training	validation	training	validation	gap
instance1	1161	5786	5126	2962	2256	55.99%
instance2	2330	18811	15978	7344	12081	24.39%
instance3	1998	12656	13987	6076	10306	26.32%
instance4	1852	14751	12860	6548	8379	34.84%
instance5	1831	11143	11906	5484	9147	23.17%
instance6	1735	12497	15103	4375	8508	43.67%
instance4	999	6180	5328	2399	3182	40.28%
instance8	737	4123	3760	1432	2316	38.40%

References

1. Caserta, M., Voß, S.. A Corridor Method-Based Algorithm for the Pre-marshalling Problem. EvoWorkshops, volume 5484 of Lecture Notes in Computer Science Springer, (2009), page 788-797.
2. Jovanovic, R., Tuba, M., Voß, S., A multi-heuristic approach for solving the pre-marshalling problem, Central European Journal of Operations Research, 2015, pages 1–28.

Time-Bucket Relaxation Based Mixed Integer Programming Models for Scheduling Problems: A Promising Starting Point for Matheuristics*

Günther R. Raidl, Thomas Jatschka, Martin Riedler, and Johannes Maschler

Institute of Computer Graphics and Algorithms,
TU Wien, Vienna, Austria
{raidl|riedler|maschler}@ac.tuwien.ac.at, jatschka.thomas@gmail.com

1 Introduction

In job shop and project scheduling problems, generally speaking, a set of activities shall be scheduled over time. The execution of the activities typically depends on certain resources of limited availability and diverse other restrictions like precedence constraints. A feasible schedule is sought that minimizes some objective function like the makespan. For such problems, *mixed integer linear programming* (MIP) techniques are frequently considered, but also known to have severe limitations.

Basically, there are few general MIP modeling strategies for approaching such scheduling problems: Firstly, it is sometimes possible to come up with a compact model where the starting times of activities are directly expressed by means of corresponding variables. Resource constraints, however, impose a particular challenge in this respect. While they can be often treated in principle, e.g., by discrete-event models [2], these models are typically rather weak. A second, frequently applied option are so-called *time-indexed* (TI) formulations. They are based on a discretization of time, i.e., the activities may only start on a limited set of possible starting times. Binary variables are used that are additionally indexed by these possible starting times. The success of such TI models strongly depends on the resolution of the time discretization. While such models can have strong *linear programming* (LP) relaxations, the number of variables and constraints increases dramatically with the number of possible starting times. Frequently, a rather crude discretization can therefore only be applied to obtain any result in reasonable computation time. Further MIP techniques for approaching the considered scheduling problems make use of exponentially sized models and apply advanced techniques such as column generation, Lagrangian decomposition, or Benders decomposition, see, e.g., [2]. While they are frequently very successful, they are also substantially more complex to develop and implement.

Here, we consider a relaxation of a potentially very fine-grained TI model in which the set of possible starting times is partitioned into so-called *time-buckets*

* We thank EBG MedAustron GmbH, Wiener Neustadt, Austria, for the collaboration on particle therapy patient scheduling and partially funding this work.

(TB). This TB relaxation is typically much smaller than the original TI model and can be solved relatively quickly. An obtained solution provides a lower bound for the TI model's solution value but in general does not directly represent a feasible schedule as activity start times are only restricted to certain time-intervals. This solution, however, provides a promising starting point for matheuristics. On the one hand, we may try to derive a feasible schedule by heuristically fixing the start times to specific values, trying to fulfill all constraints. On the other hand, we can further subdivide some time-buckets and re-solve the resulting refined model to obtain an improved bound and a model that comes closer to the TI model. Doing this refinement iteratively yields a matheuristic that in principle converges to a provably optimal solution. In practice, it is crucial to subdivide the time-buckets in a sensible way in order to increase the model's size only slowly while hopefully obtaining significantly stronger bounds. (Meta-)heuristic techniques and dual variable information may provide a strong guidance.

The basic idea of the time-bucket relaxation originates in work from Wang and Regan [4] on the traveling salesman problem with time windows. Dash et al. [1] build upon this work and suggest an iterative refinement based on the solution to the LP-relaxation. We are not aware of any work that applies this principle already in the scheduling domain. There is just other work where the TI model is applied with different resolutions for the time discretization, but such approaches do in general not yield lower bounds and introduce imprecisions and are therefore conceptually different.

2 Resource Constrained Scheduling with Precedence Constraints

The above sketched general approach is more specifically investigated on a resource constrained scheduling problem with precedence constraints. This problem, for example, arises as a subproblem in the daily planning of activities to treat cancer patients with modern particle therapy [3]. Our experimental evaluation considers benchmark instances from this application.

We are given a set of resources $R = \{1, \dots, \rho\}$, a set of activities $A = \{1, \dots, \alpha\}$, and for each activity $a \in A$ a processing time p_a, a release time t^{r}_a, a deadline t^{d}_a, and a subset of required resources $Q_a \subseteq R$. Let the overall (huge) set of discrete times be $T = \{T^{\min}, \dots, T^{\max}\}$. Each resource $r \in R$ is only available at certain time intervals specified by set $W_r \subseteq T$. Last but not least, precedence constraints among the activities are stated by a directed acyclic graph $G = (A, P)$ with $P \subset A \times A$ and for each precedence relation expressed by an arc $(a, a') \in P$ minimum and maximum end-to-start time lags $L^{\min}_{a,a'}, L^{\max}_{a,a'} \in \mathbb{N}_{\geq 0}$ with $L^{\min}_{a,a'} \leq L^{\max}_{a,a'}$ need to be obeyed.

A solution $S = (S_1, \dots, S_\alpha) \in T^\alpha$ assigns to each activity $a \in A$ a starting time $S_a \in T$, from which on the activity is performed without preemption. We are looking for a feasible solution that minimizes the makespan.

3 Time-Bucket Relaxation and Matheuristic

Let $B = \{B_1, \ldots, B_\beta\}$ be a partitioning of T into subsequent time-buckets with $B_b = \{B_b^{\text{start}}, \ldots, B_b^{\text{end}}\}, \forall b = 1, \ldots, \beta$, and $B_b^{\text{end}} + 1 = B_{b+1}^{\text{start}},\ \forall b = 1, \ldots, \beta - 1$. We further make the following definitions.

- $I(B) = \{1, \ldots, \beta\}$ is the index set referring to all buckets in B.
- $W_r^B(b) = |B_b \cap W_r|$ denotes the aggregated availability of resource $r \in R$ over the whole bucket $b \in I(B)$.
- $C_a = \{C_{a,1}, \ldots, C_{a,\gamma_a}\} \subseteq 2^{I(B)}$ refers to all subsets of consecutive buckets in B to which an activity $a \in A$ can be jointly assigned so that some part of activity a is performed in each of the buckets. These sets can be determined by "sliding" the activity over all time-slots and taking the covered buckets.
- Let $t_{a,c}^{\text{smin}}$ be the earliest time-slots from T at which activity a can possibly start when it is assigned to bucket sequence $C_{a,c}$, and $t_{a,c}^{\text{smax}}$ the latest.
- For each bucket sequence $C_{a,c} \in C_a$ and each contained bucket $b \in C_{a,c}$ we further determine a lower bound $z_{a,b,c}^{\min}$ and an upper bound $z_{a,b,c}^{\max}$ for the number of time-slots at which activity a can possibly take place in bucket b when activity a is assigned to $C_{a,c}$.

The TB relaxation can now be stated as follows.

$$\min MS \tag{1}$$

$$\sum_{c=1}^{\gamma_a} y_{a,c} = 1 \qquad \forall a \in A \tag{2}$$

$$\sum_{c=1}^{\gamma_a} t_{a,c}^{\text{smin}} \cdot y_{a,c} + p_a \leq MS \qquad \forall a \in A \tag{3}$$

$$\sum_{a \in A, C_{a,c} \in C_a | b \in C_{a,c} \wedge r \in Q_a} z_{a,b,c}^{\min} \cdot y_{a,c} \leq W_r^B(b) \qquad \forall r \in R,\ b \in I(B) \tag{4}$$

$$\sum_{c'=1}^{\gamma_{a'}} t_{a',c'}^{\text{smax}} \cdot y_{a',c'} - \sum_{c=1}^{\gamma_a} t_{a,c}^{\text{smin}} \cdot y_{a,c} \geq p_a + L_{a,a'}^{\min} \qquad \forall (a, a') \in P \tag{5}$$

$$\sum_{c'=1}^{\gamma_{a'}} t_{a',c'}^{\text{smin}} \cdot y_{a',c'} - \sum_{c=1}^{\gamma_a} t_{a,c}^{\text{smax}} \cdot y_{a,c} \leq p_a + L_{a,a'}^{\max} \qquad \forall (a, a') \in P \tag{6}$$

$$y_{a,c} \in \{0, 1\} \qquad \forall a \in A, c = 1, \ldots, \gamma_a \tag{7}$$

$$MS \geq 0 \tag{8}$$

Variable MS represents the makespan to be minimized (1). Binary variables $y_{a,c}$ indicate if activity $a \in A$ is completely performed in bucket sequence $C_{a,c}$. Equations (2) ensure that for each activity exactly one bucket sequence is chosen from C_a. Inequalities (3) are used for determining the makespan MS. Inequalities (4) consider for each time bucket the aggregated resource availabilities and resource consumptions for performing the respective activities. Finally, inequalities (5) and (6) represent the precedence constraints with the minimum and maximum time lags, respectively.

Our matheuristic works as follows. We initially solve the TB relaxation for a rather crude partitioning of T into buckets. Then we try to derive a feasible schedule from the solution of the TB relaxation, i.e., we try to choose valid activity starting times as far as possible in correspondence to the selected bucket sequences. This is done by a greedy construction heuristic that considers the time buckets in chronological order and the assigned activities in a topological order, taking care of the precedence constraints and resource constraints as far as possible. Should we be able to find a feasible schedule whose makespan corresponds to the solution value of the TB relaxation, then this schedule is optimal and we can terminate.

Otherwise, the bucket partitioning is further refined by splitting buckets related to violated constraints. Furthermore, valid inequalities cutting off current infeasibilities may be added to the model. The refined model is solved again and the whole process iterated. We investigate and compare several strategies for the bucket splitting, considering also dual variable information from the relaxation.

4 Results and Conclusions

An experimental comparison with a compact discrete-event model and a classical TI formulation clearly shows the advantages of the TB relaxation: while the discrete-event model is only applicable to tiny instances due to its poor LP relaxation, the TI formulation suffers from its huge size when considering practically reasonable time discretizations. The matheuristic based on the iterative refinement of the TB model, however, soon yields reasonable lower bounds as well as feasible heuristic solutions, and both are improved over time.

The described approach is relatively generic and can rather easily be adapted to related scheduling problems. Clearly, there are many ways to enhance the basic concept: Intermediate heuristic solutions may be further improved by advanced local search techniques, the model may be strengthened by additional valid inequalities, possibly extending the approach to a branch-and-cut algorithm. More generally the field of hybrid metaheuristics and matheuristics provides plenty of opportunities to further exploit the proposed time-bucket relaxation.

References

1. Dash, S., Günlük, O., Lodi, A., Tramontani, A.: A time bucket formulation for the traveling salesman problem with time windows. INFORMS Journal on Computing 24(1), 132–147 (2012)
2. Hooker, J.N.: Planning and scheduling by logic-based Benders decomposition. Operations Research 55(3), 588–602 (2007)
3. Maschler, J., Riedler, M., Stock, M., Raidl, G.R.: Particle therapy patient scheduling: First heuristic approaches. In: Proceedings of the 11th Int. Conference on the Practice and Theory of Automated Timetabling (to appear 2016)
4. Wang, X., Regan, A.C.: On the convergence of a new time window discretization method for the traveling salesman problem with time window constraints. Computers & Industrial Engineering 56(1), 161–164 (2009)

A Dynamic Programming-Based Matheuristic for Dynamic Berth Allocation Problems

Tatsushi Nishi[1], Tatsuya Okura[1], Eduardo Lalla-Ruiz[2], and Stefan Voß[2]

[1] Graduate School of Engineering Science, Osaka University, Japan
[2] Institute of Information Systems, University of Hamburg, Germany

The increasing maritime traffic growth forces terminal operators to efficiently reduce the container ships' service time in order to maintain or increase their market share. This situation gave rise to the well-known berth allocation problem. Its goal is to determine the allocation and berthing time of each container ship arriving to a port with the aim of minimizing the total service time. For addressing this problem, we propose a dynamic programming based matheuristic approach that allows us to derive lower and upper bounds, and therefore, evaluate the optimality of the provided solutions. Its performance is assessed over realistic problem instances and compared to some of the best approaches proposed in the literature. In this regard, the computational results show that the performance exhibited by our proposed approach is competitive and significantly better than those approaches from the related literature.

A General Corridor Method Based Approach for Capacitated Facility Location

Marco Caserta[1] and Stefan Voß[2]

[1] IE University, Madrid, Spain
[2] Institute of Information Systems, University of Hamburg, Von-Melle-Park 5, 20146 Hamburg, Germany
[3] Escuela de Ingenieria Industrial, Pontificia Universidad Católica de Valparaíso, Chile

We present a corridor method based algorithm for the general Capacitated Facility Location Problem (CFLP). The algorithm exploits solutions obtained via Lagrangean relaxation and builds corridors around such solutions. A thorough exploration of the neighborhood induced by the corridor is carried out using an MIP solver. To show the genericity of the method we show that the algorithm can be used, with little or no modifications, on a number of variants of the CFLP. More precisely, we solve to (near) optimality over 500 benchmark instances, using the single-source as well as the multi-source formulations, both in the nominal, i.e., deterministic, and in the robust, i.e., stochastic, versions. The performance of the algorithm is competitive when compared with the best approaches proposed in the literature for each variant of the CFLP, especially considering that the algorithm has not been designed with a specific CFLP formulation in mind. In addition, we show that the algorithm outperforms the commercial MIP solver in terms of running time and is able to find near-optimal solutions very early in the search for all the CFLP variants analyzed.

MIP-Based Constructive Heuristics for the Three Dimensional Multiple Bin Size Bin Packing Problem with Air Transportation Constraints

Célia Paquay[1], Sabine Limbourg[1], José F. Oliveira[2], and Michael Schyns[1]

[1] University of Liege (ULg), HEC Management School, QuantOM
[2] INESC TEC, Faculdade de Engenharia, Universidade do Porto

The problem tackled here is the selection of containers in order to pack a set of cuboid boxes. The aim is to minimize the unused space inside the selected containers. The set of boxes is highly heterogeneous while there are few types of containers to select. According to the typology of cutting and packing problems, this is a three dimensional Multiple Bin Size Bin Packing Problem (MBSBPP) in which, in addition to the geometry constraints, some additional constraints encountered in practical packing situations are considered: the container weight limit, the orientation constraints, the load stability, the fragility of the boxes and the weight distribution within a container. Moreover, as the original problem is an air cargo application, we extend the definition of the MBSBPP to include situations in which the large objects may be truncated parallelepipeds. Indeed, in this context, containers are called Unit Load Devices (ULD) and may have specific shapes to fit inside aircraft.

This MBSBPP has been formulated as a mixed integer linear programming problem, but only poor results are achieved for even fairly small problem sizes. Therefore, the aim of the current research is to develop set of constructive matheuristics based on that formulation that may be able to provide better quality solutions to then be used as final solutions for the problem or as initial feasible solutions that may speed-up the mathematical programming model resolution.

Three methodologies based on the decomposition of the original problem into easier subproblems are considered: the Relax-and-Fix, the Insert-and-Fix and the Fractional Relax-and-Fix matheuristics. They were tested on real data sets to determine the best parameters for each technique and then compared to the results of the Branch-and-Bound. Finally, deeper experiments are carried out on the two best approaches to find their limitations.

Scaling analysis of high-performance TSP algorithms

Holger H. Hoos[1] and Thomas Stützle[2]

[1] University of British Columbia, Department of Computer Science, Vancouver, Canada
[2] Université libre de Bruxelles (ULB), IRIDIA, Brussels, Belgium

In this talk, we will give an overview of our work on the analysis of the scaling behavior of high-performing exact and heuristic algorithms for the traveling salesman problem. For the empirical scaling analysis we have used a novel approach, which is applicable to solvers for many other problems. Among the TSP solvers, we have considered concorde, the state-of-the-art exact TSP solver and two state-of-the-art heuristic algorithms, namely Helsgauns implementation of the Lin-Kernighan heuristic versions 1.3 and 2.0 and the evolutionary algorithms with EAX-crossover by Nagata and Kobayashi. We show that (i) for exact algorithms the empirical median run-time required to solve random uniform Euclidean (RUE) instances scales using a root-exponential scaling, (ii) the fraction of the time taken by exact algorithms to find the optimal solution is a large fraction of the overall run-time and growing with instance size, (iii) the heuristic algorithms scale better than exact algorithms when considering the finding times of optimal solutions in terms of constants and some also in terms of scaling law, and (iv) automatic configuration can have a significant impact on the scaling behavior of at least the heuristic algorithms.

The irace Package: Iterated Racing for Automatic Algorithm Configuration

Manuel López-Ibáñez[1], Leslie Pérez Cáceres[2], Jérémie Dubois-Lacoste[2], Mauro Birattari[2], and Thomas Stützle[2]

[1] University of Manchester, Alliance Manchester Business School, Manchester, UK
[2] Université libre de Bruxelles (ULB), IRIDIA, Brussels, Belgium

Modern optimization algorithms typically require the setting of a large number of parameters to optimize their performance. This is particularly also true for Matheuristics as such methods combine some elements of mathematical programming and heuristic search methods and, thus, result therefore often in algorithms with potentially even more parameters than algorithms that are based on a single paradigm. The immediate goal of automatic algorithm configuration is to find, automatically, the best parameter settings of such an optimizer. Ultimately, automatic algorithm configuration has the potential to lead to new design paradigms of optimization software.

The irace package is a software package that implements a number of automatic configuration procedures. In particular, irace offers iterated racing procedures, which have been used successfully to automatically configure various state-of-the-art algorithms. The iterated racing procedures implemented in irace include the iterated F-race algorithm and several extensions and improvements over the original iterated F-race proposal.

In our presentation, we will present shortly the rationale underlying the iterated racing procedures and introduce a number of recent extensions. Some of these extensions are described in the recent software description that is submitted for publication in Operations Research Perspectives (see following pages). These extensions include a restart mechanism to avoid premature convergence, the use of truncated sampling distributions to handle correctly parameter bounds, and an elitist racing procedure for ensuring that the best configurations found are also those evaluated in the higher number of training instances. We will discuss also additional changes that we are currently testing and that include the inclusion of surrogate models to predict the performance of configurations using random forest models, the inclusion of different sampling models and the extension of the irace package to handle better configuration objectives that involve the minimization of the computation time of target algorithms. Currently available preliminary results on the latter extensions indicate promising performance.

Author Index