Network Optimization INOC 2019

Benoit Darties, Michael Poss

Network Optimization
INOC 2019

9th International Network Optimization Conference
Avignon, France, June 12–14, 2019
Proceedings

Editors

Benoit Darties
Michael Poss

open
proceedings

Network Optimization (INOC 2019)
Proceedings of the 9th International Network Optimization Conference
Avignon, France, June 12–14, 2019

Series ISSN: 2510-7437

Editors

Benoit Darties, LIRMM, University of Montpellier, CNRS, Montpellier, France
Michael Poss, LIRMM, University of Montpellier, CNRS, Montpellier, France

OpenProceedings.org
University of Konstanz
University Library
78457 Konstanz, Germany

OpenProceedings ISBN: 978-3-89318-079-0

DOI of this front matter: `10.5441/002/inoc.2019.01`

Foreword

This volume corresponds to the Special Issue dedicated to the International Network Optimization Conference (INOC 2019), held in Avignon, France, from June 12 to June 14, 2019. This volume contains 20 papers (6 pages long) that were subject to a review process, each of them corresponding to a presentation at the conference.

INOC conferences are organized biannually by the European Network Optimization Group (ENOG), a working group of EURO. The aim of this conference is to provide researchers from different areas of Operations Research, with the opportunity to present and discuss their results and research on the field of Network Optimization, in an inspiring and bridge building environment where fruitful ideas may flow freely. INOC 2019 is the 9th edition of this event and the second to take place in France. Previous editions were held in Lisbon (2017), Warsaw (2015), Tenerife (2013), Hamburg (2011), Pisa (2009), Spa (2007), Lisbon (2005) and Paris (2003).

INOC 2019 was organized in Avignon, at the University of Avignon, in collaboration with the Laboratoire d'Informatique d'Avignon and the Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier.

In INOC 2019 there were two types of submissions: full papers (4-6 pages long) and extended abstracts (1-2 pages long). We received a total of 28 full papers and 39 extended abstracts. After a peer-review process 21 full papers and 38 extended abstracts were accepted. The papers included in this volume correspond to 20 of the accepted full papers. In total, we had 58 contributed presentations. In addition, there were 3 invited plenary sessions

- "Scalable On-Demand Mobility Services" by Pascal Van Hentenryck (Georgia Tech, USA)
- "Hub Location Problems: Applications, Models and Solution Methods" by Hande Yaman (KU Leuven, Belgium)
- "Analyzing Network Robustness via Interdiction Problems" by Rico Zenklusen (ETH Zurich, Switzerland)

and two tutorials

- "Modern Branch-and-Cut-and-Price for Vehicle Routing Problems" by Ruslan Sadykov (Inria Bordeaux Sud-Ouest, France)
- "Linearization techniques for MINLP: recent developments, challenges and limits" by Sandra Ulrich Ngueveu (Toulouse INP, France)

Benoit Darties, Michael Poss

Program Committee Members

Edoardo Amaldi, Politecnico di Milano (Italy)
Zacharie Alès, UMA / CEDRIC ENSTA ParisTech (France)
Walid Ben-Ameur, Telecom SudParis (France)
Andreas Bley, Universität Kassel (Germany)
Christina Büsing, RWTH Aachen University (Germany)
Fabio D'Andreagiovanni, ZIB (Germany)
Bernard Fortz, Université Libre de Bruxelles (Belgium)
Bernard Gendron, University of Montreal (Canada)
Eric Gourdin, Orange Labs (France)
Luís Gouveia, Universidade de Lisboa (Portugal)
Arie Koster, RWTH Aachen University (Germany)
Markus Leitner, Vrije Universiteit Amsterdam (Netherlands)
Ivana Ljubic, ESSEC (France)
Abilio Lucena, Universidade Federal do Rio de Janeiro (Brazil)
Dritan Nace, Université de Technologie de Compiegne (France)
Adam Ouorou, Orange Labs (France)
Pierre Pesneau, Université de Bordeaux (France)
Michal Pioro, Warsaw University of Technology (Poland) and Lund University (Sweden)
Michael Poss, LIRMM, Université de Montpellier (France), *Chair*
S. Raghavan, University of Maryland (USA)
Cristina Requejo, Univesidade de Aveiro (Portugal)
Juan José Salazar-Gonzalez, Universidad de La Laguna (Spain)
Maria Grazia Scutellà, Università di Pisa (Italy)
Douglas Shier, Clemson University (USA)
Amaro de Sousa, Universidade de Aveiro (Portugal)
Eduardo Uchoa, Universidade Federal Fluminense (Brazil)
Stefan Voss, Universität Hamburg (Germany)
Hande Yaman, Bilkent University (Turkey)

Table of Contents

Research Papers

Interdependent Infrastructure Network Restoration Optimization from Community and Spatial Resilience Perspectives

Deniz Berfin Karakoc
School of Industrial and Systems Engineering
University of Oklahoma
Norman, Oklahoma, USA
denizberfinkarakoc@ou.edu

Kash Barker
School of Industrial and Systems Engineering
University of Oklahoma
Norman, Oklahoma, USA
kashbarker@ou.edu

Yasser Almoghathawi
Systems Engineering Department
King Fahd University of Petroleum and Minerals
Dahran, Saudi Arabia
moghathawi@kfupm.edu.sa

ABSTRACT

Modern societies rely on the critical infrastructure networks to ensure their operability and existence. Most of the recent research and government planning revolves around maintaining the proper and continuous functioning of these critical infrastructure networks. However, these critical infrastructure networks do not exist on their own, but they perform interdependently. Thus, the study of forming resilient interdependent infrastructures against natural or man-made large-scale disruptions and planning the restoration of these critical networks becomes a more complex challenge. As such, the frequency of large-scale disruptions appears to be increasing and devastating for the surrounding communities in the long-term, the social and geographic aspects of these disruptions should be emphasized in the restoration planning studies so that resilience and well-being of the served community is also optimized. In this work, we integrate (i) a resilience-driven multi-objective mixed-integer programming formulation that schedules the restoration of disrupted components in each network with (ii) a geographically distributed social vulnerability index and population density ratio and (iii) a spatial risk measure to assign the impact of the surrounding environment to the system. This model is illustrated with an example study in Shelby County, TN in the United States.

1 INTRODUCTION

Critical infrastructure networks, such as power, natural gas, and water distribution, form the backbone of modern societies to provide their daily needs and ensure their safety, high socio-economic standards, and quality of life. However, these critical infrastructures have experienced various disruptions in the past and continue to be subject to both external and internal stressors such as aging-induced system failures, natural disasters, and malevolent attacks. Hence, given the inevitability of these large-scale disruptions, an ability to adapt and quickly recover from these disruptions is extremely crucial for both the interdependent infrastructure networks and their surrounding communities.

Moreover, these networks have become more dependent on each other where they contain a bi-directional relationship to operate properly and more efficiently [30]. This type of a complex coordination that is caused by physical, spatial, cyber, or logical interdependencies can increase performance efficiency and reduce the resource consumption of these networks since the output of one network could be the input of another. However, due to the existence of such complex coordinations, there is a possibility of chain reactions of dysfunctionality between the interdependent components due to disruption in a single network. Hence, this type of bi-directional relationships could enhance the overall network vulnerability since complete system failure could be caused by a disruption in a single network. Therefore, the study of recovery planning to ensure a desired level of resilience in these highly vulnerable networks become a crucial challenge [7, 22, 36]. Hence, the importance of addressing risks associated with the interdependencies among the critical infrastructures through building secure and resilient networks is highlighted in many governmental planning documents [28] and examined in recent literature work [1, 4, 18].

Further, the socio-economic status and the demographics of the served community, as well as the spatial risks related to the surrounding environment and the location of these networks, could increase the impact of disruptions [24] over the system performance, thus system resilience. Therefore, resilience and recovery planning studies for the interdependent infrastructure networks should take social and spatial vulnerabilities into account to reveal more reliable and comprehensive guidance to decision makers.

In this work, we study the problem of interdependent infrastructure network restoration after the occurrence of a disruptive event with a focus on the vulnerability of society that interacts with the network and additional hazard risk of the surrounding environment. As for the results, we observe that when additional community and spatial resilience measures are included in the problem, both the optimal restoration schedule of disrupted components and the performance of networks through the restoration process show changes. Therefore, the overall system resilience through time differs when community and spatial vulnerability measures are taken into account.

2 BACKGROUND

2.1 Network Resilience

The term resilience is defined as the ability to withstand, adapt to, and recover from a disruption [27]. Even though the definition is commonly agreed, many different approaches are introduced in the literature to formulate and quantify the resilience of a network. Some of the proposed measurement methods include (i) describing the resilience as the normalized area underneath the performance graph [11], (ii) representing the resilience as a function of topological measures [32], and (iii) quantifying resilience as the probability of recovery [21].

As shown in Figure 1, two primary dimensions of resilience, vulnerability and recoverability, help characterize network resilience [5]. The vulnerability of a network states the magnitude of damage in the performance of a network due to a stressor [19], where the recoverability of a network refers to the speed at which the network reaches to its desired performance level [31]. Hence, resilience is measured in this work as the of network recovery over network loss through complete recovery period [17].

ISBN 978-3-89318-079-0 on OpenProceedings, ISSN 2510-7437

DOI: 10.5441/002/inoc.2019.02

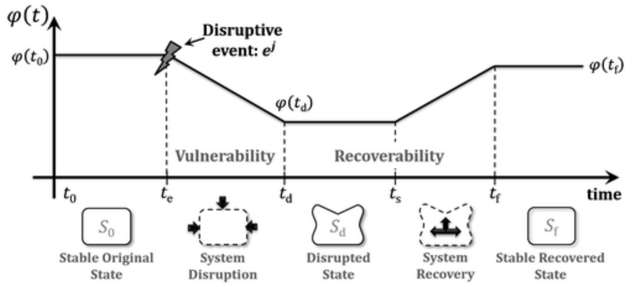


Figure 1: The Network Performance $\phi(t)$ across the Before, During and After States of a Disruptive Event [17].

2.2 Recovery of Infrastructure Networks

The study of optimally scheduling the restoration of infrastructure networks is emphasized in the last decade especially due to the high frequency of both natural disasters and the malevolent attacks. In the literature, a mixed-integer programming is developed with an objective of minimizing the total cost associated with the flow and unmet demand in the network system [20]. Another optimization model is for the restoration of disrupted components in the interdependent networks is formulated in such a way that each component is assigned with a recovery due date that should be satisfied [15]. A different mixed-integer programming model is proposed that (i) determines the set of disrupted components that should be restored and (ii) assigns the related work crews through the restoration that would ensure the minimum total cost of flow, unmet demand and restoration activities [33]. More recently, an interdependent infrastructure network design problem is introduced in the literature that schedules the restoration activities of the disrupted components under certain budgetary and resource-based constraints [16]. Finally, a different approach is defined as a two-phase recovery for physically interdependent critical infrastructures that includes both a linear and a mixed-integer programming with the objectives of minimizing the flow cost and maximizing the total amount of commodity deliveries in the system [35].

In this study, we extend a previously proposed approach for the restoration scheduling of interdependent infrastructure networks [2] in such a way that the modified resilience-driven multi-objective mixed integer programming model would account for the additional risk and vulnerability measures of the surrounding environment. Hence, the newly optimal restoration schedule of disrupted components would prioritize the community and spatial resilience perspectives.

2.3 Social Vulnerability

Social vulnerability is defined by the set of characteristics of a group or individual that influence their capacity to anticipate, cope with, resist, and recover from the impact of a hazard [6]. Many studies propose to identify the behavioral aspects, human occupancy, and response level of societies that are shaped by these different socio-economic characteristics.

The social vulnerability of a community is often defined by the number and the availability of recovery resources. Such resources for a disrupted region include the number of work crews, restoration equipment, number of physicians, their dispatch locations [26], shelter number and capacity [34], and medical capacity [3]. However, some of the proposed studies revolve around the idea that certain social and economic characteristics, namely inequalities and differences in the society, have an effect on vulnerability and recoverability. Some of the most commonly considered demographics are racial and technical inequalities [26], and educational inequalities [25, 26] where according to the way that these socio-economic characteristics are defined, they either contribute to or counteract the resilience of the communities against disruptive events.

A common algorithm to quantify social vulnerability is the Social Vulnerability Index (SoVI), which is a measure that is formulated by the different levels of age, gender, race, wealth, and occupation of the citizens [12]. This proposed algorithm considers multiple socio-economic characteristics to define vulnerability levels based on the cumulative effect of all the demographics. These socio-economic characteristics are utilized to identify the 42 variables that are grouped into 11 factors to be used in the SoVI algorithm [12]. These 11 factor groups that are listed in Table 1, are used to measure the social vulnerability index of communities to accurately estimate their recoverability, resilience capacity and response level against a possible stressor.

Table 1: Social Vulnerability Index Factors

Personal wealth	Ethnicity (Native-American)
Age	Occupation
Ethnicity (Hispanic)	Infrastructure dependence
Race (African-American)	Housing stock and tenancy
Race (Asian)	Density of the built environment
Single-sector economic dependence	

In this study, we utilize a reduced version of the SoVI algorithm, the SoVI-Lite approach [14]. The SoVI-Lite approach contains less technical implementation but efficient data compilation [14]. An overview of the SoVI-Lite implementation is as follows:

1. Calculate the ratio of the population that is included in the all possible 42 socio-economic variables
2. Standardize the percentages of variables to the z-scores
3. Assign signs to the z-scores according to the influence of a higher percentage level of the variables on social vulnerability concept
4. Sum all the z-scores

We also normalize the final sum of z-scores for each geographic region such that a SoVI of 0 suggests the least socially vulnerable and of 1 suggests the most socially vulnerable community.

2.4 Spatial Risk

In addition to social demographics, the surrounding environment and changes in spatial conditions also affect the impacts of a disruption as experienced by a community [14]. These spatial conditions refer both the type of the local region (e.g., village, sub-district) [37] and the geographic location (e.g., island, coastal area, volcanic risk area, seismic hazard zone). [9].

In this study, we consider the geographic location as the spatial risk indicator where risk is caused by the high possibility of being subjected to a specific natural disaster, an earthquake. To quantify earthquake risk, we use the peak ground acceleration (PGA) measure to formally express the expected seismic hazard impact due to a ground shake [13].

Additionally, we scale the PGA measures of different geographic regions to be between 0 and 1. For the PGA measures, similar to the SoVI scores, 0 represents spatially the least risky location and 1 represents spatially the most risky location.

3 PROPOSED MODEL

We extend a multi-objective resilience-driven restoration optimization model for restoration scheduling of interdependent infrastructure networks in such a way that differing levels of community and spatial resilience measures are taken into account while planning the recovery process optimally after a disruption. We integrate social vulnerability, population density, and spatial risk measures into the resilience maximization and total restoration cost minimization objectives to ensure that the restoration scheduling is driven by community and spatial resilience perspectives.

3.1 Model Assumptions

In the proposed multi-objective resilience-driven mixed-integer programming model, the following assumptions hold: (i) each network consists of nodes and links that are either not disrupted or fully disrupted, (ii) the recovery duration can vary for each component in each network, (iii) disrupted components are not operational unless their restoration is completed, (iv) the demand and supply of the nodes and the flow of the links are known in advance, (v) a known unmet demand penalty, restoration, and flow cost is associated with component in each network, (vi) for a component to be functional all the physically interdependent components must be functional as well,(vii) a fixed number of work crews are assigned to each network for restoration, and finally (viii) a specific disrupted component could be restored by a single work crew at a certain time period.

3.2 Model Notation

The sets, parameters, and the decision variables of the proposed optimization model for interdependent infrastructure network restoration problem are listed in Table 2, 3 and 4, respectively.

Table 2: Sets of Restoration Model

Notation	Explanation
T	Available recovery time horizon, $T = \{1, \ldots, \tau\}$
K	Interdependent infrastructure networks, $K = \{1, \ldots, \kappa\}$
N	Nodes of the networks
L	Links of the networks
N'	Disrupted nodes
L'	Disrupted links
N^k	Nodes in network $k \in K$
L^k	Links in network $k \in K$
R^k	Restoration work crews for network $k \in K$
N_s^k	Supply nodes in network $k \in K$, $N_s^k \subseteq N^k$
N_d^k	Demand nodes in network $k \in K$, $N_d^k \subseteq N^k$
N'^k	Disrupted nodes in network $k \in K$, $N'^k \subseteq N^k$
L'^k	Disrupted links in network $k \in K$, $L'^k \subseteq L^k$
Ψ	Interdependent nodes

3.3 Objectives of Model

The total amount of unmet demand in the network states the system loss that is caused by the disruptive event. Thus, decreasing the amount of total unmet demand to a desired level refers to enhancing system performance and represents the effectiveness of the restoration process. Hence, the resilience of the system would be formalized by the cumulative recovery of the interdependent infrastructure networks over the total system loss through a certain time horizon as in Eq. 1.

Table 3: Parameters of Restoration Model

Notation	Explanation
b_i^k	Amount of maximum flow at node $i \in N^k$
$SoVI_i^k$	Social vulnerability index for demand node $i \in N_d^k$
V_i^k	Social vulnerability score for demand node $i \in N_d^k$
P_i^k	Population density for demand node $i \in N_d^k$
PGA_i^k	Peak ground acceleration measure for demand node $i \in N_d^k$
G_i^k	Peak ground acceleration score for demand node $i \in N_d^k$
Q_i^k	Unmet demand of node $i \in N_d^k$ after disruption
μ^k	Weight of each network $k \in K$
fn_i^k	Restoration cost for disrupted node $i \in N'^k$
fl_{ij}^k	Restoration cost for disrupted link $(i, j) \in L'^k$
c_{ij}^k	Unitary flow cost for link $(i, j) \in L^k$
p_i^k	Unmet demand penalty cost for demand node $i \in N_d^k$
dn_i^k	Restoration duration of the disrupted node $i \in N'^k$
dl_{ij}^k	Restoration duration of the disrupted link $(i, j) \in L'^k$
u_{ij}^k	Flow capacity of link $(i, j) \in L^k$

Table 4: Decision Variables of Restoration Model

Notation	Explanation
s_{it}^k	Amount of unmet demand at node $i \in N_d^k$ at time $t \in T$
x_{ijt}^k	Flow through link $(i, j) \in L^k$ at time $t \in T$
y_i^k	Restoration status of node $i \in N'^k$
z_{ij}^k	Restoration status of link $(i, j) \in L'^k$
α_{ijt}^k	Operational status of link $(i, j) \in L'^k$ at time $t \in T$
β_{it}^k	Operational status of node $i \in N^k$
γ_{it}^{kr}	Work crew assignment to node $i \in N'^k$ for restoration
δ_{ij}^{kr}	Work crew assignment to link $(i, j) \in L'^k$ for restoration

$$\max \sum_{k \in K} \mu^k \sum_{t=1}^{\tau} \left(\frac{t\left[\sum_{i \in N_d^k} \left(Q_i^k V_i^k P_i^k G_i^k \right) - \sum_{i \in N_d^k} \left(s_{it}^k V_i^k P_i^k G_i^k \right) \right]}{\sum_{i \in N_d^k} \left(\tau Q_i^k V_i^k P_i^k G_i^k \right)} \right.$$
$$\left. \frac{-(t-1)\left[\sum_{i \in N_d^k} \left(Q_i^k V_i^k P_i^k G_i^k \right) - \sum_{i \in N_d^k} \left(s_{i(t-1)}^k V_i^k P_i^k G_i^k \right) \right]}{\sum_{i \in N_d^k} \left(\tau Q_i^k V_i^k P_i^k G_i^k \right)} \right) \tag{1}$$

The second objective of the proposed model takes the total cost associated with the (i) restoration process that includes the recovery of the disrupted nodes and links, (ii) the flow cost, and (iii) the penalty cost of the leftover unmet demand in the system which fluctuates by both the social and geographical vulnerability levels of the service areas of the demand nodes. Therefore, the minimization of the total cost objective would be formulated as in Eq. 2.

$$\min \sum_{k \in K} \left(\sum_{i \in N'^k} fn_i^k y_i^k + \sum_{(i,j) \in L'^k} fl_{ij}^k z_{ij}^k \right.$$
$$\left. + \sum_{t \in T} \left[\sum_{(i,j) \in L^k} c_{ij}^k x_{ijt}^k + \sum_{i \in N_d^k} p_i^k V_i^k P_i^k G_i^k s_{it}^k \right] \right) \tag{2}$$

3.4 Mathematical Model

The explained objectives are subject to the following constraints.

$$\sum_{(i,j)\in L^k} x_{ijt}^k - \sum_{(j,i)\in L^k} x_{jit}^k = 0, \qquad \forall i \in N^k \setminus \{N_s^k, N_d^k\}, k \in K, t \in T \tag{3}$$

$$\sum_{(j,i)\in L^k} x_{jit}^k + s_{it}^k = b_i^k, \qquad \forall i \in N_d^k, k \in K, t \in T \tag{4}$$

$$x_{ijt}^k - u_{ij}^k \beta_{it}^k \leq 0, \qquad \forall (i,j) \in L^k, i \in N^k, k \in K, t \in T \tag{5}$$

$$x_{ijt}^k - u_{ij}^k \beta_{jt}^k \leq 0, \qquad \forall (i,j) \in L^k, j \in N^k, k \in K, t \in T \tag{6}$$

$$x_{ijt}^k - u_{ij}^k \alpha_{ijt}^k \leq 0, \qquad \forall (i,j) \in L^k, k \in K, t \in T \tag{7}$$

$$\beta_{\bar{i}t}^{\bar{k}} \leq \beta_{it}^k, \qquad \forall \Big((i,k),(\bar{i},\bar{k})\Big) \in \Psi \tag{8}$$

$$z_{ij}^k = \sum_{r\in R^k} \sum_{t\in T} \delta_{ijt}^{kr}, \qquad \forall (i,j) \in L'^k, k \in K \tag{9}$$

$$y_i^k = \sum_{r\in R^k} \sum_{t\in T} \gamma_{it}^{kr}, \qquad \forall i \in N'^k, k \in K \tag{10}$$

$$\sum_{(i,j)\in L'^k} \sum_{l=t}^{min(\tau, t+dl_{ij}^k - 1)} \delta_{ijl}^{kr} + \sum_{i\in N'^k} \sum_{l=t}^{min(\tau, t+dn_i^k - 1)} \gamma_{il}^{kr} \leq 1 \qquad \forall k \in K, r \in R^k, t \in T \tag{11}$$

$$\beta_{it}^k \leq \sum_{r\in R^k} \sum_{l=1}^{t} \gamma_{il}^{kr}, \qquad \forall i \in N'^k, k \in K, t \in T \tag{12}$$

$$\alpha_{ijt}^k \leq \sum_{r\in R^k} \sum_{l=1}^{t} \delta_{ijl}^{kr}, \qquad \forall (i,j) \in L'^k, k \in K, t \in T \tag{13}$$

$$\sum_{t=1}^{dl_{ij}^k - 1} \alpha_{ijt}^k = 0, \qquad \forall (i,j) \in L'^k, k \in K \tag{14}$$

$$\sum_{t=1}^{dn_i^k - 1} \beta_{it}^k = 0, \qquad \forall i \in N'^k, k \in K \tag{15}$$

$$\sum_{r\in R^k} \sum_{t=1}^{dl_{ij}^k - 1} \delta_{ijt}^{kr} = 0, \qquad \forall (i,j) \in L'^k, k \in K \tag{16}$$

$$\sum_{r\in R^k} \sum_{t=1}^{dn_i^k - 1} \gamma_{it}^{kr} = 0, \qquad \forall i \in N'^k, k \in K \tag{17}$$

$$s_{it}^k \geq 0, \qquad \forall i \in N_d^k, k \in K, t \in T \tag{18}$$

$$x_{ijt}^k \geq 0, \qquad \forall (i,j) \in L^k, k \in K, t \in T \tag{19}$$

$$y_i^k \in \{0,1\}, \qquad \forall i \in N'^k, k \in K \tag{20}$$

$$z_{ij}^k \in \{0,1\}, \qquad \forall (i,j) \in L'^k, k \in K \tag{21}$$

$$\beta_{it}^k \in \{0,1\}, \qquad \forall i \in N^k, k \in K, t \in T \tag{22}$$

$$\alpha_{ijt}^k \in \{0,1\}, \qquad \forall (i,j) \in L'^k, k \in K, t \in T \tag{23}$$

$$\gamma_{it}^{kr} \in \{0,1\}, \qquad \forall i \in N'^k, k \in K, t \in T, r \in R^k \tag{24}$$

$$\delta_{ijt}^{kr} \in \{0,1\}, \qquad \forall (i,j) \in L'^k, k \in K, t \in T, r \in R^k \tag{25}$$

In the proposed mathematical model, the first two constraints, Eqs. (3) and (4), govern the flow conservation of node $i \in N^k$. In Eqs. (5) to (7), capacities of the network components are formulated. Eq. (8) governs the physical interdependency between nodes. In Eqs. (9) to (18), the restoration process of disrupted components is formulated, where Eqs. (9) and (10) ensure the work crew assignment for to be restored components, Eq. (11) ensures that a single work crew can restore at most one disrupted component in network $k \in K$ at a specific time $t \in T$, Eqs. (12) and (13) ensure the operability of a component when its restoration is completed, and Eqs. (14) to (18) ensure that for a disrupted component to be functional, its restoration should be completed. Finally, the nature of decision variables in the optimization model is represented in Eqs. (18) to (25).

4 ILLUSTRATIVE EXAMPLE

The proposed model is applied to data collected for Shelby County, Tennessee in the United States, whose geographic location is the epicenter of the New Madrid Seismic Zone [16].

The three distinct critical interdependent infrastructure networks, water, gas and power distribution systems, are represented in Figure 2. There is a total of 124 nodes, 37 of which are demand nodes, and a total of 176 links in the three networks. We implement a single scenario with 19 disrupted demand nodes and assign two work crews separately for each network to complete the restoration process simultaneously for all three networks in 28 time periods.

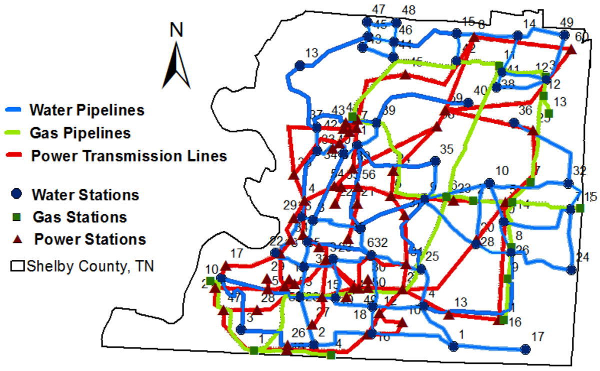


Figure 2: Layout of Interdependent Water, Gas and Power Infrastructure Networks over Shelby County [16].

In Figure 3, the geographic location of the demand nodes of all three infrastructure networks, i.e. water, gas and power, and the PGA measures that are specific to each region due to the New Madrid Seismic Zone is illustrated [13].

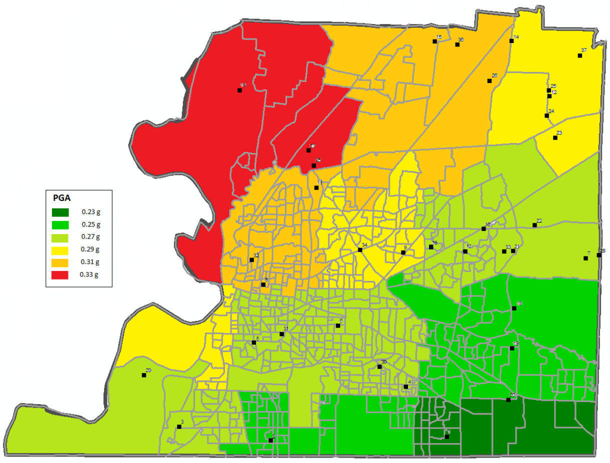


Figure 3: Distribution of Regional PGA Measures among Shelby County [13].

For the SoVI-Lite approach, the eight variables from Table 5 are used in the block group level for Shelby County, TN, where block groups are formed by multiple adjacent blocks with a total of 300 to 6000 residents [8].

To assign the social vulnerability scores and the proportional population densities that are calculated in block group level to each demand node, the block groups are distributed among them according to their location to represent the specific service area of demand nodes. For this distribution process, Voronoi diagram method is utilized [29]. The Voronoi diagram method calculates the distance from predetermined input points to any point in the sample space. Later, it sets the boundaries for the coverage area of input points in such a way that any point in the sample space is covered by its closest input point.

Table 5: SoVI-Lite Variables for Block Groups

Percentage of households that earn less than \$75.000 annually
Percentage of population that is African-American
Percentage of population that is Asian
Percentage of population that is Hispanic
Percentage of population that is over age 65
Percentage of population that is under age 5
Percentage of single-female based households
Percentage of households that live under the poverty line

Also, the social vulnerability indices $SoVI_i^k$ are normalized and relatively more importance is given to the demand nodes that are highly vulnerable by implementing an exponential effect to formulate the social vulnerability scores V_i^k, [23]. Also, a similar approach is utilized to enhance the emphasize on demand nodes with higher peak ground acceleration measure, PGA_i^k. The exponential formulation of the social vulnerability scores and the peak ground acceleration scores are represented in Eq. 26 and in Eq. 27, respectively.

$$V_i^k = e^{a*SoVI_i^k}, \quad \forall i \in N_d^k, a \in Z^+ \tag{26}$$

$$G_i^k = e^{a*PGA_i^k}, \quad \forall i \in N_d^k, a \in Z^+ \tag{27}$$

To account for the social expectations and the human occupancy levels of the service areas of demand nodes, we include the population density measure in the proposed approach as we adopted the idea that the size of the population that is represented by each demand node could also be an effective aspect in the community-resilience perspective. The population density measure is formulated as the ratio of the population that is served by demand node $i \in N_d^k$, over the total population of all service areas.

After the distribution of the block groups to demand nodes, an average of the social vulnerability scores is taken and the population density of block groups proportional to their layout in the Voronoi cells is calculated to assign these measures to the demand nodes. The visualization of the social vulnerability scores among the block groups is illustrated in Figure 4.

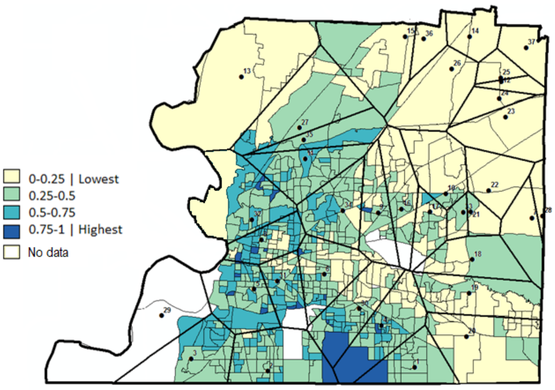


Figure 4: Distribution of Block-Group Social Vulnerability Scores and Demand Node Service Areas among Shelby County [12, 14].

The ϵ-constraint method is used in the resilient objective to transform it into a constraint with the assigned values such that $\epsilon \in [0,1]$, to solve the multi-objective problem [10]. As the resilience levels are $\in [0,1]$, the consistent ϵ-constraint formulation is in Equation 28.

$$\sum_{k \in K} \mu^k \sum_{t=1}^{\tau} \left(\frac{t\left[\Sigma_{i\in N_d^k} \left(Q_i^k V_i^k P_i^k G_i^k\right) - \Sigma_{i\in N_d^k} \left(s_{it}^k V_i^k P_i^k G_i^k\right)\right]}{\Sigma_{i\in N_d^k} \left(\tau Q_i^k V_i^k P_i^k G_i^k\right)} \right.$$
$$\left. \frac{-(t-1)\left[\Sigma_{i\in N_d^k} \left(Q_i^k V_i^k P_i^k G_i^k\right) - \Sigma_{i\in N_d^k} \left(s_{i(t-1)}^k V_i^k P_i^k G_i^k\right)\right]}{\Sigma_{i\in N_d^k} \left(\tau Q_i^k V_i^k P_i^k G_i^k\right)} \right) \leq \epsilon \tag{28}$$

The following Table 6 represents a subset of disrupted nodes in each network and the change in the restoration schedule. The second column, titled as 'With' states recovery scheduling results when social vulnerability and spatial risk measures are taken into consideration, i.e. the defined parameters of V_i^k, P_i^k and G_i^k are included in the model whereas the third column labeled as 'Without' states the restoration order without these measures. The disrupted node which is scheduled earliest in the restoration process is ranked 1 where the disrupted node that is ordered latest in the restoration process is ranked 4. Lastly, Figure 5 represents the change in the network performance for three infrastructure networks when community and spatial resilience measures are considered and not considered in the optimization model.

Table 6: A Subset of Restoration Schedule Comparison for Critical Infrastructure Networks

Water Node ID	With	Without	Gas Node ID	With	Without	Power Node ID	With	Without
45	1	2	1	1	2	34	1	4
10	2	1	15	2	4	14	2	2
48	3	4	13	3	3	5	3	1
27	4	3	9	4	1	20	4	3

Note the difference in the ranking of disrupted nodes when the additional social and environmental measures are included in restoration problem of interdependent infrastructure networks. Not only the recovery order of demand nodes is changed, that is assigned with social vulnerability and spatial risk scores as their importance measure, but also ranking of the transshipment nodes and supply nodes that are effective in the delivery of the commodity and responsible from providing the needs of these relatively more important demand nodes differ.

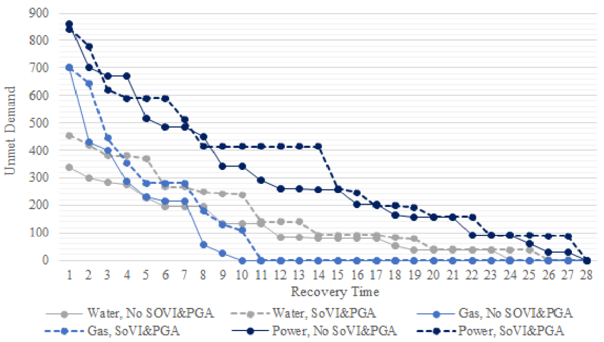


Figure 5: Illustration of the Change in the Network Performance

In this study, we encounter when additional social vulnerability and spatial risk measures are considered, optimum restoration schedule differs for each network. As the restoration schedule differs, the total unmet demand in each network hence the network performance through time differ when the results of both models are compared.

5 CONCLUSION

Modern day societies heavily depend on the continuous and proper functioning of critical infrastructure networks in terms of maintaining their existence and day-to-day operability. These physical infrastructures contain an interdependency such as they would be attached to each other logically, physically, geographically or informatively. Additionally, there exists a bi-directional relationship between the community networks and physical infrastructure networks for supply and demand manners. Therefore, the critical infrastructure networks become more vulnerable against external stressors where any disruption that would occur in these networks would impact the societies and the resilience and vulnerability levels of the societies would effect the performances of these networks.

In this study, to achieve more comprehensive understanding of the interdependent infrastructure network resilience we proposed a resilience-driven multi-objective mixed-integer programming model that is integrated with the vulnerability levels of its surrounding environment. To plan accordingly with the social expectations against disruptions and the geographical risks associated with the spatial location of these networks, the proposed approach takes into account a geographically distributed (i) social vulnerability index to represent the behavioral responses of the various socio-economic dynamics in the society, and (ii) geographic risk index measure to illustrate the differing potential disruption levels of a spatial hazard.

As for the results of our proposed study, we observe that considering the social vulnerability, population density measures of the surrounding community and the potential geographic risk of the spatial location of these networks requires a different restoration scheduling to recover from external stressors in a timely manner. The newly achieved restoration schedule of the disrupted components, and the performance of critical networks through time are both planned based on the resilience enhancement of both surrounding community and the physical networks. For future work, we believe that as more aspects of vulnerability is considered additionally in the proposed study, more extended research with higher humanitarian motivation would be accomplished.

REFERENCES

[1] Dilek Tuzun Aksu and Linet Ozdamar. 2014. A mathematical model for post-disaster road restoration: Enabling accessibility and evacuation. *Transportation research part E: logistics and transportation review* 61 (2014), 56–67.

[2] Yasser Almoghathawi, Kash Barker, and Laura A. Albert. 2019. Resilience-driven restoration model for interdependent infrastructure networks. *Reliability Engineering & System Safety* 185 (2019), 12 – 23. https://doi.org/10.1016/j.ress.2018.12.006

[3] Erik Auf der Heide and Joseph Scanlon. 2007. Health and medical preparedness and response. *Emergency management: Principles and practice for local government* (2007), 183–206.

[4] Prabin M Baidya and Wei Sun. 2017. Effective restoration strategies of interdependent power system and communication network. *The Journal of Engineering* 2017, 13 (2017), 1760–1764.

[5] Kash Barker, Jose Emmanuel Ramirez-Marquez, and Claudio M Rocco. 2013. Resilience-based network component importance measures. *Reliability Engineering & System Safety* 117 (2013), 89–97.

[6] Piers Blaikie, Terry Cannon, Ian Davis, and Ben Wisner. 1994. *At risk: natural hazards, people's vulnerability and disasters.* Routledge.

[7] Sergey V Buldyrev, Roni Parshani, Gerald Paul, H Eugene Stanley, and Shlomo Havlin. 2010. Catastrophic cascade of failures in interdependent networks. *Nature* 464, 7291 (2010), 1025.

[8] US Census Bureau. 2010. Geographic Terms and Concepts Report. (2010).

[9] Jayajit Chakraborty, Graham A Tobin, and Burrell E Montz. 2005. Population evacuation: assessing spatial variability in geophysical risk and social vulnerability to natural hazards. *Natural Hazards Review* 6, 1 (2005), 23–33.

[10] Vira Chankong and Yacov Y Haimes. 2008. *Multiobjective decision making: theory and methodology.* Courier Dover Publications.

[11] Gian Paolo Cimellaro, Andrei M Reinhorn, and Michel Bruneau. 2010. Seismic resilience of a hospital system. *Structure and Infrastructure Engineering* 6, 1-2 (2010), 127–144.

[12] Susan L Cutter. 2003. The vulnerability of science and the science of vulnerability. *Annals of the Association of American Geographers* 93, 1 (2003), 1–12.

[13] Leonardo Dueñas-Osorio, James I Craig, and Barry J Goodno. 2007. Seismic response of critical interdependent networks. *Earthquake engineering & structural dynamics* 36, 2 (2007), 285–306.

[14] Christopher T Emrich and Susan L Cutter. 2011. Social vulnerability to climate-sensitive hazards in the southern United States. *Weather, Climate, and Society* 3, 3 (2011), 193–208.

[15] Jing Gong, Earl E Lee, John E Mitchell, and William A Wallace. 2009. Logic-based multiobjective optimization for restoration planning. In *Optimization and Logistics Challenges in the Enterprise.* Springer, 305–324.

[16] Andrés D González, Leonardo Dueñas-Osorio, Mauricio Sánchez-Silva, and Andrés L Medaglia. 2016. The interdependent network design problem for optimal infrastructure system restoration. *Computer-Aided Civil and Infrastructure Engineering* 31, 5 (2016), 334–350.

[17] Devanandham Henry and Jose Emmanuel Ramirez-Marquez. 2012. Generic metrics and quantitative approaches for system resilience as a function of time. *Reliability Engineering & System Safety* 99 (2012), 114–122.

[18] Richard Holden, Dimitri V Val, Roland Burkhard, and Sarah Nodwell. 2013. A network flow model for interdependent infrastructures at the local scale. *Safety Science* 53 (2013), 51–60.

[19] Henrik Jönsson, Jonas Johansson, and Henrik Johansson. 2008. Identifying critical components in technical infrastructure networks. *Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability* 222, 2 (2008), 235–243.

[20] Earl E Lee II, John E Mitchell, and William A Wallace. 2007. Restoration of services in interdependent infrastructure systems: A network flows approach. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 37, 6 (2007), 1303–1317.

[21] Yi Li and Barbara J Lence. 2007. Estimating resilience for water resources systems. *Water Resources Research* 43, 7 (2007).

[22] Richard G. Little. 2002. Controlling Cascading Failure: Understanding the Vulnerabilities of Interconnected Infrastructures. *Journal of Urban Technology* 9, 1 (2002), 109–123. https://doi.org/10.1080/106307302317379855 arXiv:https://doi.org/10.1080/106307302317379855

[23] R Timothy Marler and Jasbir S Arora. 2010. The weighted sum method for multi-objective optimization: new insights. *Structural and multidisciplinary optimization* 41, 6 (2010), 853–862.

[24] Dennis Mileti. 1999. *Disasters by design: A reassessment of natural hazards in the United States.* Joseph Henry Press.

[25] Betty Hearn Morrow. 1999. Identifying and mapping community vulnerability. *Disasters* 23, 1 (1999), 1–18.

[26] Fran H Norris, Susan P Stevens, Betty Pfefferbaum, Karen F Wyche, and Rose L Pfefferbaum. 2008. Community resilience as a metaphor, theory, set of capacities, and strategy for disaster readiness. *American journal of community psychology* 41, 1-2 (2008), 127–150.

[27] Barack Obama. 2013. Presidential policy directive 21: Critical infrastructure security and resilience. *Washington, DC* (2013).

[28] Department of Homeland Security. 2013. National Preparedness Report. (2013).

[29] Atsuyuki Okabe, Toshiaki Satoh, Takehiro Furuta, Atsuo Suzuki, and Kyoko Okano. 2008. Generalized network Voronoi diagrams: Concepts, computational methods, and applications. *International Journal of Geographical Information Science* 22, 9 (2008), 965–994.

[30] S. M. Rinaldi, J. P. Peerenboom, and T. K. Kelly. 2001. Identifying, understanding, and analyzing critical infrastructure interdependencies. *IEEE Control Systems Magazine* 21, 6 (Dec 2001), 11–25. https://doi.org/10.1109/37.969131

[31] Adam Rose. 2007. Economic resilience to natural and man-made disasters: Multidisciplinary origins and contextual dimensions. *Environmental Hazards* 7, 4 (2007), 383–398.

[32] Daniel J Rosenkrantz, Sanjay Goel, SS Ravi, and Jagdish Gangolly. 2009. Resilience metrics for service-oriented networks: A service allocation approach. *IEEE Transactions on Services Computing* 2, 3 (2009), 183–196.

[33] Thomas C Sharkey, Burak Cavdaroglu, Huy Nguyen, Jonathan Holman, John E Mitchell, and William A Wallace. 2015. Interdependent network restoration: On the value of information-sharing. *European Journal of Operational Research* 244, 1 (2015), 309–321.

[34] Kathleen Tierney. 2009. *Disaster response: Research findings and their implications for resilience measures.* Technical Report. CARRI Research Report Oak Ridge, TN.

[35] Diman Zad Tootaghaj, Novella Bartolini, Hana Khamfroush, and Thomas La Porta. 2017. Controlling cascading failures in interdependent networks under incomplete knowledge. In *Reliable Distributed Systems (SRDS), 2017 IEEE 36th Symposium on.* IEEE, 54–63.

[36] Baichao Wu, Aiping Tang, and Jie Wu. 2016. Modeling cascading failures in interdependent infrastructures under terrorist attacks. *Reliability Engineering & System Safety* 147 (2016), 1–8.

[37] J Zeng, ZY Zhu, JL Zhang, TP Ouyang, SF Qiu, Y Zou, and T Zeng. 2012. Social vulnerability assessment of natural hazards on county-scale using high spatial resolution satellite imagery: a case study in the Luogang district of Guangzhou, South China. *Environmental Earth Sciences* 65, 1 (2012), 173–182.

On the Complexity of RSSA of Anycast Demands in Spectrally-Spatially Flexible Optical Networks

Róża Goścień and Piotr Lechowicz
Department of Systems and Computer Networks, Faculty of Electronics
Wroclaw University of Science and Technology, Wroclaw, Poland
{roza.goscien,piotr.lechowicz}@pwr.edu.pl

ABSTRACT

Spectrally-spatially flexible optical networks (SS-FONs) are proposed as a solution to overcome the expected capacity crunch caused by the rapidly growing overall Internet traffic. SS-FONs combine two network technologies, namely, flex-grid optical networks and spatial division multiplexing yielding a significant capacity increase. Moreover, network services applying anycast transmission are gaining popularity. In anycasting, the same content is provided in several geographically spread data centers (DCs), and the requested content is delivered to the network client from the most convenient DC, e.g., minimizing the network traffic and delay. The main optimization challenge in SS-FONs is routing, spectrum and space allocation (RSSA) problem, which can be solved using integer linear programming (ILP). The main goal of this paper is to compare the complexity of various ILP models for routing static anycast traffic in SS-FONs over single-mode fiber bundles (SMFBs). The proposed ILP models apply different modeling techniques, i.e., slice-based and lightpath-based. Moreover, proposed models differ with the core switching (lane changes) capability and consideration of DCs location problem. In order to test the complexity and scalability of models, we run simulations assuming a different number of demands, fibers in SMFBs, candidate paths and DCs.

1 INTRODUCTION

The rapid increase of overall Internet traffic results with the intensive effort concentrated on preventing the future capacity crunch. Spectrally-spatially flexible optical networks (SS-FONs) are proposed as a possible solution to overcome the limitations of currently deployed wavelength division multiplexing (WDM) backbone optical networks. SS-FONs combine two network technologies, namely, flexgrid optical networks and spatial division multiplexing (SDM) providing a massive capacity increase. Due to the additional spatial dimension, SS-FONs enable parallel transmission of several co-propagating optical channels in properly designed optical fibers. As the main advantages, SS-FONs allow for multi-carrier (super-channel, SCh) transmission, adaptive modulation formats selection, better spectral utilization with additional flexibility in the spatial resources management and potential cost savings due to the integrated devices. One of the possible fiber technologies supporting SS-FONs is single-mode fibers bundles (SMFBs), i.e., several single-core single-mode fibers aggregated in a single bundle [4]. The key network devices, such as reconfigurable optical add/drop multiplexers (ROADMs), are yet expected, thus, different assumptions are taken about their switching capabilities. In particular, if core switching (lane changes) is considered, the spatial switching between input and output ports is allowed in network nodes, i.e., the lightpaths may have assigned differently indexed fibers in the SMFB links belonging to the routing path. Otherwise, each lightpath has assigned fibers with the same index (resulting in lower devices complexity). The fundamental optimization challenge in SS-FONs is routing, spectrum and space allocation (RSSA). It aims to select a routing path, optical channel and spatial resources for each of the traffic requests [3]. The most common approach for solving static optimization problems is the integer linear programming (ILP). Two common ILP techniques are applied in the modeling of optical networks, namely, slice-based and lightpath-based. In the former, the starting and ending slice of each request is considered as a decision variable, while in the latter, a set of precomputed lightpaths is used, where each lightpath defines valid optical channel [10].

Concurrently, many network services, such as content delivery networks (CDNs) or cloud computing, apply anycast transmission in order to minimize the traffic load and delay in the network. According to Cisco forecast, the CDNs will cover almost 71% of total traffic in 2021 [1] making optimization of anycast traffic an important issue. In more detail, anycast is the one-to-one-of-many transmission technique, where one of the request end nodes is fixed, i.e., a client node, and the second node is selected from the set of possible nodes where the data centers (DCs) are located. DCs are spread geographically and each one provides the same content, hence, the request may be handled with any of the available DCs (e.g., the closest one) [14]. Besides improvement of the network performance, the anycast traffic makes related optimization problems more complex and challenging.

In this paper, we focus on the modeling and complexity analysis of RSSA of anycast demands. We consider scenarios with and without core switching possibility. Moreover, we study cases with given DC locations and cases that incorporate DC placement into the optimization task. Overall, we focus on four RSSA versions and each version we define using two ILP models (slice-based and lightpath-based) proposing eight ILP models. Then, we compare them in terms of complexity (measured in number of included variables and constraints), processing time and scalability.

2 RELATED WORKS

Recently, the topic of SS-FONs has been extensively studied (e.g., see [3, 5, 13]). Different ILP models have been used to define RSSA problem variations using link-path slice-based [3, 7, 8, 16], link-path lightpath-based [3, 9, 11], or node-link [3, 6] formulations. In particular, in [8], the authors propose ILP model for routing, spectrum and core allocation (RSCA) problem for SS-FONs over multi-core fibers (MCFs) minimizing the highest allocated slice index, with worst-case inter-core crosstalk (XT) awareness assuming realistic physical fiber parameters. In turn, in [6], ILP model is proposed for routing, wavelength and core allocation problem, maximizing the number of accepted traffic while minimizing the total number of used network resources in MCF-based SDM networks applying multi-input multi-output XT suppression. In [7], RSCA ILP model is given that minimizes the overall

ISBN 978-3-89318-079-0 on OpenProceedings, ISSN 2510-7437

DOI: 10.5441/002/inoc.2019.03

network cost due to the number of switching modules required for different cores assuming programmable ROADMs. In [11], ILP model of extended RSSA problem is presented, namely, routing, modulation format, baud-rate and spectrum allocation which jointly minimizes the cost of installed transceivers and spectrum occupation utilizing few-mode transmission. In [15], different lightpath-based ILP models are proposed for the RSSA considering various spatial allocation flexibility. In [9], lane changes are allowed and the lightpath-based model makes use of the relaxation of the space continuity constraint resulting with the lower number of constraints and decision variables. ILP model assuming anycast traffic in SS-FONs is only considered in [16].

To the best of our knowledge, there has been no work that compares the complexity of various ILP models formulations for the allocation of anycast demands in SS-FONs. To fill the literature gap, in this paper, we define eight ILP models assuming two modeling techniques, i.e., link-path slice-based and link-path lightpath-based. The models differ with the core switching capability and additional DCs placement problem.

3 NETWORK MODEL

The network is modeled as a directed graph $G = (V, E)$ where V is the set of network nodes and E is the set of directed optical links that connect the nodes. Each link $e \in E$ comprises a number of single-mode fibers that are aggregated in the bundle and are included in set K. Spectrum resources of each fiber $k \in K$ are divided into narrow frequency slices (slots) of 12.5 GHz width and are included in set S. Signals are transmitted using optical corridor within spectral super-channel (SCh) by grouping several adjacent slices. Each SCh has to be separated from the adjacent one using fixed-size guardband of 12.5 Ghz width. We assume that there are R DCs available in the network and each one provides the same content. Depending on the problem, either DCs are located at some nodes or the DCs location problem is considered.

A set of anycast traffic requests (D) to be served in the network is given. Each demand is defined with the client node and bit-rate. To simplify the problem, we assume only downstream (from DCs to client nodes) transmission. Let P_d denote the set of candidate routing paths for each demand $d \in D$. Each routing path $p \in P_d$ originates in one of the DCs and terminates in the client node. The number of required slices for given demand depends on the requested bit-rate and length of the applied routing path and is denoted with n_{dp}. Each demand is realized using one routing path within selected spectral SCh in such a way that each slice of each fiber of each link is used at most by one demand. Finally, if in the considered scenario core switching ability is not assumed, each SCh has to be realized using the same indexed fibers. Otherwise, it can be freely switched between fibers among the routing path.

The aim of the optimization task is to allocate all demands and minimize index of the highest allocated slice [3]. We study four problem versions which differ with the core switching possibility (with core switching — WCS, without core switching — WOCS) and DC location scenario. For DC location scenario, we analyze two cases: (*i*) the location of DCs is given in advance, (*ii*) the problem of DCs placement is a part of the optimization task. Below, we present ILP models of the four problem versions using slice-based (Sec. IV) and lightpath-based (Sec. V) approaches. All presented models are based on link-path modelling technique. Details about considered models, number of decision variables and constraints are presented in Table 1.

4 SLICE-BASED (SB) MODELS

In this section, link-path slice-based models are proposed. According to Table 1, the models complexity depends mostly on the number of demands to be realized.

sets and indices

$v \in V$	network nodes
$e \in E$	network physical links
$s \in S$	frequency slices
$k \in K$	link fibers
$d \in D$	traffic anycast (DC to client direction) demands
$p \in P_d$	candidates paths for demand d

constants

R	number of DCs (to be) placed in the network
$\lvert K \rvert$	number of fibers available on each physical link
δ_{edp}	= 1, if link e belongs to routing path p associated with demand d; 0, otherwise
n_{dp}	number of slices required to realize demand d on candidate path $p \in P_d$
$o(d, p)$	origin node of path $p \in P_d$ defined for demand $d \in D$

variables

x_{dp}	=1, if demand d is realized using candidate path $p \in P_d$; 0, otherwise (binary)
y_{dk}	=1, if demand d uses fiber k; 0, otherwise (binary)
y_{dke}	=1, if demand d uses fiber k on physical link e; 0, otherwise (binary)
w_d/z_d	index of starting/ending slice for demand d (integer ≥1)
z	index of the highest allocated slice in the network (integer ≥ 1)
a_{di}	= 1, if ending slot of demand d is greater or equal to starting slice of demand i (binary)
c_{di}	=1, if demands d and i have some common fibers and links; 0, otherwise (binary)
r_v	=1, if DC is placed in node $v \in V$; 0, otherwise (binary)

4.1 Without core-switching + given DCs (SB_WOCS)

Objective (1) and constraints (2)–(8).

objective

$$\min z. \tag{1}$$

subject to

$$\sum_{p \in P_d} x_{dp} = 1, d \in D \tag{2}$$

$$\sum_{k \in K} y_{dk} = 1, d \in D \tag{3}$$

$$z_d - w_d + 1 = \sum_{p \in P_d} x_{dp} n_{dp}, d \in D. \tag{4}$$

$$z \geq z_d, d \in D \tag{5}$$

$$|S| a_{di} \geq z_d - w_i + 1, d, i \in D : d \neq i \tag{6}$$

$$c_{di} \geq \sum_{p \in P_d} x_{dp} \delta_{edp} + y_{dk} + \sum_{p \in P_i} x_{ip} \delta_{eip} + y_{ik} - 3,$$

$$e \in E, k \in K, d, i \in D, d \neq i \tag{7}$$

$$a_{di} + a_{id} + c_{di} \leq 2, d, i \in D : d < i \tag{8}$$

The objective 1 is to minimize the overall spectrum usage defined as the width of spectrum (i.e., number of slices) required to allocate the demands. Constraint 2 ensures that exactly one routing path is selected for each demand. The selection of only one fiber for request is controlled with the equation 3. Constraint 4 defines starting and ending slice index of each demand. The value of objective is controlled by the inequality 5. Inequality 6 denotes the relation between starting slice of one demand with ending

Table 1: Modeling technique, core switching capability and DC placement location selection for considered ILP models

	ILP model	core switching	DC placement	Number of variables	Number of constraints
slice based	SB_WOCS	no	no	$\lvert D\rvert\,(\lvert P\rvert + \lvert K\rvert + 2 + 2\lvert D\rvert) + 1$	$\lvert D\rvert(4 + \frac{3}{2}\lvert D\rvert + \lvert E\rvert\lvert K\rvert\lvert D\rvert)$
	SB_WCS	yes	no	$\lvert D\rvert\,(\lvert P\rvert + \lvert K\rvert\lvert E\rvert + 2 + 2\lvert D\rvert) + 1$	$\lvert D\rvert(3 + \lvert E\rvert + \frac{3}{2}\lvert D\rvert + \lvert E\rvert\lvert K\rvert\lvert D\rvert)$
	SB_WOCS_DC	no	yes	$\lvert D\rvert\,(\lvert P\rvert + \lvert K\rvert + 2 + 2\lvert D\rvert) + \lvert V\rvert + 1$	$\lvert D\rvert(4 + \frac{3}{2}\lvert D\rvert + \lvert E\rvert\lvert K\rvert\lvert D\rvert + \lvert P\rvert) + 1$
	SB_WCS_DC	yes	yes	$\lvert D\rvert\,(\lvert P\rvert + \lvert K\rvert\lvert E\rvert + 2 + 2\lvert D\rvert) + \lvert V\rvert + 1$	$\lvert D\rvert(3 + \lvert E\rvert + \frac{3}{2}\lvert D\rvert + \lvert E\rvert\lvert K\rvert\lvert D\rvert + \lvert P\rvert) + 1$
lightpath based	LB_WOCS	no	no	$\lvert S\rvert + \lvert D\rvert\lvert L\rvert + \lvert E\rvert\lvert K\rvert\lvert S\rvert + \lvert E\rvert\lvert S\rvert$	$\lvert D\rvert + \lvert E\rvert\lvert K\rvert\lvert S\rvert + \lvert E\rvert\lvert S\rvert + \lvert S\rvert$
	LB_WCS	yes	no	$\lvert S\rvert + \lvert D\rvert\lvert L\rvert + \lvert E\rvert\lvert S\rvert$	$\lvert D\rvert + \lvert E\rvert\lvert S\rvert + \lvert S\rvert$
	LB_WOCS_DC	no	yes	$\lvert S\rvert + \lvert D\rvert\lvert L\rvert + \lvert E\rvert\lvert K\rvert\lvert S\rvert + \lvert E\rvert\lvert S\rvert + \lvert V\rvert$	$\lvert D\rvert + \lvert E\rvert\lvert K\rvert\lvert S\rvert + \lvert E\rvert\lvert S\rvert + \lvert S\rvert + 1 + \lvert D\rvert\lvert L\rvert$
	LB_WCS_DC	yes	yes	$\lvert S\rvert + \lvert D\rvert\lvert L\rvert + \lvert E\rvert\lvert S\rvert + V$	$\lvert D\rvert + \lvert E\rvert\lvert S\rvert + \lvert S\rvert + 1 + \lvert D\rvert\lvert L\rvert$

slice of another one while 7 switches c_{di} on if demands d and i have some common fibers and links. Finally, the spectrum not overlapping is controlled with the inequality 8, which is a contradiction if two demands utilize slices within the same frequency region and contain common fiber.

4.2 With core-switching + given DCs (SB_WCS)

Objective (1) and constraints (2), (9), (4)–(6), (10), (8).
subject to

$$\sum_{k \in K} y_{dke} = \sum_{p \in P_d} x_{dp}\delta_{edp}, d \in D, e \in E \tag{9}$$

$$c_{di} \geq \sum_{p \in P_d} x_{dp}\delta_{edp} + y_{dke} + \sum_{p \in P_i} x_{ip}\delta_{eip} + y_{ike} - 3,$$
$$e \in E, k \in K, d, i \in D, d \neq i \tag{10}$$

Equation 9 assures selection of exactly one fiber on each link of the selected routing path for each demand. Additionally, inequality 10 instead of 7 does account for the core switching ability.

4.3 Without core-switching + DC placement (SB_WOCS_DC)

Objective (1) and constraints (2)–(8), (11), (12).

$$\sum_{v \in V} r_v = R \tag{11}$$

$$x_{dp} \leq r_{o(d,p)}, d \in D, p \in P_d \tag{12}$$

Equality 11 ensures that exactly R nodes are chosen to host DCs, while the inequality 12 controls that each selected routing path originates in the node selected for DC location.

4.4 With core-switching + DC placement (SB_WCS_DC)

Objective (1) and constraints (2), (9), (4)–(6), (10), (8), (11), (12).

5 LIGHTPATH-BASED (LB) MODELS

In this section, link-path lightpath-based ILP models are proposed. Let L_d denote the set of precomputed available lightpaths for demand $d \in D$, where each lightpath $l \in L_d$ is associated with one routing path. Moreover, each lighpath defines utilized starting and ending slice index of the SCh and, in the case of WOCS models, fiber on each link among the routing path. Note, the size of the lightpath corresponds to the n_{dp} constant. According to Table 1, the models complexity depends mostly on the number of candidate lightpath for each demand, which in turn is determined by the number of candidate paths and available spectrum width.

sets and indices (additional)
$l \in L_d$ candidates lightpaths for demand d
constants (additional)
β_{dls} = 1, if lightpath l for demand d uses slice s; 0, otherwise
ξ_{dle} = 1, if link e belongs to lightpath l for demand d; 0, otherwise
γ_{dlk} = 1, if lightpath l for d uses core k; 0, otherwise
$o(d, l)$ origin node of lightpath $l \in L_d$ for demand d
variables (binary)
u_{dl} =1, if demand d uses lightpath $l \in L_d$; 0, otherwise
v_{eks} =1, if slice s is used on fiber k on link e; 0, otherwise
v_{es} =1, if slice s is used on any fiber of link e; 0, otherwise
v_s =1, if slice s is used on any fiber on any link; 0, otherwise

5.1 Without core-swithing + given DCs (LB_WOCS)

Objective (13) and constraints (14)–(17).
objective

$$\min \sum_{s \in S} v_s \tag{13}$$

subject to

$$\sum_{l \in L_d} u_{dl} = 1, d \in D \tag{14}$$

$$\sum_{d \in D} \sum_{l \in L_d} u_{dl}\xi_{dle}\gamma_{dlk}\beta_{dls} \leq v_{eks}, e \in E, s \in S, k \in K \tag{15}$$

$$\sum_{k \in K} v_{eks} \leq |K| v_{es}, e \in E, s \in S \tag{16}$$

$$\sum_{e \in E} v_{es} \leq |E| v_s, s \in S \tag{17}$$

The objective 13 is to minimize the overall spectrum usage. Equality 14 assures selection of exactly one lightpath for each demand. In 15, v_{esk} is switched on if slice s is used on fiber k on physical link e. Inequalities 16 and 17 control whether slice s is used on any fiber of link e and on any link, respectively.

5.2 With core-switching + given DCs (LB_WCS)

Objective (13) and constraints (14), (18), (17).

$$\sum_{d \in D} \sum_{l \in L_d} u_{dl}\xi_{dle}\beta_{dls} \leq |K| v_{es}, e \in E, s \in S. \tag{18}$$

Constraint 18 assures that the slice s on link e is used at most $|K|$ times. This constraint makes use of the relaxation of the space continuity constraint.

Table 2: Supported bit-rate and transmission reach for modulation formats.

Modulation format	BPSK	QPSK	8-QAM	16-QAM
Supported bit-rate [Gbps]	200	150	100	50
Transmission reach [km]	6300	3500	1200	600

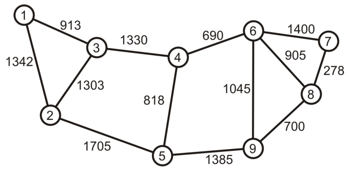


Figure 1: Int9 network topology.

5.3 Without core-swithing + DC placement (LB_WOCS_DC)

Objective (13) and constraints (14)–(17), (11), (19).

$$u_{dl} \leq r_{o(d,l)}, d \in D, l \in L_d. \tag{19}$$

Inequality 19 controls whether each selected lightpath associated with the demand d originates in a DC node.

5.4 With core-swithing + DC placement (LB_WCS_DC)

Objective (13) and constraints (14), (18), (17), (11), (19).

6 NUMERICAL EXPERIMENTS

In this section, we evaluate performance of various ILP models for the RSSA problem in SS-FONs, considering anycast demands. We run experiments on Int9 network topology (Fig. 1) composed of 9 nodes and 13 links, with an average link length of 1063 km. We assume SS-FON composed of SMFBs, where each fiber provides a 4 THz bandwidth (spectrum) divided into 320 frequency slices, each of 12.5 GHz width. As in [12], we assume that the transceiver operates at the fixed baud rate and each transceiver transmits/receives optical channel (optical carrier, OC) of 37.5 GHz width (3 slices). We consider 4 available modulation formats, namely, BPSK, QPSK, 8-QAM and 16-QAM. Supported bit-rate and transmission reach depend on the selected modulation format and are presented in Table 2 [2]. If the requested bit-rate exceeds a single transceiver capacity for the applied MF, the request is transmitted using several OCs within one SCh using a set of adjacent slices (i.e., a spectral SCh). Each request is transmitted using the most efficient MF supporting the required transmission reach, i.e., a distance adaptive transmission is used. In most of the experiments, we assume that the number of fibers per SMFB is $|K| = 2$, the number of candidate shortest paths is $|P| = 2$ (number of candidate paths between client node and one of the DCs) and number of available DCs is $R = 2$. The number of demands is equal to $|D| = \{10, 20, 30, 40, 50, 60, 80, 100\}$. For each value, we evaluate 5 randomly generated sets of demands. Each demand has randomly selected source and destination nodes and a bit-rate uniformly selected within the range from 50 Gbps to 1 Tbps, with 50 Gbps granularity. All experiments were run on a virtual machine with available 2 cores of Intel Xeon E5 series CPU at 2.90 GHz and 32 GB RAM using CPLEX v12.5 solver, executed through Java interface, with a 1-hour run-time limit.

In Table 3, average spectrum usage and time are presented for various ILP models. Normally, spectrum usage should be the same for all models considering given DCs and DC placement problem, respectively. However, for larger demands set, the results start to vary, as it is only possible to obtain feasible (not optimal) solution within given 1-hour run-time limit. Note, the corresponding time may be lower than 1 hour, as this is the average value. In particular, only part of the results may be feasible, while the rest of them may be optimal obtained in shorter time. The lack of results reflects the situation when out of memory error has occurred for all studied cases. As it can be observed, the required computational time increases rapidly even for small sizes of demands sets. In terms of models with given DCs, the SB_WOCD and LB_WCS are able to yield results for the largest demand sets due to the lower number of variables and constraints. Moreover, for 30 to 60 demands sets, LB_WCS ILP model provides worse results than SB_WOCS, despite that the required time is lower. It is possible due to the presence of far from optimal feasible results, while other instances were solved in shorter time. It can be justified with the Table 4, which presents the number of optimal, feasible and out of memory results for each tested case (the numbers are separated with the slash). In terms of models with DC placement, SB_WOCS_DC performs the best.

Next, we compare the models scalability. We study different number of candidate paths $|P| = \{2, 4\}$, fibers per SMFB $|K| = \{2, 4\}$ and number of DCs $R = \{2, 4\}$. In the presented results, let $S(X = \alpha)$ and $t(X = \alpha)$ denote the spectrum usage and time in seconds, respectively, when given parameter X has value α (e.g., $S(|P| = 2)$ denotes spectrum usage for 2 candidate paths). It is worth noting, the missing of results in the charts indicates that the run out of memory was obtained for all 5 tested cases.

Fig. 2 presents spectrum usage and time for models for candidate paths $|P| = \{2, 4\}$ and 20 demands. As it can be observed the increase of number of candidate paths yields slightly lower spectrum usage, however, required time for LB models grows quickly. In turn, number of candidate paths has the low impact on the required time of SB models. Fig 3 reports spectrum usage and time as a function of number of demands for SB_WOCS_DC for various number of candidate paths. The first conclusion is that for small instances with the larger number of candidate paths it is possible to find a lower optimal solution. However, in the presence of run-time limits, the feasible results are worse and the out of memory error is obtained for smaller instances (100 and 80 demands for 2 and 4 candidate paths, respectively). The results for models with given DCs are similar.

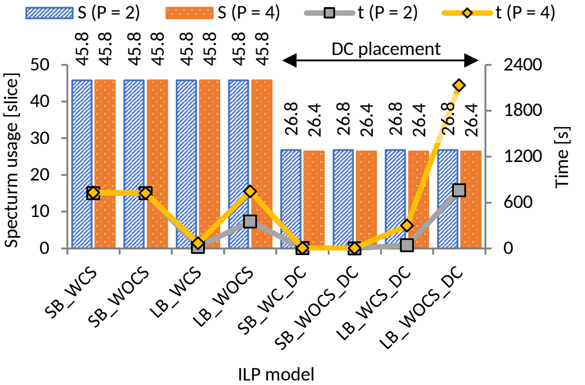


Figure 2: Spectrum usage and time as a function of ILP models for 2 and 4 candidate paths for 20 demands.

Fig 4 shows the spectrum usage as a function of ILP models for number of fibers per SMFBs $|K| = \{2, 4\}$ for 30 demands. Firstly, the increase of number of fibers allows reducing the spectrum

Table 3: Spectrum usage and time for ILP models for various number of demands.

	Number of demands															
	10	20	30	40	50	60	80	100	10	20	30	40	50	60	80	100
ILP model	Spectrum usage								Time [s]							
SB_WCS	35.0	45.8	70.3	86.5	—	—	—	—	0.4	721.8	3600.0	3600.0	—	—	—	—
SB_WOCS	35.0	45.8	70.3	91.0	99.0	142.3	201.0	233.0	0.3	721.7	3600.0	3600.0	3600.0	3600.7	3600.7	3600.7
LB_WCS	35.0	45.8	72.6	103.0	120.2	154.2	196.8	185.5	2.8	20.7	2005.1	2156.8	2309.0	2411.6	2602.0	2714.8
LB_WOCS	35.0	45.8	72.6	103.2	120.6	128.5	—	—	10.3	352.6	2892.1	2614.4	3236.4	3283.6	—	—
SB_WCS_DC	19.0	26.8	29.0	33.2	69.3	86.0	118.0	—	0.1	2.8	1682.7	3600.0	3600.0	3600.0	3600.1	—
SB_WOCS_DC	19.0	26.8	29.0	33.2	47.8	80.5	96.0	—	0.1	2.1	853.9	3600.0	3600.0	3600.0	3600.0	—
LB_WCS_DC	19.0	26.8	29.0	34.7	49.0	—	—	—	7.9	43.0	2378.1	3600.1	3600.0	—	—	—
LB_WOCS_DC	19.0	26.8	35.0	—	—	—	—	—	69.9	759.4	3600.1	—	—	—	—	—

Table 4: Number of optimal/feasible/out-of-memory results for ILP models for various number of demands.

	Number of demands							
ILP model type	10	20	30	40	50	60	80	100
SB_WCS	5/0/0	4/1/0	0/3/2	0/2/3	0/0/5	0/0/5	0/0/5	0/0/5
SB_WOCS	5/0/0	4/1/0	0/3/2	0/3/2	0/2/3	0/3/2	0/2/3	0/1/4
LB_WCS	5/0/0	5/0/0	3/2/0	4/1/0	2/3/0	4/1/0	4/1/0	1/1/3
LB_WOCS	5/0/0	5/0/0	1/4/0	4/1/0	1/4/0	2/0/3	0/0/5	0/0/5
SB_WCS_DC	5/0/0	5/0/0	3/2/0	0/5/0	0/4/1	0/2/3	0/5/0	0/0/5
SB_WOCS_DC	5/0/0	5/0/0	4/1/0	0/5/0	0/4/1	0/4/1	0/3/2	0/0/5
LB_WCS_DC	5/0/0	5/0/0	3/2/0	0/3/2	0/2/3	0/0/5	0/0/5	0/0/5
LB_WOCS_DC	5/0/0	5/0/0	0/4/1	0/0/5	0/0/5	0/0/5	0/0/5	0/0/5

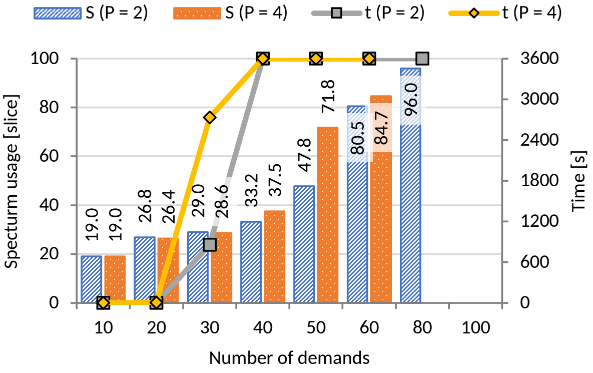


Figure 3: Spectrum usage and time as a function of number of demands for SB_WOCS_DC for 2 and 4 candidate paths.

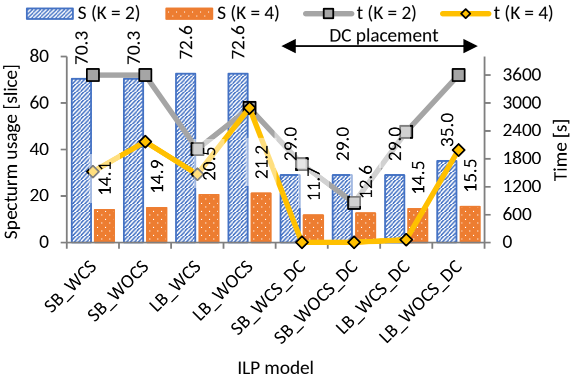


Figure 4: Spectrum usage and time as a function of ILP models for 2 and 4 fibers per SMFB for 20 demands.

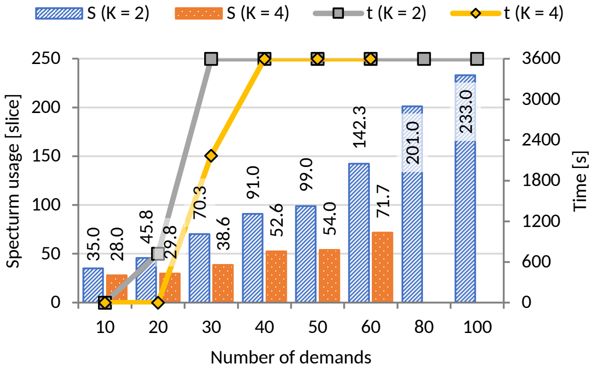


Figure 5: Spectrum usage and time as a function of number of demands for SB_WOCS for 2 and 4 fibers per SMFB.

usage around 46% and 11% for given DCs and DC placement, respectively. Surprisingly, the computational time is also lower. It may follow from the fact, that for a higher number of fibers, it is easier to allocate traffic and find a feasible good quality solution and quicker discard other worse solutions. Next, Figs 5 and 6 present spectrum usage and time as a function of number of demands for SB_WOCS and SB_WOCS_DC ILP models, respectively, for number of fibers per SMFB $|K| = \{2, 4\}$. We can easily observe that the run out of memory occurs for smaller demands sets when the number of fiber is higher, because the number of decision variables and constraints in ILP model is higher.

Fig 7 reports the spectrum usage and time as a function of ILP models for number of DCs $R = \{2, 4\}$ and 20 demands. The increase of number of DCs from 2 to 4 allows reducing required spectrum usage around 41% and 45% for given DCs and DC placement problems, respectively. Note, that for higher number of DCs, the lower time is required to solve the model. Figs 8 and 9 present spectrum usage and time as a function of number of demands for SB_WOCS and SB_WOCS_DC ILP models, respectively, for number of DCs $R = \{2, 4\}$. Despite that for higher number of DCs the spectrum utilization and required computational time is lower, again the run out of memory error occurs for smaller demands set sizes due to the higher number of variables and constraints in ILP instances.

7 CONCLUSIONS

In this paper, we study RSSA of anycast demands in SS-FONs. We analyze four problem versions which differ with the core switching possibility (WCS and WOCS) and DC location scenario (given DCs and DC placement involved into optimization task). Then, we define all RSSA versions using two modeling techniques

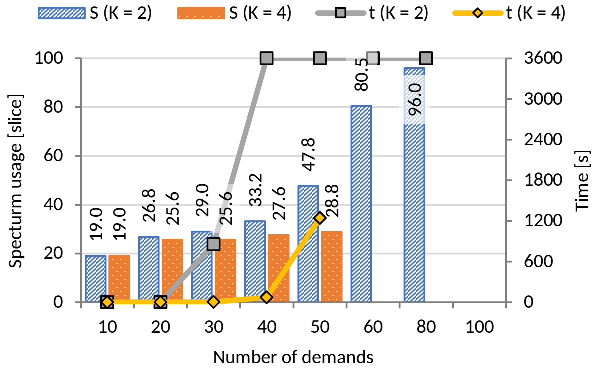


Figure 6: Spectrum usage and time as a function of number of demands for SB_WOCS_DC for 2 and 4 fibers per SMFB.

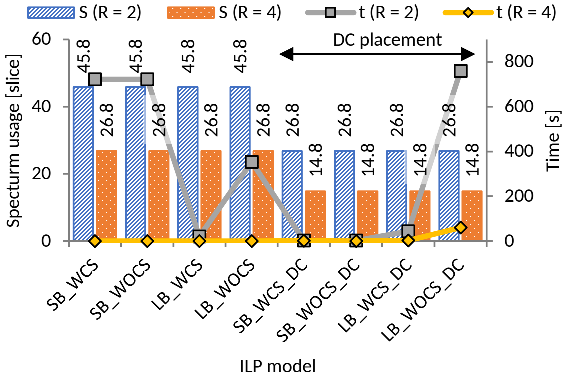


Figure 7: Spectrum usage and time as a function of ILP models for 2 and 4 DCs for 20 demands.

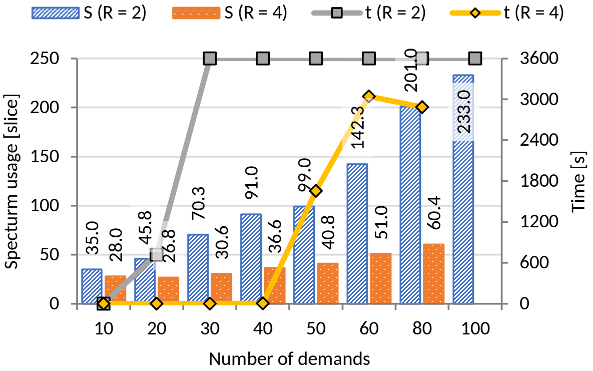


Figure 8: Spectrum usage and time as a function of number of demands for SB_WOCS for 2 and 4 DCs

(slice-based (SB) and lightpath-based (LB)) proposing eight ILP models. Next, we compare models in terms of complexity (measured in the number of variables and constraints), processing time and scalability. The analysis reveals that the complexity of SB models strongly depends on the number of demands while the complexity of LB models on the number of candidate paths and available spectrum width. The core switching possibility increases complexity of SB models while decreases complexity of LB models. The incorporation of DC placement into optimization task makes problem more complex considering both — SB and LB models. Concluding simulations, LB_WCS and SB_WOCS_DC provide the best performance for RSSA with given DC location and DC location selection problem, respectively.

In the future work, we plan to further study RSSA complexity considering different traffic types and network survivability.

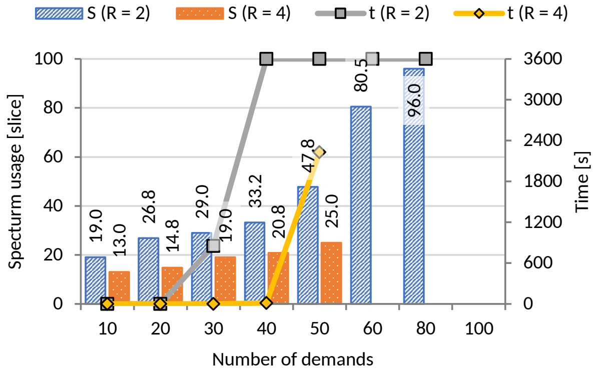


Figure 9: Spectrum usage and time as a function of number of demands for SB_WOCS_DC for 2 and 4 DCs

ACKNOWLEDGMENTS

The work was supported by National Science Centre, Poland under Grant 2015/19/B/ST7/02490 and statutory funds of the Department of Systems and Computer Networks, Wroclaw University of Science and Technology under Grant 0402/0112/18.

REFERENCES

[1] CISCO. 2017. *The Zettabyte Era: Trends and Analysis.* Technical Report. CISCO. 1–29 pages.

[2] P. S. Khodashenas, J. M. Rivas-Moscoso, D. Siracusa, F. Pederzolli, B. Shariati, D. Klonidis, E. Salvadori, and I. Tomkos. 2016. Comparison of Spectral and Spatial Super-Channel Allocation Schemes for SDM Networks. *J Lightwave Technol* 34, 11 (2016), 2710–2716.

[3] MirosÅĈaw Klinkowski, Piotr Lechowicz, and Krzysztof Walkowiak. 2018. Survey of resource allocation schemes and algorithms in spectrally-spatially flexible optical networking. *Opt Switch Netw* 27 (2018), 58 – 78.

[4] D. Klonidis, F. Cugini, O. Gerstel, M. Jinno, V. Lopez, E. Palkopoulou, M. Sekiya, D. Siracusa, G. Thouenon, and Ch. Betoule. 2015. Spectrally and Spatially Flexible Optical Network Planning and Operations. *IEEE Comm Mag* 53, 2 (2015), 69–78.

[5] Guifang Li, Neng Bai, Ningbo Zhao, and Cen Xia. 2014. Space-division multiplexing: the next frontier in optical communication. *Adv. Opt. Photon.* 6, 4 (Dec 2014), 413–487.

[6] Y. Li, N. Hua, and X. Zheng. 2015. Routing, wavelength and core allocation planning for multi-core fiber networks with MIMO-based crosstalk suppression. In *2015 Opto-Electronics and Communications Conference (OECC).* 1–3.

[7] A. Muhammad, G. Zervas, G. Saridis, E. H. Salas, D. Simeonidou, and R. Forchheimer. 2014. Flexible and synthetic SDM networks with multi-core-fibers implemented by programmable ROADMs. In *2014 The European Conference on Optical Communication (ECOC).* 1–3.

[8] A. Muhammad, G. Zervas, D. Simeonidou, and R. Forchheimer. 2014. Routing, spectrum and core allocation in flexgrid SDM networks with multi-core fibers. In *Proc. Optical Network Design and Modeling (ONDM).* 192–197.

[9] J. Perello, J. M. GenÃŤ, A. PagÃĺs, J. A. Lazaro, and S. Spadaro. 2016. Flexgrid/SDM backbone network design with inter-core XT-limited transmission reach. *J Opt Commun Netw* 8, 8 (2016), 540–552.

[10] Michal Pióro and Deepankar Medhi. 2004. *Routing, Flow, and Capacity Design in Communication and Computer Networks.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

[11] C. Rottondi, P. Boffi, P. Martelli, and M. Tornatore. 2017. Routing, Modulation Format, Baud Rate and Spectrum Allocation in Optical Metro Rings With Flexible Grid and Few-Mode Transmission. *J Lightwave Technol* 35, 1 (2017), 61–70.

[12] Cristina Rottondi, Massimo Tornatore, and Giancarlo Gavioli. 2013. Optical Ring Metro Networks With Flexible Grid and Distance-Adaptive Optical Coherent Transceivers. *Bell Labs Technical Journal* 18, 3 (2013), 95–110.

[13] G. M. Saridis, D. Alexandropoulos, G. Zervas, and D. Simeonidou. 2015. Survey and Evaluation of Space Division Multiplexing: From Technologies to Optical Networks. *IEEE Commun Surv Tut* 17, 4 (2015), 2136–2156.

[14] Krzysztof Walkowiak. 2016. *Modeling and Optimization of Cloud-Ready and Content-Oriented Networks* (1st ed.). Springer Publishing Company, Incorporated.

[15] K. Walkowiak, P. Lechowicz, M. Klinkowski, and A. Sen. 2016. ILP modeling of flexgrid SDM optical networks. In *Proc. Telecommunications Network Strategy and Planning Symposium (Networks).* 121–126.

[16] Liang Zhang, Nirwan Ansari, and Abdallah Khreishah. 2016. Anycast Planning in Space Division Multiplexing Elastic Optical Networks With Multi-Core Fibers. *IEEE Commun Lett* 20, 10 (2016), 1983–1986.

Extended linear formulation of the pump scheduling problem in water distribution networks

Gratien Bonvin
Center for Applied Mathematics,
Mines ParisTech, PSL Research University
Sophia Antipolis, France
gratien.bonvin@mines-paristech.fr

Sophie Demassey
Center for Applied Mathematics,
Mines ParisTech, PSL Research University
Sophia Antipolis, France
sophie.demassey@mines-paristech.fr

ABSTRACT

This paper presents a generic non-compact linear programming approximation of the pump scheduling problem in drinking water distribution networks. Instead of relying on the binary on/off status of the pumps, the model draws on the continuous duration of activation of pump combinations, whose entire set is computed in a preprocessing step by ignoring the pressure variation in the water tanks. Preprocessing is accelerated using network partition and symmetry arguments. A combinatorial Benders decomposition-based local search takes the approximated solution as input to derive a feasible solution. Our experiments on two different benchmark sets, with fixed- or variable-speed pumps, show the accuracy of the approximated formulation and the ability of the matheuristic to compute near-optimal solutions in seconds, where concurrent, more specialized approaches need minutes or hours.

1 INTRODUCTION

With the evolution of the power sector – because dynamic pricing is a savings opportunity for water network operators [6] – together with advances in mixed-integer nonlinear programming (MINLP), recent years have seen a renewed interest in minimizing the pumping costs in drinking water distribution networks.

The so-called *pump scheduling problem* is a hard combinatorial non-convex optimization problem. A variety of solution approaches have been investigated, but they often inefficiently deal with large or medium networks, and many small instances are still open. A first category of approaches (e.g. [6, 7, 10]) combine a numerical simulator, to compute the feasible hydraulic balances, with an exact or heuristic optimization algorithm, to schedule the pump operations at minimum cost. Separating feasibility from optimization makes the convergence of these approaches slow, resulting in sub-optimal solutions. A second category of approaches formulate the whole problem as a MINLP with simplified hydraulic constraints, based either on piecewise-linear approximations (e.g. [8, 9]) or convex relaxations [2, 3, 13]. These approaches only apply to small networks, because of the combinatorial nature of the models, and they may return impracticable solutions. Instead, Burgschweiger et al. [4] keep the non-convex constraints in their model but relax the binary on/off pump activation variables by aggregating them. This relaxation is only suitable to large city-wide networks where dozen of pumps are installed in parallel in each pumping station.

In this paper, we are interested in tackling intermediate-size networks with a mixed approach. We propose to approximate the MINLP model by decoupling feasibility and optimization in the way of Dantzig-Wolfe decomposition: by ignoring the pressure variation in the water tanks, we can compute the feasible hydraulic balances for all pump configurations as a preprocessing step, then derive a non-compact linear programming (LP) model based on the durations of activation of these configurations. We apply network partition and symmetry arguments to accelerate the preprocessing without hindering optimality. While the approximated LP solutions may be accurate enough to be practically implemented in pump controllers, we also propose to derive feasible solutions for the original MINLP with a local search approach, adapted from the combinatorial Benders decomposition of Naoum-Sawaya et al. [10]. Finally, we generalize the approach to networks with variable-speed pumps or pressure-reducing valves, which are often overlooked in the literature.

Experiments on the Poormond [6] and Van Zyl [17] benchmark networks show the efficiency of our preprocessing, the accuracy of our approximation, and the potential of the overall method to compute near-optimal solutions within seconds where concurrent methods [3, 6, 8, 10, 13] need minutes or hours to compute solutions of higher costs.

The paper is organized as follows: Section 2 defines the problem and describes the standard MINLP formulation in a simplified case. Section 3 presents our non-compact LP formulation and preprocessing reduction techniques and Section 4 the heuristic. Computational results are given in Section 5 and conclusions and perspectives in Section 6.

2 PUMP SCHEDULING PROBLEM

This section describes the problem and a standard formulation in the special case, for the sake of simplicity, of a network equipped with unidirectional pipes, fixed-speed pumps and no valves. Comprehensive formulations for more general networks can be found e.g. in [3, 4].

As illustrated in Figure 1, a water distribution network can be represented as a directed graph $\mathcal{G} = (J, L)$ with sources J_S, junctions J_J and tanks J_T as nodes J, and pipes L_P and pumps K as arcs L. Given a time horizon $\mathcal{T}$ of typically one day, the system dynamics are driven by the water demand rate $D \in \mathbb{R}_+^{J_J \times \mathcal{T}}$ at the junctions and are governed by complex hydraulic laws of conservation of flow and pressure through the network. The problem is to schedule the pump operations over $\mathcal{T}$ in order to continuously satisfy the demand and the allowed filling level of the tanks, while minimizing the operation cost.

ISBN 978-3-89318-079-0 on OpenProceedings, ISSN 2510-7437

DOI: 10.5441/002/inoc.2019.04

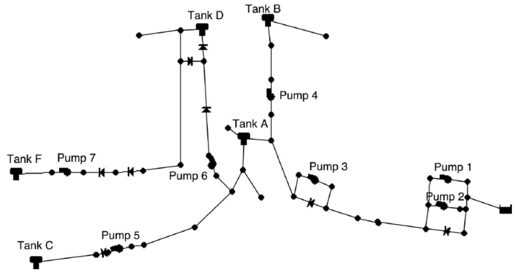


Figure 1: The Poormond network

A standard model is defined as follows, with $x \in \{0,1\}^{K\times\mathcal{T}}$ the binary on/off state of the pumps, $q \in \mathbb{R}_+^{L\times\mathcal{T}}$ the flow rate through the arcs, and $h \in \mathbb{R}_+^{J\times\mathcal{T}}$ the head at the nodes (defined as the sum of pressure and elevation):

$$(P) : \min_{x,q,h} \sum_{t\in\mathcal{T}}\sum_{k\in K} C_t \Delta_t \Gamma_k(x_{kt}, q_{kt}) \qquad (1)$$

$$s.t. \sum_{ij\in L} q_{ijt} = \sum_{ji\in L} q_{jit} + D_{jt}, \qquad t\in\mathcal{T}, j\in J_J \quad (2)$$

$$\sum_{ij\in L} q_{ijt} - \sum_{ji\in L} q_{jit} = \frac{\sigma_j}{\Delta_t}(h_{jt} - h_{jt-1}), \qquad t\in\mathcal{T}, j\in J_T \quad (3)$$

$$h_{j0} = H_j^0, \qquad j\in J_T \quad (4)$$

$$H_{jt}^{\min} \le h_{jt} \le H_{jt}^{\max}, \qquad t\in\mathcal{T}, j\in J \quad (5)$$

$$q_{kt} \le Q_k^{\max} x_{kt}, \qquad t\in\mathcal{T}, k\in K \quad (6)$$

$$h_{it} - h_{jt} = \Phi_{ij}(q_{ijt}), \qquad t\in\mathcal{T}, ij\in L_P \quad (7)$$

$$(h_{jt} - h_{it} - \Psi_{ij}(q_{ijt}))x_{ijt} = 0, \qquad t\in\mathcal{T}, ij\in K. \quad (8)$$

In this model, the time horizon is discretized $\mathcal{T} = \{1,\dots,T\}$ with a resolution Δ_t of typically 1 hour in which the system is assumed to operate in steady state. Constraints (2) and (3) enforce the conservation of flow at junctions and tanks. In (3), a tank $j \in J_T$ is assumed to be a vertical cylinder of area σ_j which links the stored water volume linearly to the head. Bounds on heads (4) and (5) depend on the node types: for a tank $j \in J_T$, they are given by the minimum and maximum filling levels, and by the initial level H_j^0; for a junction $j \in J_J$, $H_{jt}^{\min}$ stands for the minimum pressure required to serve demand D_{jt}; for a source $j \in J_S$, the head is fixed exogenously. Constraints (7) enforce the head losses due to friction in pipes. For each directed pipe $ij \in L_p$, the head loss can be accurately approximated by a quadratic function Φ_{ij} of the flow. Constraints (6) bound the flow through the pumps and bind flow values and pump activation states. Constraints (8) model the head increase through active pumps (if $x_{ijt} = 1$). For each pump $k \in K$, head-flow coupling function Ψ_k can be accurately fitted from operating points as a quadratic function. Finally, the financial cost is mainly incurred by purchasing electricity for pumping, see objective (1) with $C_t \ge 0$ the actualized electricity price on period t, and Γ_k the power consumption of pump k defined by a linear curve fit $\Gamma_k(x,q) = \lambda_k x + \mu_k q$.

Model (P) is a non-convex MINLP which is often untractable even for small networks. One option to solve (P) is to decrease the resolution of the time discretization, and thus the model size, but it has potential drawbacks: (1) the steady-state assumption is less realistic over longer time steps, (2) this artificially reduces the set of feasible schedules, and (3) the optimum increases accordingly. We investigate the opposite option, closer to the reality, by allowing to operate pumps at any time.

3 AN APPROXIMATED NON-COMPACT MODEL

We present a new approximated LP formulation of the pump scheduling problem which separates the computation of the hydraulic balances from the optimization of the schedule. The description is first given in the context of networks where fixed-speed pumps are the only operable elements. We then generalize the definition to networks with valves or variable-speed pumps.

3.1 Tank head approximation

Let $A \subseteq L$ be the set of operable elements of the network, and assume for now that $A = K$ the set of fixed-speed pumps. We call *configuration* any subset $s \subseteq A$, also denoted by its indicator function $I^s \in \{0,1\}^A$ defined by $I_a^s = 1 \Leftrightarrow a \in s$. Configuration s is said *active* at time $t \in \mathcal{T}$ if all its elements, and only these elements, are active (e.g. pumps are on): $x_{at} = 1 \iff a \in s$.

Looking at model (P), a configuration $s \subseteq A$ can be active at time $t \in \mathcal{T}$ only if, given the tank levels and the allowed variation range, the pumps belonging to the configuration offer together enough power to increase head and satisfy the demand rate D_t at all junctions. Because water tanks are usually very large containers, the level variation during 1 hour or less is relatively limited when compared to their heights. We propose to ignore this variation and assume that, at any tank $j \in J_T$, the head is fixed to an arbitrary value H_j^* during time step t. Under this assumption, we can easily compute the hydraulic balance for supplying demand D_t with configuration s. Indeed, by definition, it is a solution $(q_t, h_t) \in \mathbb{R}_+^L \times \mathbb{R}_+^J$ of the non-convex system $\{(2),(5),(6),(7),(8)\}$ restricted to a single time step $\mathcal{T} = \{t\}$ with fixed pump states $x_{at} = I_a^s \ \forall a \in A$ and fixed tank heads $h_{jt} = H_j^* \ \forall j \in J_T$. It is known [5, 6, 16] that this equation system, that we denote $\mathcal{F}_t(I^s, H^*)$, has at most one solution which can quickly be computed, in particular, by the Newton method [16] which is implemented in the popular numerical simulator EPANET [11].

To estimate if a configuration s may supply demand rate D_t, we thus propose to check the feasibility of $\mathcal{F}_t(I^s, H^*)$ with tank heads arbitrarily fixed to their median values $H_j^* = (H_j^{\min} + H_j^{\max})/2$ for all $j \in J_T$. If feasible and (q^s, h^s) its solution, we compute the corresponding instantaneous power consumption $P_t^s = \sum_{k\in s\cap K} \Gamma_k(1, q_{kt}^s)$ and net fill rate $R_{jt}^s = \sum_{ij\in L} q_{ijt}^s - \sum_{ji\in L} q_{jit}^s$ at each tank $j \in J_T$. We denote by $S_t^* = \{s \subseteq A \mid \mathcal{F}_t(I^s, H^*) \neq \emptyset\}$ the set of configurations which may be active during time step $t \in \mathcal{T}$ according to this assumption.

3.2 Configuration scheduling

Given these estimates, we reformulate the pump scheduling problem as a configuration scheduling problem where, at any time step t, any configuration $s \in S_t^*$ is allowed to be active for a duration $0 \le \delta_{st} \le \Delta_t$ within the time step. Modelling the system dynamics boils down to enforce tank head conservation between consecutive time steps, then leads to the following linear program:

$$(P^*): \min_{\delta,h} \sum_{t\in T} C_t \sum_{s\in S_t^*} P_t^s \delta_{st} \qquad (1')$$

$$s.t. \sum_{s\in S_t^*} \delta_{st} = \Delta_t, \qquad t\in\mathcal{T} \qquad (9)$$

$$h_{jt} - h_{jt-1} = \sum_{s\in S_t^*} \frac{R_{jt}^s}{\sigma_j}\delta_{st}, \qquad t\in\mathcal{T}, j\in J_T \qquad (3')$$

$$H_{jt}^{\min} \leq h_{jt} \leq H_{jt}^{\max}, \qquad t\in\mathcal{T}, j\in J. \qquad (5')$$

In (P^*), pumps can thus be operated during time steps. Furthermore, (P^*) is an approximation of (P), and not just a relaxation: an optimal configuration s at time t for (P) may not belong to S_t^* if it can indeed satisfy demand D_t but not under the half-filled tanks assumption; if s does belong to S_t^*, otherwise, then its actual consumption may not be P_t^s precisely.

3.3 Processing configurations

Computing one hydraulic balance with the Newton algorithm is almost immediate, but the exponential number of configurations to evaluate, in $O(T2^{|A|})$, can make it computationally challenging to build (P^*). We propose two techniques to significantly reduce the computation time without hindering optimality.

First, we exploit the symmetries, which are frequent in real data. For instance, when demand rate $D_t \in \mathbb{R}^{J_J}$ is constant at all junctions over a number of time steps, then the configurations need to be evaluated on only one time step, since if $D_t = D_{t'}$, then $S_t^* = S_{t'}^*$, $P_t^s = P_{t'}^s$ and $R_t^s = R_{t'}^s$ for all $s \in S_t^*$. Another symmetry arises when pumps with identical characteristics are installed in parallel at a pumping station (see e.g. pumps 1 and 2 in Figure 1). Only one symmetric configuration is then evaluated.

Second, we exploit a partition of the network along the tank nodes. Precisely, we consider the graph $\mathcal{G}' = (J', L')$ obtained from $\mathcal{G}$ by duplicating each tank node $j \in J_T$ for each incoming arc $ij \in L$ as a new node denoted j^i, i.e. $J' = J \cup \{j^i \mid ij \in L, j \in J_T\}$ and $L' = L \cup \{ij^i \mid ij \in L, j \in J_T\} \setminus \{ij \in L \mid j \in J_T\}$. Once the heads at tanks $j \in J_T$ (and at duplicate nodes j^i) are fixed to their median values H_j^*, the arc flow and node head values become independent in each connected component of $\mathcal{G}'$. We thus compute the set of feasible configurations $S_t^C \subseteq A \cap L_C$ independently for each connected component $C = (J_C, L_C) \in \mathcal{CC}(\mathcal{G}')$. These sub-configurations are then combined by summation: for all $s \subseteq A$, $P_t^s = \sum_{C\in\mathcal{CC}(\mathcal{G}')} P_t^{s\cap L_C}$ and $R_{jt}^s = \sum_{C\in\mathcal{CC}(\mathcal{G}') \mid j\in J_C} R_{jt}^{s\cap L_C}$ for $j \in J_T$. Hence, the network partition reduces both the number of computations and their complexity, being evaluated on smaller graphs. Furthermore, the symmetry condition on constant demand occurs with a higher frequency when regarding the subsets of junctions independently.

3.4 Generalization

In most water distribution networks, not only pumps but also valves $V \subseteq L$ of different types can be operated. Like fixed-speed pumps, gate valves and check valves have only two possible states (close or open) and can be modeled in (P) with a binary variable for each time step (see e.g. [4]). The definition of configuration can then be extended to $A = K \cup V$ the set of pumps and valves, saying that a valve is active if it is close. Furthermore, according to [12], $\mathcal{F}_t(I^s, H^*)$ has still at most one solution.

The presence of variable-speed pumps or pressure-reducing valves deserves more attention as they admit a continuous range of operation modes. A variable-speed pump is either off or operated within an allowed range of speed. For a pressure-reducing valve, the amount of pressure reduction is chosen within a given range and a binary state indicates the direction of the flow (see e.g. [15]). As suggested in [5], we propose to approximate the allowed operation range of each pressure-reducing valve or variable-speed pump a by a discrete set of sample values A_a. The set A is then augmented with these sample values and a configuration is now defined as $s \subseteq A$ with $|s \cap A_a| \leq 1$. Once pump speeds and pressure reductions are fixed in configuration s, the Newton method can quickly solve $\mathcal{F}_t(I^s, H^*)$ as before.

Hence, the approximation model (P^*) and configuration processing scheme apply to a comprehensive class of water networks. Still, the number of configurations to evaluate grows exponentially with the number of operable elements, unless the network partition separates these elements in small sets so that the growth becomes near linear.

4 BENDERS DECOMPOSITION-BASED HEURISTIC

This section describes an adaptation of the combinatorial Benders decomposition of [10] to search, in the neighborhood of the approximated solutions of (P^*), feasible solutions to the pump scheduling problem with pump aging constraints.

4.1 Pump aging

While in practice pumps can be operated at any time, too frequent switches are prohibited to prevent premature pump aging. Ghaddar et al. [6] proposed to enforce the following constraints in model (P) for each pump $k \in K$:

$$\sum_{t\in\mathcal{T}} y_{kt} \leq N, \qquad (10)$$

$$y_{kt} \geq x_{kt} - x_{k(t-1)}, \qquad t\in\mathcal{T} \qquad (11)$$

$$x_{kt'} \geq y_{kt}, \qquad t\in\mathcal{T}, t'\in[t, t+\tau_1] \qquad (12)$$

$$z_{kt} \geq x_{k(t-1)} - x_{kt}, \qquad t\in\mathcal{T} \qquad (13)$$

$$x_{kt'} \leq 1 - z_{kt}, \qquad t\in\mathcal{T}, t'\in[t, t+\tau_0] \qquad (14)$$

with y_{kt} (resp. z_{kt}) a binary variable equal to 1 if pump $k \in K$ is switched on (resp. off) at time t, N the maximal number of times a pump can be switched on, τ_1 (resp. τ_0) the minimum continuous duration a pump is on (resp. off).

Naoum-Sawaya et al. [10] designed a combinatorial Benders decomposition approach, where the master integer linear program denoted (M) is initialized with the aging constraints (10)-(14) alone. At each iteration, (M) returns a candidate schedule $X \in \{0,1\}^{K\times\mathcal{T}}$ to evaluate: the EPANET simulator computes the hydraulic balance and power consumption at each time step, sequentially. If a hydraulic constraint is violated or if the partial cost exceeds the best solution known so far at a given time $\bar{t} \in \mathcal{T}$, then the partial schedule up to time $\bar{t}$ is discarded from the search by adding a no-good cut to master (M):

$$\sum_{t=1}^{\bar{t}} \left(\sum_{\substack{k\in K\\ X_{kt}=0}} x_{kt} + \sum_{\substack{k\in K\\ X_{kt}=1}} (1-x_{kt}) \right) \geq 1. \qquad (15)$$

The cut is also added with $\bar{t} = T$ each time X proves to be the new incumbent, i.e. an improving solution. The algorithm stops when (M) becomes unfeasible. The algorithm theoretically converges to a certified optimal schedule, but its slow convergence requires to limit the computation time.

4.2 Truncated Benders decomposition

In the original method [10], the objective function of master (M) is initialized to 0, then systematically redefined as the minimal distance to the new incumbent, so as to search the next candidate in a neighborhood. Thus, the algorithm improves the solution progressively, as in a local search, but it may start with a low quality solution. We propose to adapt this algorithm to conduct it explicitly as a heuristic to compute good solutions fast. To this end, we initialize the algorithm with an optimal approximate solution δ^* of (P^*), then adjust the distance function, i.e. the neighborhood, iteratively until finding a feasible solution.

More precisely, we first compute the duration $\delta^*_{at} = \sum_{s\in S^*_t} I^s_a \delta^*_{st}$ of activation of any pump or valve $a \in A$ during time step $t \in \mathcal{T}$ in the approximated solution. We expect that, in an optimal schedule x, if δ^*_{at} is close to Δ_t then a is active during t (i.e. $x_{at} = 1$), and that, if δ^*_{at} is close to 0 then a is inactive during t (i.e. $x_{at} = 0$). Furthermore, we estimate that the daily duration of activation $\sum_{t\in\mathcal{T}} x_{at}\Delta_t$ of a is close to $\sum_{t\in\mathcal{T}} \delta^*_{at}$. Hence, the minimization criterion of (M) is initialized to:

$$\sum_{a\in A}\sum_{t\in\mathcal{T}}(\Delta_{at} - x_{at}\Delta_t)^2 + \sum_{a\in A}(\sum_{t\in\mathcal{T}}\Delta_{at} - \sum_{t\in\mathcal{T}} x_{at}\Delta_t)^2 \quad (16)$$

with $\Delta_{at} = \begin{cases} \delta^*_{at} & \text{if } \delta^*_{at} \in \{0, \Delta_t\} \\ \alpha_{at} & \text{otherwise,} \end{cases}$ and $\alpha_{at} \in \{0, \delta^*_{at}, \Delta_t\}$ is a parameter of diversification which is initialized to δ^*_{at} to search first around the approximated solution of (P^*), and is then updated randomly at each iteration.

Another difference with [10], is that we generalize the method to networks with variable-speed pumps or pressure-reducing valves. In this context, we propose to use a non-convex NLP solver instead of the EPANET simulator to evaluate the candidate solutions $X \in \{0,1\}^{A\times T}$ by solving the slave program, i.e. the standard MINLP formulation with the binary variables x fixed to values X. Note here that, unlike for the processing of the configurations (see Section 3.4), we do not extend the definition of the set of operable elements A by discretizing the continuous state range for variable-speed pumps and pressure-reducing valves. Finally, because we run the Benders decomposition as a heuristic, the slave problem is not required to be solved at optimality. Hence, when the global optimization of the restricted non-convex NLP is too time consuming, a fast local optimization solver can be used instead.

5 COMPUTATIONAL RESULTS

We experimented the full heuristic, sketched in Algorithm 1, on two benchmark sets: Poormond [6] and Van Zyl [17]. In this section, we evaluate the solutions in comparison with the best solutions known so far for these instances (see [3] for a comparative analysis of the results published in [3, 6, 10, 13] on Poormond).

Algorithm 1: Heuristic for (P)

```
for $C \in \mathcal{CC}(\mathcal{G}')$, $s \subseteq A \cap L_C$, $t \in \mathcal{T}$ do
    solve $\mathcal{F}(I^s, H^*)$ with Newton method
compute $P^s_t$, $R^s_t$ by summation $\forall s \in S^*_t, t \in \mathcal{T}$
solve LP $(P^*)$: get $\delta^*$
initialize $(M)$: min (16) s.t. (10)-(14)
while unfeasible do
    solve MIQP $(M)$: get $X$
    simulate $X$ with Newton method or NLP solver
    if $X$ feasible then
        return $X$
    else
        add cut (15) to $(M)$
        update (16)
```

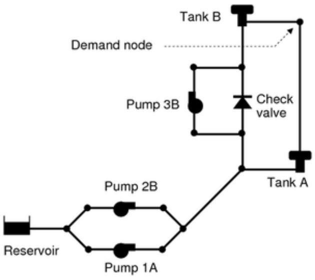


Figure 2: The Van Zyl network

5.1 Experimental set-up

The Poormond network, depicted in Figure 1, was derived by [6] from the real water distribution network of Richmond, England. It is a medium-size network with 47 nodes including 1 source and 5 tanks, 44 pipes, 7 fixed-speed pumps and 4 gate valves. The benchmark set has five daily instances, denoted from P21 to P25, each corresponding to the real dynamic power tariff, available at [14], that occurred each day in range May 21-25, 2013. The time horizon is discretized in $T = 48$ time steps of $\Delta_t = 1/2$ hour each. Pumps are required to stay on for at least 1 hour ($\tau_1 = 2$), off for at least 1/2 hour ($\tau_0 = 1$), and to be activated at most $N = 6$ times. The Van Zyl network [17], depicted in Figure 2, is a fictive, small but complex network with 1 source, 2 tanks, 15 pipes, 1 check valve and 3 pumps assumed to be variable-speed pumps after [8]. We experimented on this network using the same 5 tariff profiles set at the same time resolution. We denote the five instances Z21 to Z25 accordingly.

The computations were performed on a Xeon E5-2650V4 2.2GHz with 254 GB RAM. The processing of the configurations, including the Newton method, was implemented in Python, while the default LP solver and MINLP solver of Gurobi 7.0.2 were run on one thread to solve (P^*) and (M) respectively. For the Van Zyl instances, the slave problems of the Benders decomposition were solved with the default non-convex NLP local solver of Bonmin [1]. The step to discretize the allowed pump speed range was empirically fixed to $|A_a| = 6$.

The heuristic solutions are compared with the best solutions returned by the branch-and-cut method of [3] running in 1 hour under the same experimental set-up. In [3], a MILP outer approximation of (P) is solved with a LP-branch and bound augmented with user cuts: at each integer node, the corresponding pump configuration is evaluated as in the Benders decomposition here described, and no-good cuts generated accordingly. To make the comparison of the results valid, we implemented the exact evaluation procedure described in [3]. It includes, for the Poormond instances, a primal heuristic which slightly adjusts the duration of activation of the pumps to correct the small bound violations induced by the fixed time discretization.

5.2 Quality of the approximation

Figure 3 illustrates on instance P21, for each of the 7 pumps of the Poormond network, the optimal pump schedule δ^* returned by (P^*) after recomputation of the real flows and heads (in blue), and the feasible pump schedule returned by our heuristic (in pink). Figure 4 depicts the water filling profiles of the 5 tanks for both solutions and, below, the dynamic electricity tariff profile.

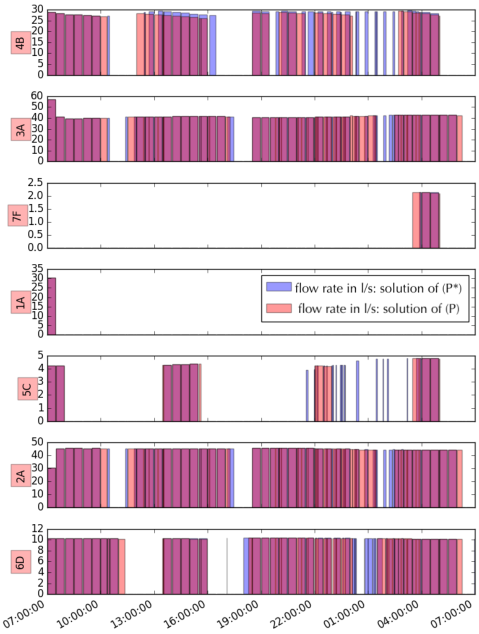


Figure 3: Approximated and feasible schedules

We observe for the approximated solution on Figure 4 that the water levels in the tanks (the blue curves) only slightly fall outside the allowed range (delimited by the black lines) which indicates that the approximated pump schedule is close to be practically feasible. When considering the modelling errors and the security margins and ignoring the formal pump aging constraints, this approximated schedule could probably directly be applied as a command for the real-time control of the pumps.

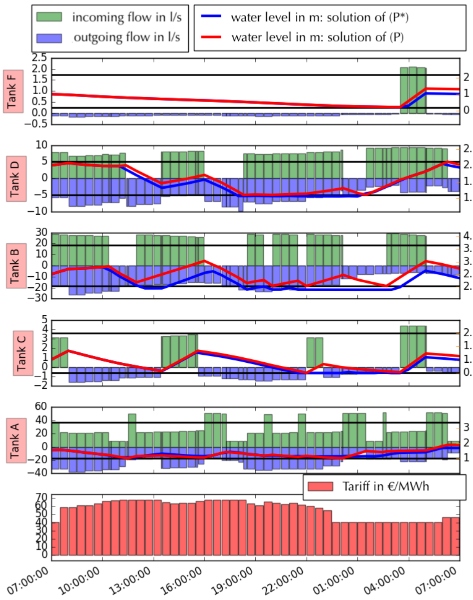


Figure 4: Tank levels in the approximated and feasible solutions

The near feasibility of the solution attests the relevancy of approximating the tank heads to their median values. Indeed, we observe an average relative deviation lower than 1% between the flow profiles delivered by the pumps, before and after recomputation with the actual tank heads. This confirms our hypothesis, we observed on a sample configuration, that the error on the flow due to this approximation is significant only when some tanks are empty while others are full. Here, on the contrary and as expected, the filling profiles of the 5 tanks all follow the same dynamic generated by the variable electricity tariff.

Perhaps more surprising, we observe on Figure 3 that the approximated and feasible pump schedules overlap extensively, from 77% for pump 5C to 100% for pump 1A, which indicates that the approximated solution mostly satisfies the fixed time discretization constraint of model (P) and the pump aging constraints, although they are entirely relaxed in (P^*). Actually, because (P^*) has comparatively few constraints $(O(T|J|))$, a basic solution has then few columns. In other words, only a fraction of the configurations over all the time steps have a non-zero duration in the optimal approximated schedule. For instance P21 depicted here, only 104 configurations are active which corresponds to 3% of the generated configurations, and, on the 48 time steps, 15 are associated to an unique configuration. This explains why pumps are activated at reasonable frequency, from 1 for pump 1A to 21 for pump 4B, in the approximated solution.

Finally as the approximated and feasible solutions are close, their costs (111.03 euros for the former and 117.50 for the latter) present a moderate gap (+6%). We observed the same proximity on all the Poormond instances and on all the Van Zyl instances too. For example, in instance Z21, the approximated solution has only 50 active configurations on more than 20,000 candidates over the 48 times steps and it satisfies all the pump aging constraints. Only one iteration of the Benders decomposition and a slight adjustment of

	Computation time (s)				Cost (euros)			
Day	conf	LP	TBD	Total	TBD	best[3]	TBD/LB	best/LB
P21	1.6	<0.1	16.3	17.9	117.50	112.48	8.2%	4.1%
P22	1.6	<0.1	11.2	12.8	118.55	116.49	5.6%	3.9%
P23	1.6	<0.1	8.0	9.6	120.93	120.85	4.1%	4.0%
P24	1.6	<0.1	10.9	12.5	137.05	134.99	4.6%	3.1%
P25	1.6	<0.1	21.2	22.8	98.74	92.53	9.8%	3.8%
Z21	2.1	<0.1	0.7	2.8	220.60	222.66	14.9%	15.7%
Z22	2.1	<0.1	1.7	3.8	230.07	230.69	14.1%	14.3%
Z23	2.1	<0.1	1.4	3.5	240.67	240.93	13.7%	13.8%
Z24	2.1	<0.1	0.6	2.6	267.77	268.91	14.4%	14.7%
Z25	2.1	<0.1	0.7	2.8	188.52	190.29	14.5%	15.3%

Table 1: Results of the heuristic

the pump speeds were needed to retrieve a feasible solution with a +4.2% cost deviation.

5.3 Performance of the heuristic

Table 1 summarizes the computational results of our heuristic on the 10 instances of Poormond and Van Zyl. On the left part, the computation times (in seconds) are detailed for each algorithmic component: the preprocessing of the configurations (conf), the solution of the approximated model (P^*) (LP), and the truncated Benders decomposition (TBD). The right part of the table gives the costs (in euros) of the solutions returned by our heuristic (TBD) compared to the solutions of the branch-and-cut approach returned in 1 hour (best [3]); TBD/LB and best/LB denote the respective optimality gaps to the best know lower bound also returned by the branch-and-cut in 1 hour.

We observe that the heuristic computed good quality solutions fast. About 2 seconds were required to generate an approximated schedule, mostly to preprocess the set of configurations since solving the LP was immediate. The graph partition has a great impact on the number of configurations to process. On the Van Zyl network, for example, the partition creates two components: the one with all the operable elements but no demand – resulting in 456 configurations which are identical for each time step (even for each instance, actually) – and the other with the unique demand node, only two pipes and no operable elements – resulting in one configuration for each time step. Hence, we computed $456 + 48$ hydraulic balances instead of 456×48.

The truncated Benders decomposition ran in 14 seconds in average on Poormond and in 1 second on Van Zyl. It stopped with a feasible solution after the first iteration, except for instance P25 which required two iterations. On Poormond, the costs of the heuristic solutions were, in average, 2.9% higher than the best solutions, and up to 5% higher for instance P25. In comparison, the branch-and-cut required 306 seconds in average to compute a first feasible solution of comparable quality (at 2.6% of the final solutions). According to [3], our heuristic solutions also improve upon the solutions reported by [6] and [10] after 1 hour of computation. On Van Zyl, the heuristic computed in 3 seconds, in average, solutions which slightly improve upon the solutions found in 1 hour by the branch-and-cut. The average optimality gap is 14.3%. In comparison, [8] reported approximated solutions with a 30% optimality gap computed in 5 minutes by solving a MILP obtained by piecewise linearization of the non-convex constraints in (P).

6 CONCLUSION

We formulated the pump scheduling problem in water distribution network as a new generic non-compact linear program, based on the approximation of the head at the water tanks and on the relaxation of the pump aging constraints. This approximation turned out to be both tight and easy to solve when experimented on two networks with different characteristics. We were then able to quickly find low cost feasible solutions by searching in a neighborhood of the approximated solutions. These results lead us to believe that this method could deal with networks larger than with the currently known approaches. Failing to dispose of such study cases, we envisage to build new realistic instances to confirm our claim. Perspectives to extend our method are, first, to exploit the new LP approximation in a global optimization approach, and, second, to exploit historical data of network operations to build the configuration set.

REFERENCES

[1] Bonami, P., Biegler, L., Conn, A., Cornuéjols, G., Grossmann, I., Laird, C., Lee, J., Lodi, A., Margot, F., and Sawaya, N. An algorithmic framework for convex mixed integer nonlinear programs. *Discrete Optimization 5*, 2 (2008), 186–204.

[2] Bonvin, G., Demassey, S., Le Pape, C., Maïzi, N., Mazauric, V., and Samperio, A. A convex mathematical program for pump scheduling in a class of branched water networks. *Applied Energy 185* (2017), 1702 – 1711.

[3] Bonvin, G., Demassey, S., and Lodi, A. Pump scheduling in drinking water distribution networks with an LP/NLP-based branch and bound. Tech. rep., CMA, Mines ParisTech, 2018. http://sofdem.github.io/art/bonvin19lpnlp.pdf.

[4] Burgschweiger, J., Gnädig, B., and Steinbach, M. Optimization models for operative planning in drinking water networks. *Optimization and Engineering 10*, 1 (2009), 43–73.

[5] D'Ambrosio, C., Lodi, A., Wiese, S., and Bragalli, C. Mathematical programming techniques in water network optimization. *European J. of Operational Research 243*, 3 (2015), 774 – 788.

[6] Ghaddar, B., Naoum-Sawaya, J., Kishimoto, A., Taheri, N., and Eck, B. A Lagrangian decomposition approach for the pump scheduling problem in water networks. *European Journal of Operational Research 241*, 2 (2015), 490 – 501.

[7] Mackle, G., Savic, G. A., and Walters, G. A. Application of genetic algorithms to pump scheduling for water supply. In *First International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications* (1995), pp. 400–405.

[8] Menke, R., Abraham, E., and Stoianov, I. Modeling variable speed pumps for optimal pump scheduling. In *World Environmental and Water Resources Congress* (2016), pp. 199–209.

[9] Morsi, A., Geissler, B., and Martin, A. Mixed integer optimization of water supply networks. In *Mathematical Optimization of Water Networks*. Springer, 2012, pp. 35–54.

[10] Naoum-Sawaya, J., Ghaddar, B., Arandia, E., and Eck, B. Simulation-optimization approaches for water pump scheduling and pipe replacement problems. *European Journal of Operational Research 246*, 1 (2015), 293–306.

[11] Rossman, L. *EPANET*, 2000.

[12] Salgado-Castro, R. O. *Computer modelling of water supply distribution networks using the gradient method.* PhD thesis, Newcastle University, 1988.

[13] Shi, H., and You, F. Energy optimization of water supply system scheduling: Novel MINLP model and efficient global optimization algorithm. *AIChE J. 62*, 12 (2016), 4277–4296.

[14] Single Electricity Market Operator. http://www.sem-o.com. [accessed: 10-Nov-2016].

[15] Skworcow, P., Paluszczyszyn, D., and Ulanicki, B. Pump schedules optimisation with pressure aspects in complex large-scale water distribution systems. *Drinking Water Engineering and Science 7*, 1 (2014), 53–62.

[16] Todini, E., and Pilati, S. A gradient algorithm for the analysis of pipe networks. In *Computer Applications in Water Supply: Vol. 1—systems Analysis and Simulation*, B. Coulbeck and C.-H. Orr, Eds. Research Studies Press Ltd., 1988, pp. 1–20.

[17] van Zyl, J., Savic, D., and Walters, G. Operational optimization of water distribution systems using a hybrid genetic algorithm. *J. Water Res. Plann. Manage.* (2004), 160.

Risk averse management on strategic multistage operational two-stage stochastic 0-1 optimization for the Rapid Transit Network Design (RTND) problem

Luis Cadarso
European Institute for Aviation Training and Accreditation (EIATA), Universidad Rey Juan Carlos
Fuenlabrada, Madrid, Spain
luis.cadarso@urjc.es

Laureano F. Escudero
Estadística e Investigación Operativa, Universidad Rey Juan Carlos
Móstoles, Madrid, Spain
laureano.escudero@urjc.es

Ángel Marín
Aeronautical and Space Engineering School, Universidad Politécnica de Madrid
Madrid, Spain
angel.marin@upm.es

ABSTRACT

The Rapid Transit Network Design planning problem along a multi-period time horizon is treated by considering uncertainty in passenger demand, strategic costs and network disruption. The problem has strategic decisions about the timing to construct stations and edges, and operational decisions on the available network at the periods. The uncertainty in the strategic side is represented in a multistage scenario tree, while the uncertainty in the operational side is represented in two-stage scenario trees which are rooted with strategic nodes. The variability in the strategic cost along the time horizon as well as the variability in the lost passenger demand to the operational transit system current conditions could be very high. In order to avoid the negative impacts of low probability but high cost or high lost demand scenarios, some risk reduction measures should be considered. In this work the expected conditional stochastic dominance functional is modeled in two flavors. First, controlling the cost in the strategic scenarios in selected groups and clusters and second, controlling the lost passenger demand in the operational scenarios. Both flavors are time consistent.

KEYWORDS

Transportation, Rapid Transit Network Design, multistage multi-horizon scenario trees, 0-1 models, risk averse, matheuristic algorithms.

1 INTRODUCTION

Transportation systems are spatially distributed systems, which are vulnerable to different incidents that may occur. Despite the unpredictable nature of these incidents in terms of location, time and magnitude, effective mitigation methods should be designed from the very first strategic stage of design.

When designing a transport network, decisions are made according to an expected value for network state variables, such as infrastructure, vehicle, and traffic conditions, which are uncertain in a planning horizon of up to decades.

In order to find resilient network designs, different research approaches can be used, such as deterministic static, two-stage stochastic, multistage stochastic and robust optimization, among others. Robust optimization features solutions which are immune to data uncertainty [8, 23]. However, these solutions have been demonstrated to be too conservative and, then, expensive on a daily basis [11]. The key is that the recovery of the system in different operational scenarios in a given strategic scenario may not be as expensive as the introduction of traditional robustness concepts.

It is also well known that deterministic models do not provide high quality solutions if long planning horizons are considered, where variability of data is prominent. It should be pointed out that the optimization of the Risk Neutral (RN) model has the drawback of providing a solution that ignores the potential variability of the objective function value in the scenarios and, so, the occurrence of low-probability high-cost scenarios. Alternatively, risk averse measures could be considered.

This work aims at advancing the state-of-the-art of rapid transit network design by introducing a novel modeling approach for a stochastic recoverable robustness.

Review of the State-of-the-Art. In the context of rail Rapid Transit Network Design (RTND), a complete review is recently given in [27]. There is an extensive literature about deterministic RTND problems, where all parameters are assumed to be known with certainty [7, 10, 12, 13, 22, 25, 29, 30]. But, stochastic optimization is currently one of the most robust tools for decision making and broadly used in real-world applications in a wide range of problems from different areas (energy, finance, production, distribution, supply chain management, etc.). It is well known that an optimization (say, minimization) problem under uncertainty with a finite number of possible supporting scenarios has a Deterministic Equivalent Model (DEM). Traditionally, special attention has been given to optimizing the DEM by minimizing the objective function expected value in the scenarios, subject to the satisfaction of all the constraints, i.e., the so-called Risk Neutral (RN) approach. Note that large DEMs can be solved by using different types of decomposition approaches, e.g., see in [1]. There have been many attempts with two-stage problems in the field of RTND, which are approximations of real problems [11, 17, 26]. Other rail related problems have been also addressed with a two-stage RN approach [9, 15, 16, 28]. Recently, in a series of works [4–6], an alternative approach so-named Service Reliability is introduced for solving large-scale mixed 0-1 models with uncertain passenger demand in RTND.

Let us point out that the optimization of the RN model has the drawback of providing a solution that ignores the potential variability of the objective function value in the scenarios and, so, the occurrence of low-probability high-cost scenarios. Alternatively, risk averse measures could be considered. A computational comparison of some risk averse measures is presented in [2]. Several versions of the multistage mixed 0-1 time-inconsistent risk averse measure based on the Stochastic Dominance (SD) functional introduced in [18] have been presented in [19], and a

ISBN 978-3-89318-079-0 on OpenProceedings, ISSN 2510-7437

DOI: 10.5441/002/inoc.2019.05

time-consistent version of the multistage mixed 0-1 risk averse SD measure is introduced in [21].

For strategic problems (such as RTND), strategic decisions should not depend, even in part, on operational uncertainties in the previous periods. Long-term uncertainty, basically passenger demand and investment costs, should be represented in a multistage scenario tree, where short-term operational uncertainty, basically RTN elements' disruptions, should be represented by considering sub-trees rooted with the strategic nodes. The mixture of those trees may be named as (strategic) multistage (operational) multi-horizon tree. It is worthy to point out that its structure strongly impacts on the model design. Additionally, that type of model should also impact on the decomposition methodologies for problem solving in an affordable effort. Partially due the problem difficulty, there is not a wide literature on the subject. As we know [20, 24, 33] are the first works dealing with multistage multi-horizon trees. A specific application is presented in [33] for a gas transportation network, where the risk averse measure Average Value-at-Risk is considered. A multistage multi-horizon modeling is presented in [3] for an electricity transmission and generation network capacity expansion planning, where the risk averse measure Time Stochastic Dominance is considered. [14] is the first work as we know that addresses the RTND problem as a RN model in a multistage scenario tree by considering dependent stage-wise non-Markovian scenarios with a mixture of the sets of strategic and operational uncertain parameters.

In order to avoid the negative impacts of low probability but high cost or high lost demand scenarios, this work presents a strategic multistage operational multi-horizon 0-1 stochastic risk averse optimization model for the RTND problem. The expected conditional stochastic dominance functional is modeled in two flavors. First, controlling the cost in the strategic scenarios in selected groups and clusters and second, controlling the lost passenger demand in the operational scenarios. Both flavors are time consistent.

This short version of the paper is organized as follows. Section 2 presents the main elements of the scenario tree partitioned in the strategic multistage tree and the operational two-sage trees rooted in nodes in the strategic tree. Section 3 is devoted to the meta model where strategic and operational constraints are considered. Section 4 presents several time-consistent risk averse measures based on the stochastic dominance functional. And, Sections 5 and 6 sketch out the solution approach and computational experiments, respectively.

2 STRATEGIC MULTISTAGE AND OPERATIONAL TWO-STAGE SCENARIO TREES

For completeness let us consider the main elements of the problem inspired in [14, 20]. To represent the uncertainty a scenario analysis approach is used, where the scenario set can be visualized in a tree. Let E be the set of stages along the time horizon, $E = |E|$, T_e be the set of periods (usually, years, semesters) in stage e, for $e \in E$, T be the set of periods in the time horizon, such that $T = \cup_{e \in E} T_e$, $T = |T|$, and Ω be the finite set of representative strategic scenarios. A *scenario* $\omega \in \Omega$ is a particular realization of the uncertain strategic parameters along the time horizon, it is represented in the tree as a root-to-leaf path. A node of the strategic scenario tree represents an event, where it is assumed that the realization of the strategic uncertain parameters and strategic decision variables take place at the first period of the related stage. Notice that the group of scenarios that have the same realization of the uncertain parameters up to any given stage have the same value for the strategic decision variables up to that stage and, thus, the well-known *nonanticipativity* principle is satisfied. Let n and N denote a node and the set of lexicographically numbered nodes $\{1, \ldots, |N|\}$ in the tree, and N_e is the set of nodes that belong to stage e, such that $N = \cup_{e \in E} N_e$. Let also $\Omega^n \subseteq \Omega$ denote the group of scenarios with one-to-one correspondence with node n in the tree. Each node represents a point in time where a strategic decision can be made. Once a decision is made, some contingencies may occur, and information related to those contingencies is available at the beginning of the next stage.

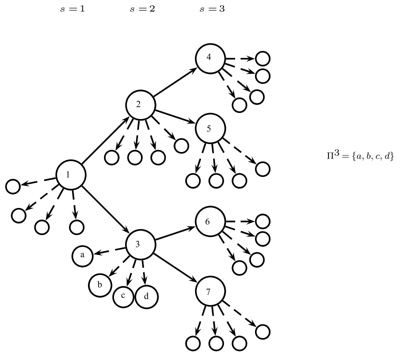


Figure 1: Strategic multistage scenario tree with operational two-stage scenario trees

The additional notation to represent the strategic multistage scenario tree is as follows:

t_e, first period in the lexicographically ordered set T_e in stage e, for $e \in E$.

e^n, stage to which node n belongs to, for $n \in N$.

A^n, set of nodes composed of node n and its ancestors in the tree, for $n \in N$. Note: $A^1 = \{1\}$.

$\tilde{A}^n$, set of nodes composed of node n and its ancestors whose related variables (i.e., representing strategic decisions) have nonzero elements in the constraints in node n, for $n \in N$. Note: $\tilde{A}^n \subseteq A^n$.

S^n, set of successor nodes of node n in the tree, for $n \in N$.

S_1^n, set of immediate successor nodes to node n, for $n \in N_e,\ e \in E$. Note: $S_1^n \subseteq S^n$.

σ^n, immediate ancestor node to node n, thus, $\sigma^n \in A^n$, for $n \in N \setminus \{1\}$.

w^ω, weight or probability assigned to scenario ω, for $\omega \in \Omega$, and $w^n = \sum_{\omega \in \Omega^n} w^\omega$, for $n \in N$.

Now, consider any node $n \in N$ also as a representative of any operational period of stage e^n. The operational uncertainty attached to node n is represented by a finite set of scenarios. They are so-called operational scenarios in a two-stage tree rooted with node n, and the realizations of the scenarios are, precisely, the nodes in the second stage. A 7-node scenario tree is depicted in Fig. 1.

In RTND problems, passenger demand may be considered as the most important uncertain parameter, since its uncertainty is the most independent one of the design of RTN; so, the strategic multistage scenario tree is generated around it. Also assume that the strategic (investment) cost is on some way correlated with the passenger demand. Therefore, passenger demand and investment cost are strategic uncertain parameters defined in strategic nodes while RTN disruptions and operational cost are operational uncertain parameters defined in operational nodes (the ones in the second stage of the two-stage tree rooted with strategic nodes). Thus, in order to have affordable dimensions in the scenario tree from a computational point of view, as an illustration consider $E = 4$ stages, with 5 periods, say years, each one. On the other hand, assume that the number of strategic immediate successor nodes of node n is $|S_1^n| = 3$ for $e^n = 1, 2, 3$ and, so, the cardinality of the scenario tree is $|N| = \sum_{e \in E} |N_e| = 1 + 3 + 9 + 27 = 40$ nodes. Some additional notation related to the RTN infrastructure is as follows:

- I, set of RTN infrastructure elements to be constructed.
- ℓ_i, latency, i.e., number of periods that are required between the period when the construction starts for the RTN infrastructure element i (e.g., an edge as a connection of two stations, a station in the network) and the period at which it becomes available for operation.
- I_i, set of RTN infrastructure elements whose construction cannot start until element i is available (i.e. its construction is over), for $i \in I$. Note: $I_i \subset I$.
- J, set of RTN operational elements.
- I^j, set of RTN infrastructure elements that should be available when operational element j is active, for $j \in J$. Note: $I^j \subset I$.

The notation for the other elements in strategic node n and its operational two-stage scenario tree is as follows, for $n \in N$:

- ι_i^n, strategic ancestor node related to RTN infrastructure element i, such that the period which it belongs to (i.e., t_e where $e \equiv e_{\iota_i^n}$) is the *latest* period *by* which element i can start its construction, so that it is available for use in the RTN at any period in set T_{e^n} for strategic node n, for $n \in N$, $i \in I$:

$$\iota_i^n = argmax_{q \in A^n} \{t_{e^q} \in T : t_{e^q} \leq t_{e^n} - \ell_i\}.$$

- Π^n, set of operational scenarios for the two-stage tree rooted with strategic node n. As an illustrative case, assume $|\Pi^n| = 8$ operational scenarios per each node n in the tree with $|N| = 40$ strategic nodes in the case that have been illustrated above. So, in total, there are 320 uncertain situations to be dealt with, being partitioned in 40 groups. It means that there are 320 RTN operational submodels within the strategic-operational one to be presented next. Many of those submodels will probably have the same or a similar topology.
- w^π, weight or probability of operational scenario π, for $\pi \in \Pi^n$, such that $\sum_{\pi \in \Pi^n} w^\pi = 1$.

3 STRATEGIC MULTISTAGE OPERATIONAL TWO-STAGE STOCHASTIC RISK NEUTRAL 0-1 MODEL

The Risk Neutral (RN) model that is introduced in this section requires the following notation for the variables:

- $(x^n)_i$, 0-1 *step variable* for RTN infrastructure element i in node n, for $i \in I$. Its value is 1 if the element starts its construction *by* period t_{e^n} and otherwise, 0, for $n \in N : t_{e^n} \leq T - \ell_i$, $i \in I$. It is a strategic variable. Let x^n be the $|I|$-dimensional vector of variables $\{(x^n)_i \, \forall i \in I\}$. Notice that $(x^n)_i - (x^{\sigma^n})_i = 1$ means that element i starts is construction *at* node n.
- $(y^\pi)_j$, 0-1 *impulse variable* for RTN operational element j in operational node π, for $\pi \in \Pi^n$, $n \in N$, $j \in J$. Its value is 1 if the element is active *at* operational scenario π in stage e^n to which strategic node n belongs to and otherwise, 0. It is an operational variable. Let y^π be the $|J|$-dimensional vector of variables $\{(y^\pi)_j \, \forall j \in J\}$.

Note: It is well-known that the modeling scheme where the step x-variables are considered is stronger than the model where they are replaced with impulse variables.

The parameters are as follows:

- $(a^n)_i$, objective function coefficient (i.e., investment cost) related to the RTN infrastructure element i if it starts its construction *at* node n, for $n \in N : t_{e^n} \leq T - \ell_i$, $i \in I$. It is assumed that the construction cost is made at the starting period t_{e^n}. Note: That assumption can be easily replaced with an ad-hoc policy.
- b^π, vector of the objective function coefficients (e.g., passenger demand lost, among others) of the operational variables in vector y^π, for $\pi \in \Pi^n$, $n \in N$.
- h_s^n, rhs of the set of constraints related to strategic node n, for $n \in N$.
- A_n^q, constraint matrix for the variables in vector x^q of ancestor node q in the strategic constraints related to node n, for $q \in \tilde{A}^n$, $n \in N$.
- h_o^π, rhs of the set of constraints related to operational scenario π, for $\pi \in \Pi^n$, $n \in N$.
- B^π, constraint matrix for the variables in vector y^π, for $\pi \in \Pi^n$, $n \in N$.
- k, interest rate by period.

The DEM RN 0-1 model can be expressed as follows:

$$\begin{aligned} z = \min \sum_{i \in I} \sum_{\substack{n \in N: \\ t_{e^n} \leq T - \ell_i}} \frac{1}{(1+k)^{t_{e^n}}} w^n (a^n)_i ((x^n)_i - (x^{\sigma^n})_i) + \\ \sum_{n \in N} \frac{1}{(1+k)^{t_{e^n}}} w^n |T_{e^n}| \sum_{\pi \in \Pi^n} w^\pi b^\pi y^\pi, \end{aligned} \tag{1}$$

subject to

$$\begin{array}{ll} \sum_{q \in \tilde{A}^n} A_n^q x^q = h_s^n & \forall n \in N \\ (x^{\sigma^n})_i \leq (x^n)_i & \forall n \in N : t_{e^n} \leq T - \ell_i, \, i \in I \\ (x^n)_{i'} - (x^{\sigma^n})_{i'} \leq (x^{\iota_i^n})_i & \forall n \in N : t_{e^n} \leq T - \ell_i, \, i' \in I_i, \, i \in I \\ (y^\pi)_j \leq (x^{\iota_i^n})_i & \forall \pi \in \Pi^n, \, n \in N, \, i \in I^j, \, j \in J \\ B^\pi y^\pi = h_o^\pi & \forall \pi \in \Pi^n, \, n \in N \\ (x^n)_i \in \{0, 1\} & \forall n \in N : t_{e^n} \leq T - \ell_i, \, i \in I \\ (y^\pi)_j = 0 & \forall \pi \in \Pi^n, \, n \in N, \, j \in J \\ y^\pi \in \{0, 1\} & \forall \pi \in \Pi^n, \, n \in N, \, j \in J. \end{array} \tag{2}$$

4 RISK AVERSE EXPECTED CONDITIONAL STOCHASTIC DOMINANCE FUNCTIONALS

There are some risk averse approaches that deal with risk management [31], see a computational comparison in [2]. Among them, the Stochastic Dominance (SD)-based measures reduce the risk of the negative impact of the solutions in non-wanted scenarios in a better way than the others under some circumstances. See in [18, 32] its theoretical foundations, among others.

In a rapid transit network, the variability in cost along the time horizon in the strategic scenarios (where the operational scenarios are considered) and the variability in lost passenger demand in the operational scenarios for the related strategic node in the stages could be very high. In order to avoid the negative impact of the solution in the scenarios, mainly those with low probability and high cost or high lost demand, some risk reduction measures should be considered.

Time-consistency

Roughly, a risk-averse measure is time-consistent if the solution to be obtained from the submodel supported by a subtree rooted with a node at a given stage in a multistage scenario tree is the same as the one for that node and successors in the model supported by the full multistage scenario tree.

The rationale behind a time-consistent risk measure is that the solution value to be obtained in any node n and its successors for the related submodel "'solved"' at stage e^n should have the same value as in the original model "'solved"' at stage $e = 1$. Obviously, the RN model given by (1) and (2) and the model given by (1), (2) and (5) supported by the operational two-stage trees rooted at the strategic nodes are time-consistent. Additionally, it is not difficult to prove that the model given (1),(2),(3) and (5) is also time-consistent; in another context, see [21].

Section 4.1 presents the expected conditional stochastic dominance (ECSD) version for controlling the objective function value (i.e., the overall strategic-operational cost) in the *scenario groups* for modeler-driven subset of stages. Section 4.2 presents a stochastic dominance (SD) risk averse functional for controlling the objective function value (i.e., the overall strategic-operational cost) in a modeler-driven set of *scenario clusters*. The conditions to be satisfied by the SD functional in order to have the time-consistency property are also given. And Section 4.3 presents the ECSD version for controlling the lost passenger demand in the operational scenarios.

4.1 Objective function excess risk reduction for strategic scenario groups

The risk averse measure ECSD for the Net Present Value (NPV) of the expected objective function value composed of the expected investment cost on stations and edges of the new network (for short, expected strategic cost) and the expected operational cost of the available infrastructure elements of the new network for each strategic node in the whole time horizon requires the following additional sets of modeler-driven scenario groups and profiles:

E^{St}, subset of stages in set E, whose scenario groups with one-to-one correspondence with strategic nodes (including the related operational ones) are to be considered.

P^n, set of profiles for scenario group Ω^n, for $n \in N_e,\ e \in E^{St}$.

For each profile $p \in P^n$, let the following modeler-driven parameters:

ϕ^p, objective function (i.e., cost) threshold in the whole time horizon to consider for any scenario in group Ω^n (i.e., group with one-to-one correspondence with strategic node n), where the operational scenarios in set Π^n are taken into account.

$\tilde{s}^p$, upper bound of the expected cost excess over threshold ϕ^p for any scenario ω in group Ω^n.

$\bar{s}^p$, upper bound of the expected cost excess over threshold ϕ^p in group Ω^n as a whole.

The profile contents are inspired in the second-order stochastic dominance functional induced by integer-linear recourse for multistage stochastic problems, see its time-consistent version in [21].

The variable for pair (ω, p), where ω is a strategic scenario in group Ω^n and p is the index of profile in P^n is as follows:

$s^{\omega,p}$, continuous variable that takes the expected cost excess over threshold ϕ^p in strategic scenario ω in group Ω^n, where the operational scenarios in set Π^n are taken into account.

The objective function (i.e., overall strategic-operational cost) risk reduction ECSD constraint system can be expressed as:

$$\begin{aligned}
&\sum_{i\in I}\ \sum_{\substack{q\in A^{\omega}:\\ t_{e^q}\leq T-\ell_i}} \frac{1}{(1+k)^{t_{e^q}}}(a^q)_i((x^q)_i-(x^{\sigma^q})_i)+\\
&\sum_{q\in A^{\omega}} \frac{1}{(1+k)^{t_{e^q}}}|T_{e^q}| \sum_{\pi\in\Pi^q} w^{\pi} b^{\pi} y^{\pi} - s^{\omega,p} \leq \phi^p \text{ and}\\
&0 \leq s^{\omega,p} \leq \tilde{s}^p \quad \forall \omega \in \Omega^n,\ p \in P^n,\ n \in N_e,\ e \in E^{St}\\
&\sum_{\omega\in\Omega^n} (w^{\omega}/w^n)s^{\omega,p} \leq \bar{s}^p \quad \forall p \in P^n,\ n \in N_e,\ e \in E^{St}.
\end{aligned} \tag{3}$$

Notice that the key element in constraint system (3) is that those scenario-cross constraints are related to scenarios that belong to the same group at any stage in set E^{St}.

4.2 Objective function excess risk reduction for strategic scenario clusters

A risk reduction functional for the objective function value in scenario clusters is presented in this section with a similar scheme as the one presented in the previous section for the scenario groups with one-to-one correspondence with a modeler-driven stage subset. So, it has similar notation for the risk reduction profiles. The difference between both functionals is that, now, the strategic scenarios to consider are clustered according to a modeler-driven criterion. So, let C denote the set of scenario clusters, and Ω_c is the set of strategic scenarios in the cluster indexed with c, for $c \in C$. There is a high flexibility on the structuring of the clusters, i.e., (a) a scenario could belong to more than one cluster, and (b) more than one cluster may have one-to-one correspondence with the same strategic node.

Let P_c denote the set of profiles for scenario cluster Ω_c, for $c \in C$. So, for each profile $p \in P_c$, let the following modeler-driven parameters:

ϕ^p, Objective function (i.e., cost) threshold in the whole time horizon to consider for any scenario in cluster Ω_c, where scenario ω, for $\omega \in \Omega_c$ includes the strategic nodes in it ancestor path down to the root node in the strategic multistage tree, A^{ω}, so that the operational scenarios in set Π^q are taken into account, for $q \in A^{\omega}$.

$\tilde{s}^p$, upper bound of the expected cost excess over threshold ϕ^p for any scenario ω in cluster Ω_c.

$\bar{s}^p$, upper bound of the expected cost excess over threshold ϕ^p in cluster Ω_c as a whole.

The variable for pair (ω, p), where ω is a strategic scenario in cluster Ω_c and p is the index of profile in P_c is as follows:

$s^{\omega,p}$, continuous variable that takes the expected cost excess over threshold ϕ^p in strategic scenario ω in cluster Ω^c, where the operational scenarios in set Π^q are taken into account.

The risk reduction stochastic dominance constraint system related to the objective function (i.e., overall strategic-operational cost) can be expressed as:

$$
\begin{aligned}
&\sum_{i \in I} \sum_{\substack{q \in A^\omega: \\ t_{e^q} \leq T-\ell_i}} \frac{1}{(1+k)^{t_{e^q}}} (a^q)_i((x^q)_i - (x^{\sigma^q})_i) + \\
&\sum_{q \in A^\omega} \frac{1}{(1+k)^{t_{e^q}}} |T_{e^q}| \sum_{\pi \in \Pi^q} w^\pi b^\pi y^\pi - s^{\omega,p} \leq \phi^p \text{ and} \\
&0 \leq s^{\omega,p} \leq \tilde{s}^p \quad \forall \omega \in \Omega_c,\, p \in P_c,\, c \in C \\
&\sum_{\omega \in \Omega_c} (w^\omega / w_c) s^{\omega,p} \leq \bar{s}^p \quad \forall p \in P_c,\, c \in C,
\end{aligned}
\tag{4}
$$

where $w_c = \sum_{\omega \in \Omega_c} w^\omega$ for $c \in C$.

Notice that the key element in constraint system (4) is similar to the one in system (3), here, those scenario-cross constraints are related to scenarios that belong to the same cluster.

It is worth pointing out that the risk reduction functional given in system (4) is a time-consistent one, provided that the following conditions are satisfied:

(1) The scenarios do not overlap in the clusters, i.e., $\Omega_c \cap \Omega_{c'} = \emptyset$ for any pair $c, c' \in C : c \neq c'$.
(2) Each scenario cluster Ω_c for $c \in C$ is included in some scenario strategic group Ω^n, i.e., $\exists n \in N_e : e \in E$ such that $\Omega_c \subseteq \Omega^n$, $c \in C$.

It is also worth pointing out that the time-consistency of the functional does not prevent that any scenario group is partitioned in several scenario clusters.

4.3 Risk reduction for the lost passenger demand at selected strategic nodes

The risk averse measure ECSD is specialized in this section for risk reduction in the operational scenario set Π^n, $n \in N$. Here, the function to consider is (the operational one related to) the passenger demand lost to the current transport system in the operational scenarios in the strategic nodes of a subset os stages.

The operational-based ECSD requires the following additional sets and elements for modeler-driven strategic nodes:

E^{Op}, subset of stages in set E, whose passenger demand lost is to be reduced. Note: $E^{St} \cap E^{Op}$ could be an empty set.

W, passenger groups defined by origin/destination (o/d) pairs.

g_w^π, number of passengers in group w, for $w \in W$. Notice that its demand could be lost; it is a parameter that belongs to the objective function operational vector b^π.

f_w^π, a 0-1 variable, such that its value 1 means that passenger group w is lost to the current network in operational scenario π and otherwise, 0, for $w \in W$, $\pi \in \Pi^n$, $n \in N$. Note: Variable f_w^π belongs to operational variables vector y^π.

P^n, set of profiles that are associated with operational scenario set Π^n, for $n \in N^e$, $e \in E^{Op}$, instead of been associated with strategic scenario group Ω^n as it is presented in Section 4.1.

For each profile $p \in P^n$, the following parameters are required:

γ^p, passenger demand lost threshold to consider in any operational scenario π, for $\pi \in \Pi^n$.

$\tilde{s}^p$, upper bound of the demand lost excess over threshold γ^p in any operational node π, for $\pi \in \Pi^n$.

$\bar{s}^p$, upper bound of the expected demand lost excess over threshold γ^p in set Π^n.

The variable for pair (π, p), where π is an operational node and p is the index of a profile in strategic node n, for $p \in P^n$, $\pi \in \Pi^n$, $n \in N^e$, is as follows for stage e, for $e \in E^{Op}$:

$s^{\pi,p}$, continuous variable that takes the passenger demand lost over threshold γ^p in operational scenario π, for $\pi \in \Pi^n$.

The risk reduction ECSD constraint system related to the (operational) passenger demand lost can be expressed

$$
\begin{aligned}
&\frac{1}{(1+k)^{t_e}} |T_e| \sum_{w \in W} g_w^\pi f_w^\pi - s^{\pi,p} \leq \gamma^p \text{ and} \\
&0 \leq s^{\pi,p} \leq \tilde{s}^p \quad \forall \pi \in \Pi^n,\, p \in P^n,\, n \in N^e,\, e \in E^{Op} \\
&\sum_{\pi \in \Pi^n} w^\pi s^{\pi,p} \leq \bar{s}^p \quad \forall p \in P^n,\, n \in N^e,\, e \in E^{Op}.
\end{aligned}
\tag{5}
$$

So, the ECSD model that is proposed in this work can be expressed as the expected cost (1) to minimize, subject to the strategic node-based constraint system (2), the one for linking strategic and operational variables and the operational node-based constraint system, plus the cross strategic scenario group and operational set based constraint systems (3) and (5), respectively.

5 SOLUTION APPROACH

Given the problem's complexity and the huge model's dimensions (due to the RTND static model as well as the potentially high number of scenario nodes in the multistage setting), it is unrealistic to seek for an optimal solution, even by considering decomposition approaches for problem solving. So, a decomposition approach is required for obtaining a (hopefully, good) feasible solution where its optimality gap is guaranteed.

A version of the matheuristic FLAggA (that stands for Fix-and-Lazy Aggregated / de-aggregated Algorithm) [14] is presented in the full paper to deal with the risk averse measures represented in the constraint systems (3), (4) and (5).

6 COMPUTATIONAL EXPERIMENT

This section introduces the computational experiment, whose results are not detailed due to space limitations. It is based on the RTN so-called R1, see [29], which has also been used in [10, 11, 17, 25], among others. It features 9 nodes, 15 edges and 72 passenger groups. All previous efforts for problem solving have been devoted either to the deterministic or the RN version of the network.

A broad computational study is performed to compare the performance of FLAggA and the plain use of a state-of-the-art solver on one hand. And on the other one, a computational analysis is carried out by comparing the RN version of the model with the proposed risk averse measures.

Two different scenario trees are considered in the experiment, namely a proof-of-concept tree and a tree with more realistic dimensions. The first tree features 3 stages, 7 strategic nodes, 56 operational scenarios and 4 strategic scenarios. The second tree features 4 stages, 40 strategic nodes, 320 operational scenarios and 27 strategic scenarios.

The RN solution provides a high variability in many issues, as for example in the lost passenger demand. For the 40-node scenario tree case, for illustrative purposes, the differences between the strategic scenarios is shown in Figure 2. The upper

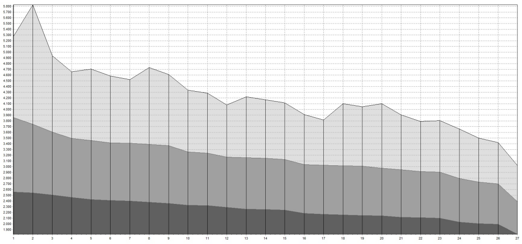


Figure 2: Scenario demand and lost demand

curve in the figure depicts the passenger demand for each of the strategic scenarios, while the middle and lower curves depict the expected lost of passenger demand for two latency strategies as provided by the incumbent solution obtained by the matheuristic algorithm FLAggA.

Table 1 shows some statistics for the demand and lost demand in the scenarios. The headings are related to the largest, smallest, average and its standard deviation. The purpose of the proposed risk averse functionals is to reduce this level of lost demand with the same allowed budget for infrastructure investment.

Table 1: Passenger demand statistics for the 27 strategic scenarios

Demand	*largest*	*smallest*	*aver*	*dev*
Scenario-based	5828.52	3025.38	4229.25	577.64
Lost for $\ell = 1$	3858.39	2391.86	3160.52	325.99
Lost for $\ell = 0$	2560.68	1823.13	3221.41	179.47

ACKNOWLEDGMENTS

The work is supported by the Spanish Ministry of Economy and Competitiveness under Grants numbered TRA2016-76914-C3-3-P and MTM2015-63710-P.

REFERENCES

[1] U. Aldasoro, L.F. Escudero, M. Merino, J.F. Monge and G. Pérez. A parallel Branch-and-Fix Coordination based matheuristic algorithm for solving large sized multistage stochastic mixed 0-1 problems. *European Journal of Operational Research*, 258:590-606, 2017.

[2] A. Alonso-Ayuso, F. Carvallo, L.F. Escudero, M. Guignard, J. Pi, R. Puranmalka and A. Weintraub. On the optimization of copper extraction in mining under uncertainty in copper prices. *European Journal of Operational Research*, 233:711-726, 2014.

[3] A. Alonso-Ayuso, L.F. Escudero, F.J. Martín-Campo and J.F. Monge. On the strategic multistage scenario tree, tactical multi-horizon graphs in electricity Transmission / Generation network capacity Expansion Planning, TGEP. EC COST TD1207-2017, Final Conference, Modena (Italy), 2017. http://www.cost-td1207conference.unimore.it/ and http://cost-td1207.zib.de/

[4] K. An and H.K. Lo. Service Reliability-based transit network design with stochastic demand, *Transportation Research Record*, 2467:101-109, 2014.

[5] K. An and H.K. Lo. A robust transit network design with stochastic demand considering development density. *Transportation Research Part B*, 81:737-754, 2015.

[6] K. An and H.K. Lo. Two-phase stochastic program for transit network design under demand uncertainty. *Transportation Research Part B*, 81:157-181, 2016.

[7] A. Bärmann, A. Martin and H. Schülldorf. A decomposition method for multiperiod railway network expansion, with a case study for Germany. *Transportation Science*, 51:1102âĂŞ1121, 2017.

[8] D. Bertsimas and M. Slim. The price of robustness. *Operations Research*, 52:35-53, 2004.

[9] V. Cacchiani, A. Caprara, L. Galli, L. Kroon, G. Maróti and P. Toth. Railway rolling stock planning: Robustness against large disruptions. *Transportation Science*, 46:217-232, 2012.

[10] L. Cadarso and A. Marín. Improved rapid transit network design model: considering transfer effects. *Annals of Operations Research*, :1-21, 2015. doi:10.1007/s10479-015-1999-x, 2015.

[11] L. Cadarso and A. Marín. Combining robustness and recovery in rapid transit network design. *Transportmetrica A: Transport Science*, 12:203-229, 2016.

[12] L. Cadarso, A. Marín and G. Maroti. Recovery of disruptions in rapid transit networks. *Transportation Research Part E: Logistics and Transportation Review*, 53:15-33, 2013.

[13] L. Cadarso, G. Maroti and A. Marín. Smooth and controlled recovery planning of disruptions in rapid transit networks. *IEEE Transactions on Intelligent Transportation Systems*, 16:2192-2202, 2015.

[14] Cadarso, L., Escudero, L. F., and Marín, A. On strategic multistage operational two-stage stochastic 0-1 optimization for the Rapid Transit Network Design problem. *European Journal of Operational Research*, 271(2):577-593 , 2018

[15] A. Caprara, L. Galli, S. Stiller and P. Toth. Recoverable-robust plataforming by network buffering. *Technical Report ARRIVAL-TR.0157*. ARRIVAL project, http://www.cti.gr/arrival, 2008.

[16] S. Cicerone, G. DâĂŹAngelo, G. Di Stefano, D. Frigioni, A. Navarra, M. Schachtebeck and A. Schöbel. Recoverable robustness in shunting and timetabling. In R. Ahuja, R. MÃűhring and C. Zaroliagis (eds.). *In Robust and On-Line Large Scale Optimization*, pp. 28-60. Springer, 2009.

[17] E. Codina, A. Marín and L. Cadarso. Robust Infrastructure Design in Rapid Transit Rail systems. *Transportation Research Procedia*, 3:660-669, 2014.

[18] D. Dentcheva and A. Ruszczynski. Optimization with stochastic dominance constraints. *SIAM Journal on Optimization*, 14:548-566, 2003.

[19] L.F. Escudero, A. Garín and A. Unzueta. Cluster Lagrangean decomposition for risk averse in multistage stochastic optimization. *Computers and Operations Research*, 85:154-171, 2017.

[20] L.F. Escudero and J.F. Monge. On capacity expansion planning under strategic and operational uncertainties based on stochastic dominance risk averse management. *Computational Management Science*, 15:479-500, 2018.

[21] L.F. Escudero, J.F. Monge and D. Romero-Morales. On time-consistent stochastic dominance risk averse measure for tactical supply chain planning under uncertainty. *Computers and Operations Research*, doi.org/10.1016/j.cor.2017.07.011, 2017.

[22] L.F. Escudero and S. Muñoz. An approach for solving a modification of the extended rapid transit network design problem. *TOP*, 17:320âĂŞ334, 2009

[23] M. Fischetti, D. Salvagnin and A. Zanette. Fast approaches to improve the robustness of a railway timetable. *Transportation Science*, 43:321-335, 2009.

[24] M. Kaut, K.T. Midthun, A.S. Werner, A. Tomasgard, L. Hellemo and M. Fodstad. Dual-level scenario trees âĂŞ scenario generation and applications in energy planning. *Computational Management Science*, 11:179-193, 2014.

[25] G. Laporte, A. Marín, J.A. Mesa and F.A. Ortega. An integrated methodology for the rapid transit network design problem. In F. Geraets, L. Kroon, A. Schoebel, D. Wagner and C.D. Zaroliagis (eds.). *Algorithmic Methods for Railway Optimization*, pp. 187-199. Springer, 2007.

[26] G. Laporte, A. Marín, J.A. Mesa and F. Perea. Designing robust rapid transit networks with alternative routes. *Journal of advanced transportation*, 45:54-65, 2011.

[27] G. Laporte and J.A. Mesa. The design of rapid transit networks. In: Laporte G., Nickel S. and Saldanha da Gama F. (eds). *Location Science*, pp. 581-594. Springer, 2015.

[28] C. Liebchen, M. LÃijbbecke, R. Möhring and S. Stiller. The Concept of recoverable robustness, linear programming recovery and railway applications. In R. Ahuja, R. Möhring and C. Zaroliagis (eds). *In Robust and Online Large-Scale Optimization*, pp. 1âĂŞ27. Springer, 2009.

[29] A. Marín. An extension to rapid transit network design problem. *TOP*, 15:231-241, 2007.

[30] A. Marín and P. Jaramillo. Urban rapid transit network capacity expansion. *European Journal of Operational Research*, 191:45-60, 2008.

[31] G.Ch. Pflug and W. Römisch. *Modeling, measuring and managing risk.* World Scientific, 2007.

[32] A. Shapiro and A. Pichler. Time and Dynamic Consistency of Risk Averse Stochastic Programs. *Optimization Online* http://www.optimization-online.org/DB HTML/2016/09/5654, 2016.

[33] A.S. Werner, A. Pichler, K.T. Midthun, L. Hellemo and A. Tomasgard. Risk measures in multihorizon scenarios tree. In R. Kovacevic, G.Ch. Pflug and M.T. Vespucci (eds.). *Handbook of Risk Management in Energy Production and Trading*, Springer, 177-201, 2013.

Formulation and Branch-and-cut algorithm for the Minimum Cardinality Balanced and Connected Clustering Problem

Alexandre Salles da Cunha*
Departamento de Ciência da Computação, Universidade Federal de Minas Gerais
Belo Horizonte, Brazil
acunha@dcc.ufmg.br

ABSTRACT

Given a graph $G = (V, E)$, integer bounds $l, u : 0 \leq l \leq u$ and positive integer weights assigned to the vertices V of G, the Minimum Cardinality Balanced and Connected Clustering Problem (MCBCCP) consists of finding a minimum cardinality partitioning of the vertices of V so that (a) each set in the partition induces a connected subgraph of G and (b) the sum of the weights of the vertices in each set belongs to the interval $[l, u]$. In this paper, we present an integer programming formulation, valid inequalities and a Branch-and-cut algorithm for MCBCCP. Our computational experiments suggest that the inequalities investigated here help the overall performance of the algorithm. In addition to the one tested here, we discuss other formulations, following different modeling arguments. Some of them are based on MCBCCP's connections to other combinatorial optimization problems found in the literature.

KEYWORDS

Combinatorial Optimization, Clustering, Spanning forests, Branch-and-cut algorithms

1 INTRODUCTION

A K partition of the vertex set of an undirected graph $G = (V, E)$ ($n = |V|, m = |E|$) is a collection $\{V_1, \ldots, V_K\}$ of K non-empty pairwise disjoint subsets of V such that $\cup_{j=1}^{K} V_j = V$. The vertices in the same partition define a cluster. Assume that positive integer weights $\{w_i \in \mathbb{Z} : i \in V\}$ are assigned to the vertices of G and that integer bounds $l, u : 0 \leq l \leq u$ are given. Given $S \subseteq V$, define $w(S)$ as $\sum_{i \in S} w_i$. If the conditions

$$l \leq w(V_j) \leq u, \qquad j = 1, \ldots, K, \tag{1}$$

are satisfied, the clusters are balanced. Define $E(S) = \{\{i, j\} \in E : i, j \in S\}$ as the edges with both endpoints in S and denote by C the collection of all connected subgraphs of G, including those with just a single vertex and no edge incident to it. If $\{(V_j, E(V_j)) \in C : j = 1, \ldots, K\}$, the clustering is connected. If $l \leq w(V_j) \leq u$ and $(S, E(V_j)) \in C$ for every $j = 1, \ldots, K$ the clustering is balanced and connected. Accordingly, each cluster V_j is balanced and connected.

The Minimum Cardinality Balanced and Connected Clustering Problem (MCBCCP) consists of finding a balanced and connected clustering of G of minimum cardinality. MCBCCP is an NP-Hard optimization problem, even for series parallel graphs, but is solvable in linear time in case G is a path [13].

Our goal is to solve MCBCCP when G does not belong to a particular graph class. To that aim, we introduce an integer programming formulation, valid inequalities and an exact solution approach, of the Branch-and-cut type [21]. These topics are addressed in Sections 2 and 3. Our preliminary computational experiments reported in Section 4 suggest that the valid inequalities discussed here were of help to enhance the overall performance of the Branch-and-cut algorithm. In addition to that, we describe, at the last section of this paper, several other modeling strategies that may lead to effective MCBCCP exact algorithms. We also highlight connections between MCBCCP and some network design problems, that may be explored from a modeling and algorithmic perspective.

In the remaining of this section, we review some clustering problems, concentrating mostly on those that closely relate to MCBCCP, and thus, require clusters to be either balanced or to satisfy some kind of connectivity constraint.

Graph clustering is a central problem in Operations Research, Computer Science and Artificial Intelligence. They arise in applications as diverse as circuit board and micro-chip design, parallel computation, sparse matrix factorization and data mining [12]. Depending on the objective function and on the similarity criteria used to group vertices, different clustering problems arise. Not surprisingly, the literature on the topic is vast; a review of several clustering problems can be found, for instance, in [25].

Ito et al. [13] discussed three problems related to optimal choosing connected and balanced clusters. One of them consists of deciding whether or not G has a balanced and connected clustering of fixed size p. The other two are MCBCCP and the problem that maximizes the number of clusters. All of them arise in practical applications such as political districting, paging systems of operation systems and image processing. Ito et al. [13] was concerned with MCBCCPs defined on trees. To that particular input graph class, the authors presented the first polynomial time algorithm.

A common sense in graph clustering is that similar vertices should be grouped together, while dissimilar ones should be separated. Quite often, the similarity of a pair of vertices i, j is *measured* by a weight c_{ij}. Therefore, an objective function that arises frequently in practice, specially for cardinality constrained clustering problems, is the minimization of the sum of the weights of the edges connecting vertices in different clusters [1, 8, 11, 12, 15, 23, 24, 27]. Since $\sum_{\{i,j\} \in E} c_{ij}$ is a constant, an equivalent problem consists of maximizing the sum of the weights of the edges connecting vertices in the same clusters. In general, graph clustering is NP-Hard [9].

The problem of finding optimal balanced clusterings of fixed cardinality has received substantial attention in the literature. Jonhson et al. [14] investigated one such problem. In addition to integer bounds l, u and weights $\{w_i : i \in V\}$, costs $\{c_{ij} : \{i, j\} \in E\}$ are also assigned to the edges of E. The problem thus consists of finding a K balanced clustering of G that minimizes the function $\sum_{k=1}^{K} \sum_{\{i,j\} \in E(V_k)} c_{ij}$. In that particular problem, input graphs are not complete and clusters do not need to induce connected subgraphs of G.

*This research is partially funded by CNPq grants 303928/2018-2, 431369/2016-0 and FAPEMIG grants CEX-PPM-00164/17, APQ-02645-16.

ISBN 978-3-89318-079-0 on OpenProceedings, ISSN 2510-7437 DOI: 10.5441/002/inoc.2019.06

Clustering under some sort of connectivity requirements has also been a topic of interest [3, 5, 7, 17]. In Lari et al. [17], the number of clusters K is defined beforehand. There is a subset S ($K = |S|$) of V that plays the role of clusters heads. The remaining vertices, those in the set $U = V \setminus S$, are denoted unit vertices. The problem asks for a K-centered connected partition of G: a partition of the vertices of G into K connected subgraphs $\{(V_j, E(V_j)) : j = 1, \dots, K\}$, such that $|V_j \cap S| = 1$ for each $j = 1, \dots, K$. One cluster head must be assigned to each cluster and unit vertices must be assigned to one vertex in S, the head of one cluster. The cost of assigning a unit vertex $u \in U$ to a cluster head $i \in S$ is denoted c_{ui}. The cost of a component (or cluster) V_j is $\sum_{i \in S \cap V_j} \sum_{u \in V_j \cap U} c_{ui}$. Additionally, weights $\{w_v : v \in V\}$ are assigned to each vertex of G. Lari et al. [17] investigated the problem of finding K centered connected partitions of G for different min-max objective functions involving either the assignment costs or the vertices' weights. The complexity class of each of these variants was also investigated.

Brucker [3] investigated the problem of clustering the vertices of V in K or fewer sets such that each vertex set induces a subgraph with diameter at most p; the diameter being the longest minimum shortest path between two vertices in the subgraph. The problem was proven to be NP-Complete and remains so even if $K = 3$ and all edges' lengths are taken from the set $\{0, 1\}$. Deogun et al. [6] investigated the same K clustering problem under binary edge costs. They introduced approximation results and algorithms, specialized for certain classes of input graphs.

Edachery et al. [7] introduced the Partition into Distance $p-$cliques Problem, another optimization problem whose decision version was proven to be NP-Complete. A vertex set $S \subseteq V$ is a distance $p-$clique if, for any pair of vertices $i, j \in S$, there is a path in the graph $(S, E(S))$ that involves at most p edges (or hops). The Partition into Distance $p-$cliques Problem thus consists of partitioning the vertex set of V in to the least number of distance $p-$cliques. For solving the problem, Edachery et al. [7] introduced heuristics and addressed their performance on various large scale graphs found in the telecommunication industry.

Nossack and Pesch [20] investigated the acyclic clustering problem, a problem defined in terms of a vertex and arc weighted directed graph $D = (V, A)$. A feasible solution to the problem is a K balanced clustering, with one additional property: the graph obtained by shrinking each cluster into a single vertex and by merging arcs that start and end at the same pair of inbound and outbound clusters must be a directed acyclic graph. The objective function consists of maximizing the sum of the weights of the arcs with both endpoints in the same cluster.

Finally, Aragão and Uchoa [5] investigated the γ-Connected Assignment Problem ($\gamma-$CAP). Given $G = (V, E)$, a set of colors $K = \{1, \dots, k\}$, a vector $\gamma = (\gamma_1, \dots, \gamma_k)$ of positive integers and costs $\{c_{iq} : i \in V, q \in K\}$, the γ-Connected Assignment Problem consists of finding a minimum cost assignment of colors to the vertices in such a way that no set of vertices assigned to the same color q induces a subgraph of G with more than γ_q connected components. Applications of the $\gamma-$CAP can be seen as variants of the Contiguity Constrained Clustering Problem [18, 19].

2 MCBCCP FORMULATION

Among other decision variables to be defined shortly, the model uses an integer variable K to denote the cardinality of the clustering we are looking for. Assuming that $l > 0$ holds, feasible values for K must satisfy $\left\lceil \frac{n}{\lfloor \frac{u}{\min\{w_i : i \in V\}} \rfloor} \right\rceil \leq K \leq \left\lfloor \frac{n}{\lceil \frac{l}{\max\{w_i : i \in V\}} \rceil} \right\rfloor$. Let $\overline{K}$ denote an upper bound on the maximum number of balanced clusters of G. If $l > 0$, $\overline{K}$ can be taken as the upper bound in the previous expression. If $l = 0$, $\overline{K} = n$.

The idea behind the formulation is to find a spanning forest of G with precisely $n - K$ edges, such that the vertices in each of its K trees define balanced clusters. Define $K_i = \min\{i, \overline{K}\}$. For a given $i \in V$, the set $\{1, \dots, K_i\}$ gives the indexes of connected components where i can be placed in. In addition to K, the formulation uses the following decision variables:

- $\mathbf{x} = \{x_{ij} \in \mathbb{B} : \{i, j\} \in E\}$. Variable x_{ij} assumes value 1 if edge $\{i, j\}$ belongs to the forest and 0 if otherwise applies. For any subset $E' \subseteq E$, define $x(E') = \sum_{\{i,j\} \in E'} x_{ij}$.
- $\mathbf{y} = \{y_i^k \in \mathbb{B} : i \in V, k = 1, \dots K_i\}$. Variable y_i^k assumes value 1 if vertex i belongs to the $k-$th connected component of the forest.
- $\mathbf{z} = \{z^k \in \mathbb{B} : k = 1, \dots, \overline{K}\}$. Variable z^k assumes value 1 if and only if the forest has k or more connected components.

The formulation is:

$$\min \left\{ K : (K, \mathbf{x}, \mathbf{y}, \mathbf{z}) \in \mathcal{P} \cap (\mathbb{R} \times \mathbb{B}^m \times \mathbb{B}^\sigma \times \mathbb{B}^{\overline{K}}) \right\}, \qquad (2)$$

where $\sigma := \sum_{i=1}^n K_i$ and $\mathcal{P}$ is the polyhedral region defined by:

$$x(E) + K = n \qquad (3)$$

$$x(E(S)) \leq |S| - 1 \qquad S \subset V, |S| \geq 2 \qquad (4)$$

$$\mathbf{x} \geq 0 \qquad (5)$$

$$\sum_{k=1}^{K_i} y_i^k = 1 \qquad i \in V \qquad (6)$$

$$\sum_{i=k}^{n} y_i^k \geq z^k \qquad k = 1, \dots, \overline{K} \qquad (7)$$

$$\sum_{i=k}^{n} w_i y_i^k \leq u z^k \qquad k = 1, \dots, \overline{K} \qquad (8)$$

$$\sum_{i=k}^{n} w_i y_i^k \geq l z^k \qquad k = 1, \dots, \overline{K} \qquad (9)$$

$$x_{ij} + y_i^\tau + \sum_{k=1}^{\tau-1} y_j^k + \sum_{k=\tau+1}^{K_j} y_j^k \leq 2 \qquad \begin{smallmatrix}\{i,j\} \in E, i<j \\ \tau=1,\dots,K_i\end{smallmatrix} \qquad (10)$$

$$\sum_{k=1}^{K_i} k y_i^k - \sum_{k=1}^{K_j} k y_j^k \geq -M_{ij}(1 - x_{ij}) \qquad \{i, j\} \in E \qquad (11)$$

$$\sum_{k=1}^{K_i} k y_i^k - \sum_{k=1}^{K_j} k y_j^k \leq M_{ij}(1 - x_{ij}) \qquad \{i, j\} \in E \qquad (12)$$

$$\sum_{k=1}^{\overline{K}} z^k = K \qquad (13)$$

$$z^k \leq z^{k-1} \qquad k = 2, \dots, \overline{K} \qquad (14)$$

$$y_i^k \leq z^k \qquad \begin{smallmatrix} i \in V \\ k=1,\dots,K_i \end{smallmatrix} \qquad (15)$$

$$y_i^k \geq 0 \qquad \begin{smallmatrix} i \in V \\ k=1,\dots,K_i \end{smallmatrix} \qquad (16)$$

Aiming to cope with formulation symmetries, variables y_i^k for $k > K_i$ are not used; only y_i^k for $k = 1, \dots K_i$ are in place.

Constraints (3)-(5) model the spanning forest with $n - K$ edges of G. Subtour elimination constraints (SECs) (4) avoid that K is decreased to an infeasible value, by selecting edges in excess of what an acyclic subgraph $(S, E(S))$ of G allows.

Constraints (6) enforce that each vertex must be assigned to one connected component of G. These constraints make a clear distinction of the indexes of components that can be assigned to the vertices. More precisely, they state that vertex $i = 1$ can only belong to the first connected component ($K_1 = \{1\}$). Likewise, vertex $i = 2$ either belongs to the first connected component or to the second, and so on. Constraints (8)-(9) enforce that the sum of weights of each connected component lies in the desired interval $[l, u]$. Constraints (7) impose that each cluster must include at least one vertex of V.

Note that inequalities (10) are a lifting of the logical constraints

$$x_{ij} + y_i^{k_1} + y_j^{k_2} \leq 2, k_1 \in K_i, k_2 \in K_j, k_1 \neq k_2, \{i,j\} \in E. \quad (17)$$

The latter, as well as its stronger version (10), enforces that edge $\{i,j\}$ cannot belong to the forest if its endpoints are assigned to two different connected components. Disjunctive constraints (11) and (12) serve the same purpose, but use different modeling arguments; they enforce that an edge $\{i,j\}$ can only be included in the forest if its endpoints are assigned to the same connected component index. For these constraints, big-M parameter M_{ij} represents the maximum difference between the indexes of the clusters where vertices i and j are placed in and is given by

$$M_{ij} = \min\{\max\{i,j\}, \overline{K}\} - 1.$$

Note that if $x_{ij} = 0$, constraints (11) and (12) are trivially satisfied. However, if $x_{ij} = 1$, $\sum_{k=1}^{K_i} k y_i^k = \sum_{k=1}^{K_j} k y_j^k$ must hold. Because of the discreteness of $\mathbf{y}$ and due to (6), $y_i^k = y_j^k$ must hold, for a given $k = 1, \ldots, \min\{K_i, K_j\}$. Although we do not have numerical or theoretical evidence showing that constraints (11) and (12) improve the linear programming bounds, provided that (10) are in place, they were kept in the model. We decided to do so, since we empirically found that their inclusion helped the overall performance of the Branch-and-cut algorithm.

Constraints (13) state that the number of clusters is precisely the number of variables z^k activated. Constraints (15) impose that a vertex i cannot belong to a cluster k unless the forest has at least k components.

2.1 Additional valid inequalities

In the remaining of the text, assume that $U(S)$ denotes the minimum number of bins of size u, required to fit the vertices in $S \subseteq V$. Accordingly, let $\underline{U}(S)$ denote any valid lower bound on $U(S)$. A widely known lower bound on $U(S)$ is $\lceil \frac{w(S)}{u} \rceil$.

Subtour elimination constraints (4) can be lifted to the *capacity constraints* (CC)

$$x(E(S)) \leq |S| - \underline{U}(S), \qquad \forall S : S \subset V, |S| \geq 2. \quad (18)$$

To the best of our knowledge, capacity constraints (18) were introduced for the Capacitated Vehicle Routing Problem in [16]. In that context, weights $\{w_i : i \in V\}$ represent demands that must be collected by a vehicle of capacity u and $U(S)$ denotes the minimum number of vehicles of capacity u needed to collect the total demand $w(S)$ associated to S. Here, these inequalities avoid that clusters with weights exceeding u induce trees of the forest. Although such conditions are already granted by constraints (8), the inclusion of CCs (18) improves linear programming relaxation bounds. From now on, assume that $\mathcal{P}^+$ denotes the intersection of polytope $\mathcal{P}$ with capacity constraints (18).

The formulation can also be strengthened by the capacity cutset constraints (CCC)

$$x(\delta(S)) \geq 1 \quad (19)$$

defined by sets $S \neq \emptyset, S \subset V$, satisfying the following conditions:

(1) $(S, E(S)) \in C$;
(2) $w(S) < l$.

In inequalities (19), $\delta(S) = \{\{i,j\} \in E : i \in S, j \notin S\}$ stands for the subset of edges with exactly one endpoint in $S \subset V$. Validity of inequalities (19) for MCBCCP comes from the following observations. Let $V_1 \subset V$ be a balanced and connected cluster such that $S \cap V_1 \neq \emptyset$. Since $w(S) < l$, it is clear that $V_1 \setminus S \neq \emptyset$ since otherwise $w(V_1) < l$ would hold. Since $(V_1, E(V_1))$ is connected, at least one edge connecting S to $V_1 \setminus S$ must be chosen.

CCCs are also valid for subsets $S : w(S) < l$ whose subgraphs $(S, E(S))$ are not connected. However, for this case, the right hand side can be lifted to the number of connected components of $(S, E(S))$ and the inequality can be seen as a (weaker) surrogate version of those defined by the connected subsets contained in S. For the remaining of the text, assume that $\mathcal{P}^{++}$ denotes the intersection of $\mathcal{P}^+$ with CCCs (19).

3 BRANCH-AND-CUT ALGORITHM

In this section, we describe the main features of the MCBCCP Branch-and-cut algorithm BC^{++} based on formulation $\mathcal{P}^{++}$. The algorithm dynamically separates two classes of MCBCCP valid inequalities, CCs (18) and CCCs (19). BC^{++} first solves the Linear Program (LP)

$$\min \left\{ K : (K, \mathbf{x}, \mathbf{y}, \mathbf{z}) \in \hat{\mathcal{P}} \right\}, \quad (20)$$

where $\hat{\mathcal{P}}$ is the polytope given by the intersection of constraints (3), SECs (4) defined by sets $S = \{i,j\} \in E$, (5)-(16) and

$$x(\delta(S)) \geq 1, \ S \in \mathcal{S}, \quad (21)$$

where $\mathcal{S} = \{S \subset V : w(S) < l, 1 \leq |S| \leq Max, (S, E(S)) \in C\}$. We found advantageous to include all CCCs defined for sets $S : |S| \leq Max$ in the very first linear programming relaxation (in our implementation, $Max = 4$). CCCs defined by sets with more than Max vertices are identified on-the-fly, as described in the sequence.

Assume that the LP (20) is feasible and denote by $(\hat{K}, \hat{\mathbf{x}}, \hat{\mathbf{y}}, \hat{\mathbf{z}})$ an optimal solution to it. BC^{++} then attempts to strengthen the relaxation $\hat{\mathcal{P}}$ by appending to it some CCs and CCCs, violated by $\hat{\mathbf{x}}$. The separation procedures for these two sets of valid inequalities makes use of a separation engine designed for the SEC (4) separation problem, outlined in [2]. The engine comprises heuristic and exact algorithms for the identification of violated SECs. The idea is to use these SEC separation procedures to provide candidate sets of vertices for which the violation of CCCs and CCs can be checked.

More precisely, the SEC separation engine involves a Kruskal like heuristic and the exact SEC separation algorithm introduced in [22]. The heuristic first sorts the edges in $\hat{E} = \{\{i,j\} \in E : \hat{x}_{ij} > 0\}$, in a non increasing order of their linear programming relaxation values $\{\hat{x}_{ij} : \{i,j\} \in \hat{E}\}$. In turn, edges in the list are used to find a maximum cardinality spanning forest of G, in a Kruskal like fashion. Assuming that $e = \{i,j\}$ is the edge to be processed, the heuristic merges the connected components where i and j are placed into a single subset S of vertices. The lower bound $\underline{U}(S) = \lceil \frac{w(S)}{u} \rceil$ is computed and the corresponding CC (18) is checked for violation. The same set S is checked for the identification of a violated CCC (19): whenever $w(S) < l$, we check if $\hat{x}(\delta(S)) < 1$ applies.

Violated CCs and CCCs are stored in two separate lists. At the end of the application of the heuristic, the most violated inequality in each list reinforces the relaxation $\hat{\mathcal{P}}$. Remaining inequalities in the list are only included in $\hat{\mathcal{P}}$ if they are sufficiently orthogonal to the most violated inequality of its class, found at that separation round. To be more specific, assume that $\mathbf{a}_1^T\mathbf{x} \geq \mathbf{b}_1$ and $\mathbf{a}_2^T\mathbf{x} \geq \mathbf{b}_2$ are two violated valid inequalities of the same class, CC or CCC, stored in the list, at that separation round. Assume as well that the first is the most violated inequality of that class. Inequality $\mathbf{a}_2^T\mathbf{x} \geq \mathbf{b}_2$ is only added to $\hat{\mathcal{P}}$ if $\frac{|\langle \mathbf{a}_1, \mathbf{a}_2 \rangle|}{\|\mathbf{a}_1\|_2 \|\mathbf{a}_2\|_2} \leq \epsilon$, where $\epsilon \in [0, 1)$ is an implementation parameter that controls the desired level of orthogonality (in our implementation, $\epsilon = 0.5$). Inequalities that do not satisfy the orthogonality criteria are discarded.

The exact SEC separation algorithm in [22] is called next, only if no violated inequalities were found by the SEC separation heuristic. Without giving much of the details of that procedure, it suffices to mention that the algorithm solves a series of $n - 2$ max-flow (min-cut) problems in a conveniently defined directed network, obtained from $\hat{\mathbf{x}}$ and $\hat{E}$, in order to find violated SECs. The optimal set of vertices in each of these min-cut problems is checked for the violation of CCs and CCCs. Again, the same orthogonality criteria is applied in order to decide which inequalities stored in the lists are discarded or used to reinforce the relaxation $\hat{\mathcal{P}}$.

The process outlined above is carried out for each BC^{++} node, until no violated inequalities are found by the separation engine. If the solution to that linear programming relaxation is integer feasible, the optimal solution to the node under investigation was found, and the node is pruned by optimality. Being fractional, BC^{++} branches on variables, giving preference to branch first on variables $\mathbf{x}$.

BC^{++} makes use of the XPRESS mixed integer optimization package (release 27.01.02) in charge of managing the search tree. All pre-processing and cutting plane generation procedures embedded in the solver are turned off. No multi-threading is allowed. Since we still have not implemented MCBCCP primal heuristics to speed up the search, preference is given to find feasible solutions as early as possible. Thus, BC^{++} implements a depth-first search and the solver's linear programming heuristics are turned on.

4 COMPUTATIONAL EXPERIMENTS

Our computational experiments were conduced with two sets of test instances: cb and g. The first one, cb, comprises benchmark instances for the min-cut problem addressed by Johnshon et al. [14]. They represent real compiler construction problems, where each vertex in the (typically very sparse) input graph represent modules of code that need to be combined to form clusters. In total, five graphs representing the compiler construction problem were used. Each of these instances is indicated by cb_*id*, where *id* represents one of the five graphs. In that application, edges' weights represent the communication cost between modules of code. In the MCBCCP case, these weights are not applicable, being simply ignored. In the application described in [14], clusters are restricted in their total memory storage. We used the same values of $u \in \{450, 512\}$ considered in [14]. Since in that reference l was assumed to be zero, we used $l = \lfloor 10\% u \rfloor$ and $l = \lfloor 20\% u \rfloor$.

The second set of instances considered here, g, represent grid graphs. These instances were generated here, in an attempt to address another application of MCBCCP, that of clustering geographical contiguous areas into political districts. Each grid has $s \in \{8, 12\}$ rows and columns. For each value of s, five instances were generated. Each instance is identified by the word g_s_id, where $id \in \{1, \ldots, 5\}$ is an integer representing a particular instance of size s. One vertex, placed at the center of each of these s^2 grid cells, represents the cell. The set E includes one edge $\{i, j\}$ for each pair of neighboring cells i and j. Integer weights $\{w_i : i \in V\}$ were randomly chosen in the interval $[10, 100]$, with uniform probability. Depending on the values of s, different values of u were chosen: for $s = 8$, $u = 350$ and for $s = 12$, $u = 650$. These values of u account for about 10% of $w(V)$. The values of l were chosen as before: $l = \lfloor 10\% u \rfloor$ and $l = \lfloor 20\% u \rfloor$. Considering the different values of u and l involved in our experiments, 40 MCBCCP instances were tested. Data for the instances used in our testings are given in Table 1. That table provides, for each instance, the corresponding values of n, m, the minimum and the maximum values of w_i and, finally, $w(V)$.

The algorithms outlined in this paper were implemented in C and compiled with gcc with optimization flags -O3 turned on, under Linux OS. Experiments were conducted with a Intel i7-5820K processor, running at 3.3GHz, with 32Gbytes of RAM memory.

In Table 2, we present computational results of two Branch-and-cut implementations: BC^{++} and BC^{+}. The first one is based on formulation $\mathcal{P}^{++}$ and, thus, makes use of CCCs (19). The other one does not, being based on the (weaker) model $\mathcal{P}^{+}$. Therefore, BC^{+} neither includes inequalities (21) in its first linear programming relaxation nor calls the procedures for the indentification of violated constraints (19), within the separation engine we described earlier. Apart from that, the two algorithms share every other implementation strategy.

The first two columns of the table provide the instance and the value of u under consideration. The table is divided in two sets of rows; the upper part of the table is dedicated to the $l = \lfloor 20\% u \rfloor$ instances, while the bottom reports results for $l = \lfloor 10\% u \rfloor$. For each BC implementation, the table presents: the root node lower bound (LB), the best lower (BLB) and upper bound (BUB) found at the end of the search or when the time limit of 1800 seconds was

Table 1: Data for the instances used in the computational experiments.

Inst.	n	m	min w_i	max w_i	$w(V)$
cb_1	30	47	19	298	2497
cb_2	45	98	14	298	3325
cb_3	47	99	14	298	3425
cb_4	47	101	14	278	3890
cb_5	61	187	15	165	3704
g8_1	64	112	11	99	3472
g8_2	64	112	13	100	3732
g8_3	64	112	10	100	3272
g8_4	64	112	10	100	3411
g8_5	64	112	12	100	3757
g12_1	144	264	10	100	8000
g12_2	144	264	10	99	7999
g12_3	144	264	11	100	8013
g12_4	144	264	10	100	8153
g12_5	144	264	10	100	7997

hit, the CPU times (in seconds) needed to obtain these bounds, $t(s)$, and finally the number of nodes explored by the search trees. An indication "-" is provided in the CPU time columns, whenever the time limit was hit and the corresponding BC implementation either did not prove the instance infeasibility or did not solve it to proven optimality. Whenever an algorithm did not find a feasible solution within the time limit, ∞ is reported in the corresponding BUB column entry.

The values of l and u involved in our testings led to 39 feasible instances out of the 40 available. BC^{++} managed to prove the infeasibility of that instance quite fast, right at the root node, in contrast to BC^{+}, that spent the entire time limit without concluding so. Considering the 39 feasible instances of our study, better results were also obtained by BC^{++}. While it managed to solve 24 out of the 39 feasible instances to proven optimality within the 1800 seconds time limit, BC^{+} solved only 12. Considering the 11 instances solved to optimality by both methods, computational results also lean in favor of BC^{++}. It was faster than BC^{+} in 7 out of these 11 cases. BC^{++} was also capable of delivering higher quality integer feasible solutions, if our attention now moves to the 14 instances left unsolved by both methods. In all these cases, BC^{++} found sharper upper bounds than BC^{+}. The latter did not find a feasible solution within the time limit for 14 out of the 39 feasible instances of our test set.

It is worthwhile mentioning that the inclusion of inequalities (19) resulted in small (if any) increases in the root node linear programming bounds, for the lowest values of l. That comes as a result of two aspects. The first is the nature of the objective function, that involves the minimization of K, while every other variable has the same null cost. The second is the heuristic nature of our separation engine for the separation problem associated to CCs (18) and to CCCs (19). As a consequence of that, there is no guarantee that the cutting plane algorithm of BC^{++} delivers a stronger bound even if, for a particular instance, the optimal solution $\mathbf{x}^{++}$ to the linear programming relaxation defined by $\mathcal{P}^{++}$ does not belong to $\mathcal{P}^{+}$. That explains why, for two cases, BC^{+} delivered root node lower bounds slightly stronger than those provided by BC^{++}. Nevertheless, the inclusion of (19) as well as the lifting (17) of (10) had a positive impact on the performance of the full search tree. In general, fewer nodes are investigated in less CPU time.

5 CONCLUSIONS AND FUTURE WORK

In this paper, we investigated the Minimum Cardinality Balanced and Connected Clustering Problem (MCBCCP). We introduced an integer programming formulation along with valid inequalities for the problem. Additionally, we implemented and tested a Branch-and-cut algorithm based on that model.

Our preliminary numerical experiments indicated that the valid inequalities introduced here, the lifting (10) of the logical constraints (17) and the capacity cutset constraints (19), had a positive impact on the computational results. In particular, the Branch-and-cut algorithm that separates capacitated cutset inequalities (19) obtained more optimality certificates than its version that does not. It also found higher quality solutions for those instances left unsolved when the time limit is hit.

The capacity cutset constraints (19) introduced here, alongside capacity constraints (18), allows the problem to be formulated without the need of variables $\mathbf{y}, \mathbf{z}$. To be more specific, denote by $\mathbf{P}_x \subset \mathbb{R}^{m+1}$ the intersection of (3) and (5) alongside the capacity constraints (18) and cutset constraints (19). MCBCCP can be formulated as $\min\{K : (K, \mathbf{x}) \in \mathcal{P}_x \cap (\mathbb{R} \times \mathbb{B}^m)\}$.

A drawback of the formulation introduced here is its symmetry. To alleviate that, we restricted the indexes of the clusters that can be assigned to the vertices of G. The lifting (10) of (17) also helps in that matter. From an algorithmic perspective, we enforced that the Branch-and-cut algorithm first branches on $\mathbf{x}$ variables.

A promising alternative to deal with symmetry is to use the concept of *representatives* [4] to formulate the problem. For the MCBCCP case, such a formulation involves binary decision variables $\mathbf{v} = \{v_{ij} \in \mathbb{B} : i, j \in V\}$. Variable v_{ij} assumes value 1 if

Table 2: Branch-and-cut algorithm: Computational results with and without inequalities (19)

$l = \lfloor 20\% u \rfloor$

		BC^{++}					BC^{+}				
Inst.	u	LB	BLB	BUB	$t(s)$	nodes	LB	BLB	BUB	$t(s)$	nodes
cb_1	450	infeasible			0.2	1	7	9	∞	-	57002
cb_2	450	9	10	10	163.0	7376	8.5	9	∞	-	26262
cb_3	450	9	10	10	106.6	3517	8.2	9	∞	-	30337
cb_4	450	9	9	9	439.6	8556	9	9	10	-	16424
cb_5	450	9	9	9	44.9	818	9	9	∞	-	20373
cb_1	512	6	6	6	0.6	9	5.3	6	6	33.7	7028
cb_2	512	7.5	8	8	80.6	3564	7.3	8	∞	-	28250
cb_3	512	7	8	8	124.2	3145	7	7	∞	-	20071
cb_4	512	8	8	8	79.8	2404	8	8	8	119.9	5621
cb_5	512	8	8	8	31.7	468	8	8	8	216.5	9204
g8_1	350	11	11	11	23.4	521	11	11	11	206.4	8891
g8_2	350	11	12	12	254.3	6165	11	12	13	-	16150
g8_3	350	10	10	10	34.1	766	10	10	11	-	24266
g8_4	350	10	11	11	831.3	7625	10	11	11	434.1	4161
g8_5	350	11	11	12	-	8041	11	12	12	207.0	4161
g12_1	650	13	13	15	-	8902	13	13	∞	-	9912
g12_2	650	13	13	19	-	8558	13	13	∞	-	9336
g12_3	650	13	13	16	-	6591	13	13	∞	-	9954
g12_4	650	13	13	15	-	12252	13	13	19	-	11618
g12_5	650	13	13	15	-	6196	13	13	∞	-	9493

$l = \lfloor 10\% u \rfloor$

		BC^{++}					BC^{+}				
Inst.	u	LB	BLB	BUB	$t(s)$	nodes	LB	BLB	BUB	$t(s)$	nodes
cb_1	450	6.5	8	8	1.5	167	7	8	8	6.1	1069
cb_2	450	9	9	10	-	25546	8.4	9	12	-	23702
cb_3	450	8.3	9	11	-	27518	8.2	9	13	-	23303
cb_4	450	9	9	9	234.3	6198	9	9	9	208.0	5374
cb_5	450	9	9	9	61.5	1531	9	9	12	-	19561
cb_1	512	5.3	6	6	4.0	639	5.2	6	6	12.3	3869
cb_2	512	7.3	8	9	-	29238	7.2	8	10	-	23258
cb_3	512	7	7	8	142.7	6899	7	7	8	-	28586
cb_4	512	8	8	8	24.5	1340	8	8	8	92.2	5426
cb_5	512	8	8	8	29.9	340	8	8	8	42.9	610
g8_1	350	11	11	11	42.8	1275	11	11	11	1775.8	13153
g8_2	350	11.0	12	12	881.8	9273	11.1	12	13	-	14082
g8_3	350	10	10	11	-	9296	10	10	11	-	14329
g8_4	350	10	11	11	1245.9	9267	10	10	11	-	16581
g8_5	350	11	12	12	1399.0	8919	11	12	12	201.8	3141
g12_1	650	13	13	17	-	5432	13	13	∞	-	4079
g12_2	650	13	13	16	-	7043	13	13	∞	-	3938
g12_3	650	13	13	18	-	3129	13	13	∞	-	3723
g12_4	650	13	13	18	-	3747	13	13	∞	-	4445
g12_5	650	13	13	16	-	4302	13	13	∞	-	4941

vertex j is the representative (or cluster head) of the connected component where i is placed. Following the strategy to reduce symmetry and the number of $\mathbf{y}$ variables, these variables can be restricted to $\{v_{ij} : i,j \in V, j \leq i\}$, indicating that the representative of a cluster is always the vertex with the least index among those in the same tree of the forest. This formulation does not involve decision variables $\mathbf{y}$ and $\mathbf{z}$ and the objective function, to be minimized, is $\sum_{i \in V} v_{ii}$. In addition to constraints $\{v_{ij} \leq v_{jj}, i,j \in V, j < i\}$, the model includes constraints akin to the inequalities defining $\mathcal{P}^{++}$, except to (14) to (7) that have no meaning in the new variable setting. Preprocessing these $\mathbf{v}$ variables could be carried out as follows. Define $D = (V, A)$ as the directed graph obtained by duplicating the edges of E, into two arcs of opposite directions. Assume that the length of an arc $(i,j) \in A$ is $c_{ij} = w_j$. Whenever the length of the shortest path connecting i and j exceeds u, the representative of i and j cannot be the same, and assuming that $j < i$ applies, variable v_{ij} would not be required.

We also plan to investigate set partitioning formulations for MCBCCP and Branch-and-cut-and-price algorithms based on them. One possible formulation along these lines is similar to the one introduced in [14] for a fixed cardinality clustering problem. In addition to edge variables $\mathbf{x}$, it uses exponentially many binary decision variables associated to all balanced subsets of vertices of V. The master problem is defined by set partitioning constraints enforcing that every vertex must be included in one of such subset of vertices, constraints (19) and (18), as well as another type of constraints that couple $\mathbf{x}$ and the exponentially many variables of sets of vertices. Such coupling constraints express the following idea: *an edge $\{i,j\}$ cannot be selected by the master program unless its endpoints i and j are included in the same balanced set.* Because of that type of coupling constraints, the associated pricing problem consists of solving a Constrained Quadratic Binary Problem, a variation of the Maximum Weight Binary Knapsack Problem (QKP), where not only knapsack constraints $w(S) \leq u$ but also covering constraints $w(S) \geq l$ must be enforced. A nice feature of this problem is that, due to the sign of the dual variables in the master program, diagonal entries of the quadratic cost matrix should be negative while off-diagonal ones should be positive.

MCBCCP also has connections with network design problems found in the literature, for instance, the Capacited Minimum Spanning Tree Problem (CMSTP) [10, 26]. Actually, we can reformulate MCBCCP as a variation of the CMSTP. To that aim, consider a directed graph $D = (V \cup \{r\}, A)$, where the arc set A involves two arcs, in opposite directions, for every edge in E, as well as one arc pointing from the artificial root vertex r to every vertex $i \in V$. Arcs connecting r to $i \in V$ cost one while the remaining ones cost zero. The goal is to find a minimum cost spanning arborescence of D, rooted out of r, such that the sum of the weights of the vertices in every tree rooted in $i \in V$ satisfy (1). We plan to investigate models and algorithms for MCBCCP based on such an idea, including those where the desired arborescence topology is enforced by network flow and set partitioning constraints.

So far, we have not investigated primal heuristics for the problem. One possible approach to fill that gap benefits from the Dynamic Programming algorithm in [13], capable of exactly solving MCBCCP in polynomial time, when G is a spanning tree. The idea is to build a spanning tree of G, driven by the linear programming relaxations $\{\hat{x}_{ij} : \{i,j\} \in \hat{E}\}$ provided by our BC algorithms. In turn, that spanning tree is used as an input graph for the exact MCBCCP algorithm in [13].

These ideas, in full or in part, should complement the material presented in this paper and should be presented at the next International Network Optimization Conference.

REFERENCES

[1] E.R. Barnes. 1982. An algorithm for partitioning the nodes of a graph. *SIAM Journal on Algebraic and Discrete Mathematics* 3 (1982), 541–555.

[2] Bicalho, L., da Cunha, A.S and Lucena, A. 2016. Branch-and-cut-and-price algorithms for the Degree Constrained Minimum Spanning Tree Problem. *Computational Optimization and Applications* 63 (2016), 755–792.

[3] P. Brucker. 1978. *Optimization and Operations Research.* Lecture Notes in Economics and Mathematical Sciences, Vol. 157. Springer, Chapter On the complexity of clustering problems, 45–54.

[4] Manoel Campêlo, Ricardo Corrêa, and Yuri Frota. 2004. Cliques, holes and the vertex coloring polytope. *Inform. Process. Lett.* 89, 4 (2004), 159 – 164.

[5] Marcus Poggi de Aragão and Eduardo Uchoa. 1999. The γ−connected assignment problem. *European Journal of Operational Research* 118 (1999), 127–138.

[6] Jitender S. Deogun, Dieter Kratsch, and George Steiner. 1997. An approximation algorithm for clustering graphs with dominating diametral path. *Inform. Process. Lett.* 61, 3 (1997), 121 – 127.

[7] J. Edachery, A. Sen, and F. J. Brandenburg. 1999. *Graph drawing.* Lecture Notes in Computer Science, Vol. 1731. Springer, Chapter Graph Clustering Using Distance-k cliques, 98–106.

[8] J. Falkner, F. Rendl, and H. Wolkowicz. 1994. A computational study of graph partitioning. *Mathematical Programming* 66 (1994), 211–224.

[9] M.R. Garey, D.S. Johnson, and L. Stockmeyer. 1976. Some simplified NP-complete graph problems. *Theoretical Computer Science* 1, 3 (1976), 237–267.

[10] Bezalel Gavish. 1983. Formulations and Algorithms for the Capacitated Minimal Directed Tree Problem. *J. ACM* 30, 1 (Jan. 1983), 118–132.

[11] W. Hager and Y. Krulyuk. 2002. Multiset graph partitioning. *Mathematical Methods of OR* 55 (2002), 1–10. Issue 1.

[12] William W. Hager, Dzung T. Phan, and Hongchao Zhang. 2013. An exact algorithm for graph partitioning. *Mathematical Programming* 137 (2013), 531–556.

[13] Takehiro Ito, Takao Nishizeki, Michael Schröeder, Takeakin Uno, and Xiao Zhou. 2012. Partitioning a Weighted Tree into Subtrees with Weights in a Given Range. *Algorithmica* 62 (2012), 823–841.

[14] Ellis L. Johnson, Anuj Mehrotra, and George L. Nemhauser. 1993. Min-cut clustering. *Mathematical Programming* 62 (1993), 133–151.

[15] N. P. Kruyt. 1997. A conjugate gradient method for the spectral partitioning of graphs. *Parallel Comput.* 22 (1997), 1493–1502.

[16] Gilbert Laporte, Yves Nobert, and Martin Desrochers. 1985. Optimal Routing under Capacity and Distance Restrictions. *Operations Research* 33 (1985), 1050–1073. Issue 5.

[17] Isabella Lari, Federica Ricca, Justo Puerto, and Andrea Scozzari. 2016. Partitioning a Graph into Connected Components with Fixed Centers and Optimizing Cost-Based Objective Functions or Equipartition Criteria. *Networks* 67 (2016), 69–81. Issue 1.

[18] F. Murtagh. 1985. A Survey of Algorithms for Contiguity-constrained Clustering and Related Problems. *Comput. J.* 28 (1985), 82–88. Issue 1.

[19] F Murtagh. 2003. Maximum Split Clustering Under Connectivity Constraints. *Journal of Classification* 20 (2003), 143–180.

[20] Jenny Nossack and Erwin Pesch. 2014. A branch-and-bound algorithm for the acyclic partitioning problem. *Computers & Operations Research* 41 (2014), 174–184.

[21] M. W. Padberg and G. Rinaldi. 1991. A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems. *SIAM review* 33, 1 (1991), 60–100.

[22] M. W. Padberg and L. A. Wolsey. 1983. Trees and cuts. *Annals of Discrete Mathematics* 17 (1983), 511–517.

[23] A. Pothen, H. D. Simon, and K. P. Liou. 1990. Partitioning sparse matrices with eigenvectors of graphs. *SIAM J. Matrix Anal. Appl.* 11 (1990), 430âĂŞ452.

[24] F. Rendl and H. Wolkowicz. 1995. A projection tecnhique for partitioning the nodes of a graph. *Ann. Oper. Res.* 581 (1995), 172–191.

[25] Satu Elisa Schaeffer. 2007. Graph clustering. *Computer Science Review* 1, 1 (2007), 27 – 64.

[26] E. Uchoa, R. Fukasawa, J. Lysgaard, A. Pessoa, M Poggi de Aragão, and D. Andrade. 2008. Robust branch-cut-and-price for the Capacitated Minimum Spanning Tree problem over a large extended formulation. *Mathematical Programming* 112 (2008), 446–472.

[27] H. Wolkowicz and Q. Zhao. 1999. Semidefinite programming relaxations for the graph partitioning problem. *Discrete Appl. Math.* 96/97 (1999), 467–547.

A Branch-and-Bound Algorithm for the Maximum Weight Perfect Matching Problem with Conflicting Edge Pairs

Temel Öncan[*]
Endüstri Mühendisliği Bölümü
Galatasaray Üniversitesi
İstanbul, TÜRKİYE
ytoncan@gsu.edu.tr

M. Hakan Akyüz
Endüstri Mühendisliği Bölümü
Galatasaray Üniversitesi
İstanbul, TÜRKİYE
mhakyuz@gsu.edu.tr

İ. Kuban Altınel
Endüstri Mühendisliği Bölümü
Boğaziçi Üniversitesi
İstanbul, TÜRKİYE
altinel@boun.edu.tr

ABSTRACT

This paper introduces a branch-and-bound (B&B) algorithm for the maximum weight perfect matching problem with conflicting edge pairs which is an $\mathcal{NP}$-hard problem. The proposed B&B algorithm is based on the relaxation obtained by removing the cardinality restriction on the feasible matchings and uses a non-dichotomized branching rule considering exposed vertices in a relaxed optimum solution. We have performed extensive computational experiments on randomly generated test instances and compared the proposed B&B algorithm with two Binary Integer Linear Programming models solved with an off-the-shelf commercial solver. According to our experiments, we have observed that the proposed B&B algorithm yields promising performance.

KEYWORDS

Integer Programming, Maximum Weight Perfect Matching, Branch-and-Bound, Conflicts

1 INTRODUCTION

The well-known *Maximum Weight Perfect Matching Problem* (MWPMP) consists of finding a perfect matching with maximum total weight [9]. The MWPMP is known to be polynomially solvable and it has several applications in scheduling, facility location and workforce planning [1]. In this work, we address an extension of the MWPMP with additional conflicting edge pair constraints. The so-called conflict constraints are also referred to as *the exclusionary side constraints* or *the disjunctive constraints*. Hence, the extended problem is named as the *Maximum Weight Perfect Matching Problem with Conflicting Edge Pairs* (MWPMC) which deals with determining a maximum weight perfect matching such that no two conflicting edges are in the solution at the same time, namely a maximum weight conflict free perfect matching. The MWPMC is known to be $\mathcal{NP}$-hard [7].

As a practical application of the MWPMC, we can mention the case arising in logistics, where toxic chemical substances and foods are prohibited to be stored in the neighbor locations. In a potential extension of the ordinary Symmetric Traveling Salesman Problem (STSP) there can be an incompatibility relation between the edges incident with vertices: some of them may not be selected if a particular edge is in the tour and a tour can consist of only compatible edges. This scenario is possible due to security reasons during the routing of an important person. Recall that a tour for a salesperson is a connected spanning subgraph in which all points have degree 2. If we drop the connectedness requirement, i.e. the subtour elimination constraints, the STSP turns into the determination of optimum 2-factors. The 2-factor problem is a natural extension of the perfect matching problem and in fact the determination of an optimal 2-factor reduces to the determination of an optimal perfect matching. In other words, the MWPMC can be viewed as a relaxation of the mentioned STSP extension.

In the literature, several combinatorial optimization problems with conflict constraints have been addressed. Among them we can mention, the minimum spanning tree problem with conflict constraints [6, 7, 16, 22, 24], the shortest path problem with conflict constraints [7], the transportation problem with exclusionary side constraints [10, 13, 23], the knapsack problem with conflict constraints [3, 4, 19], the bin packing problem with conflict constraints [5, 12, 21], the maximum flow problem under conflict and forcing constraints [20] and the minimum cost non-crossing flow problem on layered networks [2].

For all we know, the only work addressing the MWPMC is performed by Darmann et al. [7] where they provide complexity results of this problem and discuss its approximation hardness. As a special case, the MWPMC on bipartite graphs has been considered by Öncan et al. [16] and Öncan and Altınel [15]. In [16], the authors have introduced some complexity results as well as polynomially solvable cases. They have also proposed heuristics and lower bounding procedures. Recently, Öncan and Altınel [15] have developed two branch-and-bound (B&B) algorithms with dichotomized branching rules for the MWPMC in bipartite graphs.

The motivation of this work is to devise an exact solution approach, namely a specially tailored B&B algorithm, for the MWPMC in general graphs. Two Binary Integer Linear Programming (BILP) formulations are also proposed for the MWPMC. Computational experiments are performed on randomly generated test instances in order to compare the performance of the proposed B&B algorithm with the ones of the BILP formulations solved with CPLEX Mixed-Integer Linear Programming (MILP) solver. We have observed that the proposed B&B algorithm yields an outstanding performance for most of the cases.

In the next section, we introduce some definitions which are used throughout the paper and present two BILP formulations for the MWPMC. Then, in Section 3, we give the outline of the new B&B algorithm. Section 4 is where we report the experimental results. Finally, concluding remarks are discussed in Section 5.

2 TWO BINARY INTEGER LINEAR PROGRAMMING FORMULATIONS

Let $G = (V(G), E(G))$ be a graph, where $V(G)$ and $E(G)$ stand for the set of vertices and edges, respectively. We associate non-negative weights w_e for all edges $e \in E(G)$ and let $\delta_G(v)$ denote the subset of edges incident with vertex $v \in V(G)$. Then, the degree of vertex $v \in V(G)$ is defined as $d_G(v) = |\delta_G(v)|$ where

[*] Corresponding author

ISBN 978-3-89318-079-0 on OpenProceedings, ISSN 2510-7437

DOI: 10.5441/002/inoc.2019.07

$|\cdot|$ stands for the cardinality of a set. Besides, the *complement* of G is defined as the graph $\overline{G} = (V(\overline{G}), E(\overline{G}))$ where $V(\overline{G}) = V(G)$ and $E(\overline{G}) = \{\{u, v\} \notin E(G) : u, v \in V(G), u \neq v\}$.

A *stable set* (independent vertex set) of G is any subset $S \subseteq V(G)$ such that no two vertices in S are adjacent. The *Maximum Cardinality Stable Set Problem* (MCSP) consists of finding a stable set with a maximum number of vertices. This number is so-called as the stability number of G which is designated as $\alpha(G)$. When we associate weights to all vertices in $V(G)$, for every subset $S \subseteq V(G)$ the weight $w(S)$ is computed as the sum of the weights of vertices in S. The *Maximum Weight Stable Set Problem* (MWSP) tries to find a stable set S of G with maximum weight $w(S)$ which is represented as $\alpha_w(G)$. Both MCSP and MWSP are well-known $\mathcal{NP}$-complete combinatorial optimization problems [11].

A *matching* (independent edge set) $M = (V(M), E(M))$ of G is defined as a subset of $E(G)$ where no two edges share the same vertex. A *perfect matching* stands for a matching such that each vertex of $V(G)$ is incident with exactly one of the edges in the matching [8, 14]. The weight of a matching, i.e. $w(E(M))$, is calculated as the sum of the edge weights in the matching. Formally speaking, MWPMP tries to find a perfect matching M of G with maximum $w(E(M))$.

A *clique* $K = (V(K), E(K))$ is a complete subgraph of G. A clique is maximal if no other vertex $v \in V(G) \setminus V(K)$ is adjacent to all vertices in $V(K)$. Clearly, each clique in G corresponds to a stable set in $\overline{G}$. Therefore, the MCSP and the MWSP defined on G are equivalent to the *Maximum Clique Problem* (MCP) and the *Maximum Weight Clique Problem* (MWCP) on $\overline{G}$, respectively.

Now we are at the stage to present two BILP formulations for the MWPMC. Given a set of conflicting edges with each edge $e \in E(G)$, the MWPMC tries to find a perfect matching M such that no two conflicting edges e and f are allowed to be in $E(M)$. The conflicting edge pairs can be represented with a conflict graph $C = (V(C), E(C))$ where $V(C) \equiv E(G)$ and each conflicting edge pair corresponds to an edge in $E(C)$. The set of conflicting edges with edge $e \in E(G)$ is denoted as $\delta_C(e)$ and the degree of vertex e in the conflict graph C is $|\delta_C(e)| = d_C(e)$. In other words, the set of edges incident with vertex e is represented with $\delta_C(e) \subseteq E(C)$ in the conflict graph. Note that, when $f \in \delta_C(e)$ then $e \in \delta_C(f)$ and for two edges $e, f \in E(G)$ such that $\{e, f\} \in E(C)$.

Let the binary decision variable x_e be equal to 1 if and only if edge $e \in E(G)$ is in the perfect matching. Recall that w_e represents non-negative weight for edge $e \in E(G)$. Then we can formulate the MWPMC as follows:

$$\max z = \sum_{\{e\} \in E(G)} w_e x_e \tag{1}$$

subject to

$$\sum_{e \in \delta_G(v)} x_e = 1 \quad \text{for } v \in V(G) \tag{2}$$

$$x_e + x_f \leq 1 \quad \text{for } e \in E(G); f \in \delta_C(e) \tag{3}$$

$$x_e \in \{0, 1\} \quad \text{for } e \in E(G) \tag{4}$$

The objective function (1) is to minimize the weight of the perfect matching, i.e. $w(E(M))$. Constraints (2) enforce that every vertex is connected to exactly one of the edges in the solution. Constraints (3) obviate the conflicting edge pairs to be in the perfect matching. Constraints (4) are for the binary restrictions on the decision variables.

Notice that when we aggregate constraints (3) for all $f \in \delta_C(e)$ we obtain the following equivalent inequalities:

$$\sum_{f \in \delta_C(e)} x_f + d_C(e) x_e \leq d_C(e) \quad \text{for } e \in E(G) \tag{5}$$

We will call the formulation which consists of (1)-(4), as *STRONG* and the formulation including (1), (2), (4) and (5) as *WEAK*. Note that, when we define P_S and P_W as the polytopes corresponding to feasible solution sets of the Linear Programming (LP) relaxation of the *STRONG* and *WEAK* formulations respectively, then $P_S \subseteq P_W$ holds.

3 A BRANCH-AND-BOUND ALGORITHM FOR THE MWPMC

The proposed B&B algorithm employs maximum weight matching with conflicting edge pair (MWMC) relaxation of the MWPMC, which is obtained when we replace the equality signs '=' in constraints (2) with inequalities '≤'. Hence, at each node of the B&B tree, including the root node, we solve the MWMC relaxation of the MWPMC.

During the exploration of the B&B tree, the solution of the MWMC relaxation yields a conflict free matching which is not necessarily perfect, and hence there may exist a set of exposed vertices in the relaxed optimal solution. Hence, given an exposed vertex $v \in V(G)$, subproblems of the B&B tree are generated by enforcing one by one the edges incident to v to be in the solution. Note that the B&B tree is not necessarily a binary tree.

All B&B nodes but the root node, are characterized by a set of edges. The ones which must be included in the solution and the edges that must be excluded from the solution. The edges in the former set are called as *included edges* and the edges in the latter one are named as *excluded edges*. The remaining edges of $E(G)$ are the *free edges*. Broadly speaking, during the run of the algorithm, at each node of the B&B tree, we consider a set of free edges and enforce them to be included in the solution. Meanwhile, we prune a B&B node either by comparing its upper bound value with the best known lower bound value or by making sure that the current node can not provide a feasible solution, i.e. a conflict free perfect matching. A formal outline of the proposed B&B algorithm is depicted with Algorithm 1.

Now we will discuss the details of the B&B algorithm. To this end, we will introduce some additional notation. Let t be the B&B node index and let $I^{(t)}$ and $X^{(t)}$ stand for the subsets of edges which must be included to and excluded from a conflict free perfect matching at node t of the B&B tree, respectively. Then the subproblem at node t is denoted by $MWPMC^{(t)}$ which is the MWPMC solved on the subgraph $G^{(t)} = (V(G^{(t)}), E(G^{(t)}))$ of the original graph G, with the vertex set $V(G^{(t)})$ obtained by deleting the vertices incident with the edges in $I^{(t)}$ and the edge set $E(G^{(t)})= E(G) \setminus \{I^{(t)} \cup X^{(t)}\}$. For an upper bound on the $MWPMC^{(t)}$ we solve its maximum weight matching with conflicting edge pair relaxation, namely $MWMC^{(t)}$.

Let us define *extended conflict graph* $C^{(t)} = (V(C^{(t)}), E(C^{(t)}))$, at node t of the B&B tree, corresponding to $G^{(t)}$, where $V(C^{(t)})$ and $E(C^{(t)})$ are the set of vertices and edges of $C^{(t)}$, respectively. Vertices of the extended conflict graph $C^{(t)}$ correspond to the edges of $G^{(t)}$, i.e. $E(G^{(t)})$. The weights associated with the edges in $E(G^{(t)})$ are the weights of the vertices in $C^{(t)}$. On the other hand, the edges of the conflict graph represent the set of conflicting edge pairs and the set of incident edge pairs in $G^{(t)}$. Furthermore, we define the complement of $C^{(t)}$ as $\overline{C^{(t)}}$.

3.1 Initialization

At the initialization of the proposed B&B algorithm, we set $t = 0$ and we start with the original graph $G^{(0)} = G$ and its corresponding conflict graph $C^{(0)}$. The best known lower bound $\underline{z}$ is set to $-\infty$. The set of active problem list is initialized with $MWPMC^{(0)}$ and initially, both of the edge subsets $I^{(0)}$ and $X^{(0)}$ are empty.

3.2 Lower Bound

First of all, we check whether $\left|E(G^{(t)})\right| < \frac{|V(G)|}{2} - \left|I^{(t)}\right|$ holds in order to guarantee that the graph $G^{(t)}$ has the potential to yield a perfect matching. Otherwise we prune node t.

Next, we perform another check at the lower bounding step right after finding the maximum cardinality conflict free matching on $G^{(t)}$ which is denoted with $S^{(t)}$. Let $\alpha(\mathrm{C}^{(t)})$ be the size of a maximum cardinality stable set $S^{(t)}$ on $C^{(t)}$. In case $\alpha(C^{(t)}) = \frac{|V(G)|}{2} - \left|I^{(t)}\right|$ holds then we again prune node t since we have a feasible solution for the $MWPMC^{(t)}$, i.e. a conflict free perfect matching, which consists of $I^{(t)} \cup S^{(t)}$. Here, this feasible solution gives us a chance to update the best known lower bound $\underline{z}$. After a lower bound is calculated we proceed to perform the upper bound computation. The determination of $\alpha(C^{(t)})$ requires the solution of the NP-hard MCSP at every node of the B&B tree. However, the use of exact value helps fathoming a larger number of nodes, which can balance the increase in the computational cost.

3.3 Upper Bound

At each node t, we compute an upper bound for the $MWPMC^{(t)}$ by solving the subproblem $MWMC^{(t)}$. In case the $MWMC^{(t)}$ has an optimum solution with weight $z^{(t)}$ then we have a maximum weight conflict free matching $M^{(t)}$ with value $z^{(t)}$. Hence, we determine an upper bound value at node t as $\overline{z}^{(t)}$=$z^{(t)} + \sum_{e \in I^{(t)}} c_e$. In case the $MWMC^{(t)}$ has no solution then we set $\overline{z}^{(t)} = -\infty$ in order to prune current node t. $MWMC^{(t)}$ is actually equivalent to the solution of the NP-hard MWSP on $C^{(t)}$. Hence, the determination of an upper bound at every node of the B&B tree has similar negative effect on the overall computational cost of the B&B algorithm. Again similarly, this can be balanced by the increasing ability to fathom more nodes because of the tightness of the upper bound.

3.4 Pruning

At this stage we either prune the current active node t or proceed to the division operation. Actually, we consider three cases. First, we check whether the current upper bound value is less than the best known lower bound value, i.e. $\overline{z}^{(t)} \leq \underline{z}$ holds. In such case the current active node is not taken into further consideration and fathomed. In the second case, the solution obtained in the upper bounding procedure, i.e. matching $M^{(t)}$ which is obtained by solving $MWMC^{(t)}$ on $G^{(t)}$, yields a conflict free perfect matching together with the edge subset $I^{(t)}$. Then, we have a chance to update the best known lower bound value. For the remaining case, we have at least one exposed vertex which will be considered in the division operation.

3.5 Branching Rule for Division

We perform branching operation considering the selected exposed vertex $v \in V(G^{(t)})$. Note that this operation does not necessarily outputs a dichotomized B&B tree. Recall that, at each node t of the B&B tree, $I^{(t)}$ and $X^{(t)}$ stand for the set of edges which must be included to and excluded from a conflict free perfect matching, respectively. Hence, given $M^{(t)}$ which is the maximum weight conflict free matching obtained by solving $MWMC^{(t)}$ on $G^{(t)}$ and $v \in V(G^{(t)})$ be an $M^{(t)}$ exposed vertex, we create $d^{(t)}$ =$d_{G^{(t)}}(v)$ new subproblems by enforcing one by one each edge e_i incident to v, i.e. $e_i \in \delta_{G^{(t)}}(v)$, to be in the solution. Therefore, we generate subproblems with the following characterizations:

$$\left\{\begin{array}{l} I^{(ti)} = I^{(t)} \cup \{e_i\} \\ X^{(ti)} = X^{(t)} \cup \delta_C(e_i) \\ E(G^{(ti)}) = E(G) \setminus \{I^{(ti)} \cup X^{(ti)}\} \end{array}\right\} \text{ for } i = 1, \ldots, d^{(t)} \qquad (6)$$

3.6 Stable Sets on Extended Conflict Graph

During the run of the B&B algorithm, we try to find a maximum cardinality conflict free matching and a maximum weight conflict free matching on $G^{(t)}$ in order to compute lower and upper bound values for the $MWPMC^{(t)}$, respectively. To compute a lower bound for the $MWPMC^{(t)}$, we solve the $MCSP$ on the extended conflict graph $C^{(t)}$ and find a stable set $S^{(t)}$ with the stability number $\alpha(C^{(t)})$. For that purpose, we solve the MCP on the complement of $C^{(t)}$, namely $\overline{C^{(t)}}$ by running the exact algorithm by Östergård [18]. On the other hand, to find an upper bound for the $MWPMC^{(t)}$ we solve the subproblem $MWMC^{(t)}$ by transforming it into an equivalent MWCP on $\overline{C^{(t)}}$. For that purpose, we employ the MWCP algorithm by Östergård [17].

4 COMPUTATIONAL EXPERIMENTS

We have performed the computational experiments in order to compare the performance of the proposed B&B algorithm with two BILP formulations solved by the state-of-the-art MILP solver CPLEX 12.7.0. All computations are performed on an HPE SRV DL380 GEN9 Server with a 2.20 GHz E5-2650v4 Processor and 192 GB RAM operating within Windows Server 2016 environment. To the best of our knowledge, there is no standard test library for MWMCP hence, we have generated random test instances.

4.1 Test Instances

In Table 1 we report the properties of the randomly generated instances. The first column includes the name of the instance sets where each of which contains 5 randomly generated test problems. The number following the letter "N" stands for the number of vertices of the corresponding instance set. Besides, a suffix is added to represent the density of the graph G. For example, a suffix of "H" is used to represent high edge density of the graph generated for the test instance.

In Table 1 configurations are presented in the columns two to six. The second column gives the number of vertices in G, i.e. $|V(G)|$, for each instance set. In our test bed, $|V(G)|$ changes from 36 to 58 vertices. The third column incorporates the number of edges in G, i.e. $|E(G)|$ which varies from 126 to 770 edges. The fourth column includes the number of conflicting edge pairs in G or equivalently the number of edges in the conflict graph C, i.e. $|E(C)|$. The fifth column stands for the edge density of G, i.e. $d(G)$, which is calculated as the number of edges in G divided by the maximum possible number of edges. We have employed three levels for edge density of graphs 0.2, 0.5 and 0.8 respectively for

Algorithm 1: Branch-and-bound algorithm for solving MWPMC using MWMC relaxations

```
Input: A graph $G = (V(G), E(G))$ edge weights $w_e \geq 0$,
  conflict graph $C = (V(C), E(C))$;
Output: A maximum weight conflict free perfect matching
  $M^* = (V(M^*), E(M^*))$
begin
(Initialization): Set $t = 0$, MWPMC$^{(0)} \longleftarrow$ MWPMC, G$^{(0)}$= $G$,
  C$^{(0)}$= $C$, $\mathcal{L}$ = {MWPMC$^{(0)}$}, $I^{(0)}$= $\emptyset$, $X^{(0)}$= $\emptyset$,$\underline{z} = -\infty$
(Termination test): If $\mathcal{L} = \emptyset$, then output $E(M^*)$ and stop.
(Lower bounding): Select and delete a problem from $\mathcal{L}$, say
  MWPMC$^{(t)}$,
if $\left|E(G^{(t)})\right| < \frac{|V(G)|}{2} - \left|I^{(t)}\right|$ then
   there is no conflict free perfect matching of G with edges
  in $I^{(t)} \cup E(G^{(t)})$. Set $\overline{z}^{(t)} = -\infty$, to prune MWPMC$^{(t)}$ and
  go to Pruning
else
    Find the maximum cardinality conflict free matching in
  $G^{(t)}$, which is $S^{(t)}$ and let its size be $\alpha(C^{(t)})$
    if $\alpha(C^{(t)}) < \frac{|V(G)|}{2} - \left|I^{(t)}\right|$ then
       there is no conflict free perfect matching of G with
       edges in $I^{(t)} \cup E(G^{(t)})$.
       Set $\overline{z}^{(t)} = -\infty$ to prune MWPMC$^{(t)}$ and go to Pruning
    else
      if $\alpha(^{(t)}) = \frac{|V(G)|}{2} - \left|I^{(t)}\right|$ then
         $I^{(t)} \cup S^{(t)}$ are the edges of a conflict free perfect
         matching;
         if $w(I^{(t)} \cup S^{(t)}) > \underline{z}$ then
            Update the lower bound and incumbent by setting
            $\underline{z} = w(I^{(t)} \cup S^{(t)})$, $E(M^*) \longleftarrow I^{(t)} \cup S^{(t)}$ and
            go to Upper bounding
         end if
      end if
    end if
end if
(Upper bounding): Solve MWMC$^{(t)}$ relaxation on G$^{(t)}$
if MWMC$^{(t)}$ has a solution then
    Let M$^{(t)}$ be a maximum weight conflict free matching
    on G$^{(t)}$ and z$^{(t)}$ be its optimal value.
    Set $\overline{z}^{(t)} = z^{(t)} + w(I^{(t)})$
else
    Set $\overline{z}^{(t)} = -\infty$
end if
(Pruning):
    i.  If $\overline{z}^{(t)} \leq \underline{z}$, then go to Termination test.
    ii.  If there is no M$^{(t)}$-exposed vertex in G$^{(t)}$ (i.e. M$^{(t)}$ is a
       perfect matching in G$^{(t)}$ and $I^{(t)} \cup E(M^{(t)})$ are the edges
       of a perfect matching of G) and $\overline{z}^{(t)} < \underline{z}$ then set $\overline{z}^{(t)} = \underline{z}$
       and $E(M^*) \longleftarrow I^{(t)} \cup E(M^{(t)})$ go to Termination test
    iii. If there is an $M^{(t)}$-exposed vertex in $G^{(t)}$(i.e. $M^{(t)}$ is not
       a perfect matching in $G^{(t)}$) then go to Division.
(Division): Select an $M^{(t)}$-exposed vertex v of $G^{(t)}$ and create
  $i = 1, 2, \ldots, d^{(t)}$ =$d_{G^{(t)}}(v)$ subproblems.
Let {MWPMC$^{(ti)}$} obtained from {MWPMC$^{(t)}$} by enforcing
edge $e_i$ to be in the perfect matching for $e_i \in \delta_{G^{(t)}}(v)$. Add
them to the active node list $\mathcal{L}$ with $\overline{z}^{(ti)}$=$\overline{z}^{(t)}$ for
$i = 1, 2, \ldots, d^{(t)}$ and go to Termination test
end
```

low (L), medium (M) and high (H) edge density. The last column is for the density of the conflict graph, namely $d(C)$.

Instance generation process of the MWPMC is not straightforward and hence must be carefully handled. For that purpose, an initial perfect matching is arbitrarily generated and it is kept to guarantee the feasibility of the MWPMC test instance. Therefore, $|V(G)/2|$ edges which correspond to the initial perfect matching and some more edges are randomly generated, summing up to $|E(G)|$ edges. The generation of edges are performed such that each vertex has a degree of at least two in order to avoid trivial solutions. Besides, $|E(C)|$ conflicting edge pairs are randomly selected among possible edge pairs excluding the edge pairs of the initial perfect matching. Finally, the edge weights w_e are randomly generated such that $w_e \in [10, 900]$ is satisfied. The process is repeated for each instance and we have generated a total of 110 test instances in 22 sets reported in Table 1.

4.2 Computational Results

In Table 2 we present the results obtained with the proposed B&B algorithm. Table 3 includes the results output by the solution of STRONG and WEAK formulations with the CPLEX MILP solver. All experiments are performed with a CPU time limit of 600 secs.

In Table 2 and Table 3, the rows correspond to the average values for the instance sets. In the last rows, the overall averages of the corresponding columns are given. The **act** column includes the average number of active nodes remaining in the B&B node list $\mathcal{L}$ when the B&B algorithm stops. The **LB** and **UB** columns incorporate the average lower and upper bound values output by the B&B algorithm, respectively. In the **CPU(s)** column we provide the average CPU time required in seconds. The rightmost column, i.e. column **exp**, reports the average number of explored nodes during the run of the B&B algorithm.

Table 3 introduces the results obtained with the solution of the BILP formulations via CPLEX MILP solver with default options with a CPU time limit of 600 secs. The last row denotes the overall average values of the corresponding columns. The values under **Bound** columns are the average solution values obtained with CPLEX MILP solver. The columns **CPU(s)** are for the average CPU times is seconds required by the CPLEX MILP solver.

Considering the results reported with Table 2 and Table 3, we can observe that the B&B algorithm is more efficient than solving the BILP formulations via CPLEX MILP solver. Observe that, the overall average CPU time requirement of the B&B algorithm is 34.76 secs. compared to the ones by STRONG and WEAK formulations which are 97.12 secs. and 61.20 secs., respectively. Furthermore, we should state that, all instances except the ones in the set `N56-M` are solved to optimality by both the B&B algorithm and CPLEX MILP solver within the CPU time limit of 600 secs.

Last but not least, we should report that, the B&B algorithm could not yield the optimum in only 2 instances in the set `N56-M`. On the other hand, the *STRONG* and *WEAK* formulations solved with CPLEX MILP solver could not output the optimum in 3 and 2 cases in the set `N56-M` within the CPU time limit, respectively.

5 CONCLUDING REMARKS AND DISCUSSION

We have proposed an exact solution procedure and two mathematical programming formulations for the Maximum Weight Matching Problem with Conflicting Edge Pairs (MWPMC). Considering our preliminary computational experiments on randomly generated test instances we can state that the proposed

Table 1: Instance properties

	$\|V(G)\|$	$\|E(G)\|$	$\|E(C)\|$	d(G)	d(C)
N36-H	36	504	80000	0.8	0.74
N36-L	36	126	5000	0.2	0.73
N36-M	36	315	30000	0.5	0.71
N38-H	38	563	90000	0.8	0.67
N38-L	38	141	7500	0.2	0.85
N38-M	38	352	40000	0.5	0.75
N42-H	42	689	110000	0.8	0.56
N42-L	42	173	12000	0.2	0.89
N42-M	42	431	60000	0.5	0.74
N44-L	44	190	14000	0.2	0.86
N44-M	44	473	70000	0.5	0.71
N46-L	46	208	16000	0.2	0.82
N46-M	46	518	80000	0.5	0.68
N48-L	48	226	18000	0.2	0.78
N48-M	48	564	73800	0.5	0.65
N52-L	52	266	22000	0.2	0.70
N52-M	52	663	104000	0.5	0.55
N54-L	54	287	24000	0.2	0.65
N54-M	54	716	108000	0.5	0.49
N56-L	56	309	26000	0.2	0.61
N56-M	56	770	112000	0.5	0.45
N58-L	58	331	28000	0.2	0.58
Average	**45.73**	**400.68**	**51377.27**	**0.40**	**0.69**

Table 2: Performance of the B&B Algorithm

	act	LB	UB	CPU (s)	exp
N36-H	0	537.2	537.2	1.5	1043.4
N36-L	0	485.2	485.2	0.0	34.2
N36-M	0	560.2	560.2	0.3	436.6
N38-H	0	815.6	815.6	6.7	3608.8
N38-L	0	851.2	851.2	0.0	15.2
N38-M	0	455.8	455.8	0.2	335.8
N42-H	0	1032.4	1032.4	129.8	48724.8
N42-L	0	631.4	631.4	0.0	8.8
N42-M	0	491.6	491.6	0.5	442.4
N44-L	0	981.4	981.4	0.0	12
N44-M	0	932.6	932.6	0.7	557.2
N46-L	0	1025.6	1025.6	0.0	23.2
N46-M	0	840.4	840.4	1.7	1101.4
N48-L	0	760.2	760.2	0.0	31.8
N48-M	0	991.0	991.0	2.0	1102.8
N52-L	0	906.2	906.2	0.0	99.6
N52-M	0	1041.8	1041.8	38.7	15486.8
N54-L	0	874.6	874.6	0.1	137.8
N54-M	0	1051.4	1051.4	88.4	28790
N56-L	0	945.6	945.6	0.1	242.2
N56-M	16.4	1092.6	1096.8	491.5	130753
N58-L	0	1095.0	1095	0.2	293.4
Average	**0.75**	**836.32**	**836.51**	**34.67**	**10603.67**

B&B algorithm outperforms the MILP solver. It should be borne in mind that in its current form of the B&B algorithm two NP-hard problems have to be solved at every search node, which can be bound to pay a very high computational price with the increase of instance size and conflict density. However, it can be possible to compute upper bounds to both $\alpha(C^{(t)})$ and the optimum value of $MWMC^{(t)}$ using heuristics and further relaxations with considerably lower costs, which remains as a part of our future investigations. Furthermore, efficient heuristics and as well as meta-heuristics for the solution of MWPMC can be another fertile research avenue.

ACKNOWLEDGMENTS

This work is supported by the Galatasaray University Scientific Research Project (Grant No: 18.402.009) and the Scientific and Technical Research Council of Turkey - TÜBİTAK (Grant No: 217M477)

Table 3: Performance of the BILP Formulations

	STRONG		WEAK	
	Bound	**Cpu (s)**	**Bound**	**Cpu (s)**
N36-H	537.2	7.8	537.2	17.9
N36-L	485.2	0.3	485.2	0.2
N36-M	560.2	1.3	560.2	1.5
N38-H	815.6	77.5	815.6	174.5
N38-L	851.2	0.3	851.2	0.3
N38-M	455.8	2.5	455.8	1.9
N42-H	1032.4	363.0	1032.4	293.3
N42-L	631.4	0.5	631.4	0.3
N42-M	491.6	2.7	491.6	6.1
N44-L	981.4	0.7	981.4	0.5
N44-M	932.6	5.0	932.6	8.4
N46-L	1025.6	1.1	1025.6	2.2
N46-M	840.4	35.5	840.4	9.4
N48-L	760.2	1.4	760.2	0.5
N48-M	978.6	159.3	978.6	15.1
N52-L	906.2	1.6	906.2	1.1
N52-M	1041.8	316.0	1041.8	190.1
N54-L	874.6	1.7	874.6	1.1
N54-M	1051.4	572.8	1051.4	238.0
N56-L	945.6	1.7	945.6	1.4
N56-M	974.2	581.7	1092.6	381.2
N58-L	1095.0	2.1	1095.0	1.3
Average	**830.37**	**97.12**	**835.75**	**61.20**

REFERENCES

[1] Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. 1993. *Network Flows: Theory, Algorithms and Applications.* Prentice-Hall, Englewood Cliffs, N.J.

[2] İ. Kuban Altınel, Necati Aras, Zeynep Şuvak, and Z. Caner Taşkın. 2018. Minimum Cost Non-crossing Flow Problem on Layered Networks. *Discrete Appl. Math.* (2018). https://doi.org/10.1016/j.dam.2018.09.016

[3] Mariem Ben Salem, Raouia Taktak, A. Ridha Mahjoub, and Hanene Ben-Abdallah. 2018. Optimization Algorithms for the Disjunctively Constrained Knapsack Problem. *Soft Comput.* 22, 6 (March 2018), 2025–2043. https://doi.

org/10.1007/s00500-016-2465-7

[4] Andrea Bettinelli, Valentina Cacchiani, and Enrico Malaguti. 2017. A Branch-and-Bound Algorithm for the Knapsack Problem with Conflict Graph. *INFORMS J. Comput.* 29, 3 (2017), 457–473. https://doi.org/10.1287/ijoc.2016.0742

[5] Renatha Capua, Yuri Frota, and Luiz Satoru Ochi. 2018. A Study on Exponential-size Neighborhoods for the Bin Packing Problem with Conflicts. *J. Heuristics* 24, 4 (August 2018), 667–695. https://doi.org/10.1007/s10732-018-9372-2

[6] Francesco Carrabs, Raffaele Cerulli, Rosa Pentangelo, and Andrea Raiconi. 2018. Minimum Spanning Tree with Conflicting Edge Pairs: a Branch-and-Cut Approach. *Ann. Oper. Res.* (2018). https://doi.org/10.1007/s10479-018-2895-y

[7] Andreas Darmann, Ulrich Pferschy, Joachim Schauer, and Gerhard J. Woeginger. 2011. Path, Trees and Matchings under Disjunctive Constraints. *Discrete Appl. Math.* 159, 16 (September 2011), 1726–1735. https://doi.org/10.1016/j.dam.2010.12.016

[8] Ran Duan and Seth Pettie. 2014. Linear-Time Approximation for Maximum Weight Matching. *J. Assoc. Comput. Mach.* 61, 1 (January 2014), 1–23. https://doi.org/10.1145/2529989

[9] Jack Edmonds. 1965. Maximum Matching and a Polyhedron with 0,1-Vertices. *J. Res. Natl. Bur. Stand.* 69B, 1 and 2 (1965), 125–130.

[10] Annette M. Ficker, Frits C.R. Spieksma, and Gerhard J. Woeginger. 2018. Transportation Problem with Conflicts. *Ann. Oper. Res.* (2018), 1–21. https://doi.org/10.1007/s10479-018-3004-y

[11] Michael R. Garey and David S. Johnson. 1979. *Computers and Intractability: a Guide to the Theory of NP-Completeness.* W. H. Freeman, New York.

[12] Michel Gendreau, Gilbert Laporte, and Frederic Semet. 2004. Heuristics and Lower Bounds for the Bin Packing Problem with Conflicts. *Comput. Oper. Res.* 31, 3 (March 2004), 347–358. https://doi.org/10.1016/S0305-0548(02)00195-8

[13] Dries Goossens and Frits C.R. Sieksma. 2009. The Transportation Problem with Exclusionary Side Constraints. *4OR-Q. J. Oper. Res.* 7, 1 (March 2009), 51–60. https://doi.org/10.1007/s10288-007-0067-z

[14] Chien-Chung Huang and Telikepalli Kavitha. 2017. New Algorithms for Maximum Weight Matching and a Decomposition Theorem. *Math. Oper. Res.* 42, 2 (May 2017), 411–426. https://doi.org/10.1287/moor.2016.0806

[15] Temel Öncan and İ. Kuban Altınel. 2018. A Branch-and-Bound Algorithm for the Minimum Cost Bipartite Perfect Matching Problem with Conflict Pair Constraints. *Electron. Notes Discrete Math.* 64 (February 2018), 5–14. https://doi.org/10.1016/j.endm.2018.01.002

[16] Temel Öncan, Ruonan Zhang, and Abraham P. Punnen. 2013. The Minimum Cost Perfect Matching Problem with Conflict Pair Constraints. *Comput. Oper. Res.* 40, 4 (April 2013), 920–930. https://doi.org/10.1016/j.cor.2012.10.022

[17] Patric R.J. Östergård. 1999. A New Algorithm for the Maximum-Weight Clique Problem. *Electron. Notes Discrete Math.* 3 (May 1999), 153–156. https://doi.org/10.1016/S1571-0653(05)80045-9

[18] Patric R.J. Östergård. 2002. A Fast Algorithm for the Maximum Clique Problem. *Discrete Appl. Math.* 120 (August 2002), 197–207. https://doi.org/10.1016/S0166-218X(01)00290-6

[19] Ulrich Pferschy and Joachim Schauer. 2009. The Knapsack Problem with Conflict Graphs. *J. Graph Algorithms Appl.* 13, 2 (2009), 233–249. https://doi.org/10.7155/jgaa.00186

[20] Ulrich Pferschy and Joachim Schauer. 2013. The Maximum Flow Problem with Disjunctive Constraints. *J. Comb. Optim.* 26, 1 (July 2013), 109–119. https://doi.org/10.1007/s10878-011-9438-7

[21] Ruslan Sadykov and François Vanderbeck. 2013. Bin Packing with Conflicts: a Generic Branch-and-Price Algorithm. *INFORMS J. Comput.* 25, 2 (2013), 244–255. https://doi.org/10.1287/ijoc.1120.0499

[22] Phillippe Samer and Sebastian Urrutia. 2015. A Branch and Cut Algorithm for Minimum Spanning Trees under Conflict Constraints. *Optim. Lett.* 9, 1 (January 2015), 41–55. https://doi.org/10.1007/s11590-014-0750-x

[23] Minghe Sun. 2002. The Transportation Problem with Exclusionary Side Constraints and Two Branch-and-Bound Algorithms. *Eur. J. Oper. Res.* 140, 3 (August 2002), 629–647. https://doi.org/10.1016/S0377-2217(01)00239-9

[24] Ruonan Zhang, Santosh N. Kabadi, and Abraham P. Punnen. 2011. The Minimum Spanning Tree Problem with Conflict Constraints and its Variations. *Discrete Optim.* 8, 2 (May 2011), 191–205. https://doi.org/10.1016/j.disopt.2010.08.001

Minimum-Cost Virtual Network Function Resilience

Yannick Carlinet
Orange Labs Networks
Chatillon, France
yannick.carlinet@orange.com

Nancy Perrot
Orange Labs Networks
Chatillon, France
nancy.perrot@orange.com

Anderson Alves-Tzitas
Univ. Federal de Minas Gerais
Brazil
andersontzitas02@gmail.com

ABSTRACT

In the future 5G networks, a wide range of new services with strong requirements will be delivered in the form of chains of service functions on independent virtual networks. These virtual networks will be deployed on demand, each one adapted to the specific service requirements. For infrastructure providers a real challenge consists in providing and setting up the required virtual networks (network slices) while guaranteeing strict Service Level Agreements. One of the major stakes is to be able to provide failure protection for the service function chains at minimal cost. In this work, we consider a set of deployed service chains, and we study the best strategy to protect them at minimal cost. We propose mathematical formulations that provide optimal backup functions placement over a network, and the associated backup paths for each VNF of all the chains. We develop an efficient ILP-based heuristic relying on a separation of the problem into smaller ones to solve large scale instances. We show that our heuristic is competitive, both regarding the solution quality and the solving time.

1 INTRODUCTION

1.1 Telecom context

Network operators face the challenge of making their network more flexible and cost-effective. They also have to plan the evolution of their networks for the incoming 5G networks. Indeed, some 5G services, such as the Ultra-Reliable Low Latency (URLL), require the network functions to be executed as close as possible to the end-user. This means that the operators will have to split and distribute the network functions over multiple network nodes. Thanks to the maturity of virtualization technologies, Virtual Network Functions (VNF) have the capability to run inside Virtual Machines (VM) or containers on commodity hardware. In addition, the concept of Network Slicing will bring even more flexibility, as it allows several virtual networks to run on a unique physical infrastructure, provided by one or several infrastructure providers. This flexibility brings the following benefits:

- *On-Demand Network.* Network Functions can be deployed and updated remotely, as opposed to manually installing and plugging a hardware equipment to the network. Network capacity can also be scaled down or up on-the-fly, depending on the current demand.
- *Cost-effectiveness.* Maintenance and exploitation of the physical infrastructure are mutualized between the service providers.
- *Automatization.* Operations on software are easily automatized, enabling scaling, healing, reduced delays to market new services, among others.

A Network Slice could be seen as a set of VNFs that are organized into Service Function Chains (SFC), that specifies the sequence of VNFs crossed by the traffic for a specific service. Figure 1 shows an example of a Service Function Chain taken from a use-case defined at IETF (Internet Engineering Task Force) [10]. The provided service is video optimization and it consists of three basic Virtual Network Functions: the Steering Proxy, which is able to redirect HTTP traffic, a DPI-based (Deep Packet Inspector based) controller, which checks for video traffic, and an optimizer, which transcodes the video into a suitable format for the user terminal.

```
[Steering Proxy]---[DPI Contr.]---[Optimizer]
```

Figure 1: Example of Service Function Chain (SFC).

However, this shift in the networking paradigm comes with many technological obstacles to overcome. One of them concerns the resilience of the slices and the associated services.

1.2 Problem Statement

To operate a service, the VNFs of the service chains must be installed in some network nodes, and the traffic routed through the installed VNFs. Some of the VNFs could possibly be shared among several different chains to reduce the exploitation costs, while the capacity limitations and security requirements are met. Figure 2 illustrates a small example of two different SFCs to be routed between an origin-destination pair $(1, 6)$. SFC1 (respectively SFC2), in red color (resp. blue color), corresponds to a service that requires the functions VNF1 and VNF3 (resp. VNF1, VNF2 and VNF4). A SFC routing and VNF placement solution is illustrated. VNF1 is installed in node 2 and shared by both SFCs.

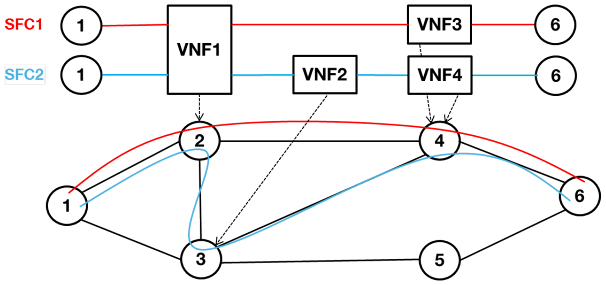


Figure 2: Example of deployed Service chains

For a commercially exploited network, having a resiliency scheme for each failure scenario is mandatory. Figure 3 illustrates a rerouting scenario of the previous example in case of a failure of the node 2, assuming that a backup of VNF1 has been previously installed on node 3. In this example both SFCs are to be rerouted through the backup node 3.

The aim of this paper is to answer the question of how to make the Service Function Chains resilient to node and link failures. More specifically, the goal is to protect all the Service Function Chains already deployed against a single node or link failure, at minimum cost. It means that all the Virtual Network Functions

ISBN 978-3-89318-079-0 on OpenProceedings, ISSN 2510-7437

DOI: 10.5441/002/inoc.2019.08

that compose the Service Chains must be protected. We assume that a backup VNF can be used as a backup VNF for several nominal VNFs. Similarly, nominal VNFs with spare capacity can be used as backup VNF. The new routes to the backup VNFs must be disjoint from the corresponding nominal route, to protect from link failure, while satisfying the same latency and capacity constraints as for the nominal path.

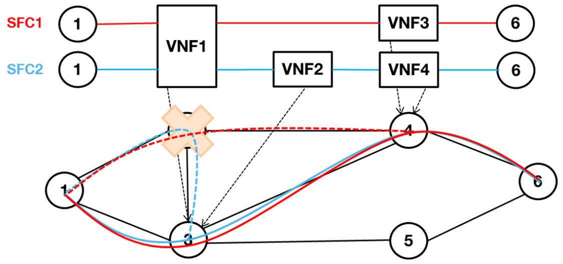


Figure 3: Example of node and routing back-up

Considering routing and placement of nominal and backup SFCs jointly would yield better solutions, as it would allow a better usage of network resources. However, in operational settings, realistic scenarios are likely to consider a two-phase deployment.

1.3 State of the Art

In [1], Allybokus et al. consider the problem of VNF placement for the composition of Service Function Chains. Resiliency is taken into account to a certain extent with the addition of anti-affinity rules in the model.

The ETSI (European Telecommunications Standards Institute) has studied the issue of reliability in Network Function Virtualization and has given a comprehensive list of features and models for end-to-end reliability [4]. In [6], Hmaity et al. solve the problem of the placement of primary VNFs and backup VNFs, so that the service chains are protected against node and/or link failure. In the end-to-end protection scheme, the backup path does not use any link or node from the primary path. In this scheme, the service chains are resilient against the failure of all the links and all the nodes. In [11], Qu et al. aim at finding the best service chain embedding and routing, for minimizing the overall network bandwidth consumption. The problem considers reliability of the service chains with constraints to guarantee that there are enough backup VNF for a given target reliability. In [8], Kong et al. determine the number of VNF replicas required to guarantee the availability of the SFC, and place these replicas on the routing path. In [3], Engelmann and Jukan study the reliability of SFCs with flow parallelism. They evaluate the number of backup VNF necessary to reach a certain service reliability. They also study various placement strategy for the backup VNF. In [13], Wang and Doucette present an algorithm that aims at maximizing the network availability with the approach offered by Shared-Backup Path Protection (SBPP).

We observe that in the cited work, the backup path is disjoint from the working path. In practice, this level of resilience is often too costly and hardly needed since the event of all nodes and all links failing at the same time should seldom occur (if at all). The practical approach for network resiliency is to protect the network operations from a certain number of simultaneous failures. In this paper, we propose and solve a problem that is more relevant to the practical cases of network resiliency design. In addition, our approach is to help network architects to design a resilient network, in a cost-efficient manner. This is why we seek to minimize the cost of deployment, by contrast to the cited works.

2 THE VIRTUAL NETWORK FUNCTION RESILIENCE PROBLEM (VNFR)

2.1 Problem Definition

The network is represented by a directed graph $G(V, A)$, where V is the set of network nodes and A the set of arcs. Each network node u corresponds to a site in which a network function could be executed, either by running it on existing server or by opening a new site. Opening a new site has a high cost m_u and corresponds to setting up a new small DC on an eligible site. We assume that the nominal service chains are already deployed in the network. This assumption corresponds to many planned real use-case, in particular when protecting service chains related to critical virtualized functions like vEPC (virtual Evolved Packet Core, [9]) or vRAN (virtual Radio Access Network, [2]). The nominal functions already in place in the network can be shared and used to protect other service chains, and the residual capacity $C_u \in \mathbb{N}$ of used servers can be used to install new functions.

Each arc (uv) of G is characterized by a weight l_{uv} that is a function of the transmission delay between nodes u and v, by a maximal bandwidth capacity B_{uv}, and by a unitary usage cost e_{uv}.

The set of all the VNFs to be protected against failure is denoted by F. Each virtual function is characterized by a type $f \in F$, and a maximal flow rate t_f the function is able to process. The placement cost of a new function of type f in a node u is denoted h_u^f.

A service chain $k \in K$ is composed of an ordered set of virtual network functions, and an associated routing path P_k. It is characterized by a traffic demand between origin-destination nodes, named respectively o_k and d_k. Without loss of generality, we decompose each service chain k into micro chains $i, i = 1, \ldots, |F^k|$ composed of *one* function each, of type $f_k^i \in F$.

Then, from now, a *section* (i, k) corresponds to a unique function between an origin node o_k^i and a destination node d_k^i along the routing path P_k of the global service chain. Then, o_k^i is the node where f_k^{i-1} is placed if $i \geq 2$, o_k otherwise, and d_k^i the node where f_k^{i+1} is placed if $i \leq n - 1$, d_k otherwise. The paths of these micro chains are sections S_k^i of the path P_k. An example of nominal service chain path is given in Figure 4. The represented SFC is routed along a path $P1$ from the source node o_1 to the destination one d_1, and the flow runs 3 functions f_1^i, with $i = 1, 2, 3$. The three corresponding sections S_1^i, with $i = 1, 2, 3$ are respectively represented in figures 5, 6 and 7.

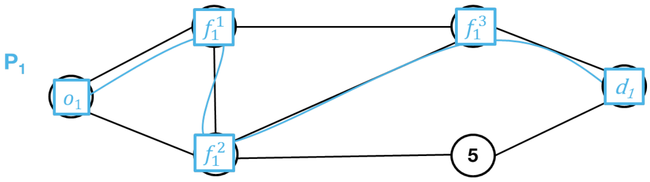


Figure 4: A service function chain path P_1, composed of 3 functions

The service chain sections deployed in the network are defined by two set of parameters:

- p^{ikf}_u that has value 1 is a VNF of type f in section i of service chain k is installed on u, 0 otherwise, and
- q^{ik}_{uv} whose value is 1 if the traffic in section i of service chain k is routed along arc (uv), 0 otherwise.

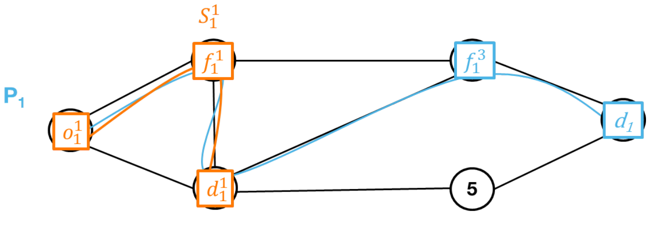


Figure 5: First section of Path P_1, S^1_1

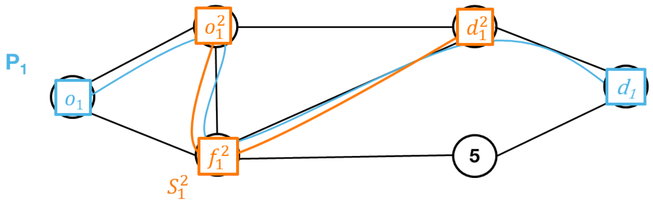


Figure 6: Second section of Path P_1, S^2_1

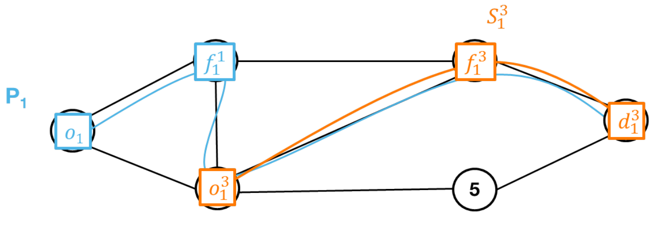


Figure 7: Third section of Path P_1, S^3_1

All the notations previously introduced in this section are summarized in Table 1.

2.2 Problem Formulation

The Virtual Network Function Resilience (VNFR) Problem can be formulated as an integer linear program. The four types of decision variables are :

- Routing variables

$$r^{ik}_{uv} = \begin{cases} 1 \text{ if } (u,v) \text{ is used to re-route section } i, k \\ 0 \text{ otherwise} \end{cases}$$

- Function placement

$$y^{ikf}_u = \begin{cases} 1 \text{ if backup function of } f \text{ for section } i \text{ of } k \text{ is} \\ \quad \text{placed on node } u \\ 0 \text{ otherwise} \end{cases}$$

- Number of instances of a same type of the backup function f installed on node u

$$x^f_u \in \mathbb{N}$$

- Opening of a new site

$$z_u = \begin{cases} 1 \text{ if the node } u \text{ is newly used to install at least} \\ \quad \text{one back up function} \\ 0 \text{ otherwise} \end{cases}$$

Then, a compact formulation for the problem is as follows :

$$\min \sum_{u\in V}\sum_{f\in F} h^f_u x^f_u + \sum_{u\in V\setminus N} m_u z_u + \sum_{(uv)\in A}\sum_{k\in K}\sum_{i\in S_k} e_{uv} b_k r^{ik}_{uv} \tag{1}$$

subject to

$$\sum_{(uv)\in A} r^{ik}_{uv} - \sum_{(v'u)\in A} r^{ik}_{v'u} = \begin{cases} 1 & \text{if } u = o^i_k \\ -1 & \text{if } u = d^i_k \\ 0 & \text{otherwise} \end{cases} \quad \begin{array}{l} \forall i \in S_k, \forall k \in K, \\ \forall u \in V. \end{array} \tag{2}$$

$$\sum_{k\in K}\sum_{i\in S_k} r^{ik}_{uv} b_k \le B_{uv} \qquad \forall (u,v) \in A \tag{3}$$

$$\sum_{f\in F} x^f_u \le C_u \qquad \forall u \in V \tag{4}$$

$$\sum_{(uv)\in A} r^{ik}_{uv} l_{uv} \le L^i_k \qquad \forall k \in K, \forall i \in S_k \tag{5}$$

$$\sum_{\substack{k\in K\\ i\in S_k}} \left[b_k y^{ikf}_u - \sum_{\substack{k'\in K\setminus k\\ i\in S_{k'}}} p^{ik'f}_u (t_f - b_{k'}) \right] \le t_f x^f_u \quad \forall u \in V, \forall f \in F \tag{6}$$

$$p^{ikf}_u \le \sum_{u'\in V\setminus u} y^{ikf}_{u'} \qquad \forall u \in V, \forall k \in K, \forall f \in F, \forall i \in S_k \tag{7}$$

$$r^{ik}_{uv} + q^{ik}_{uv} \le 1 \qquad \forall k \in K, \forall (u,v) \in A, \forall i \in S_k \tag{8}$$

$$\sum_{f\in F} x^f_u \le C_u z_u \qquad \forall u \in V\setminus N \tag{9}$$

$$y^{ikf}_u \le \sum_{(uv)\in A} \left(r^{ik}_{uv} + r^{ik}_{vu} \right) \qquad \forall u \in V, \forall k \in K, \forall f \in F, \forall i \in S_k \tag{10}$$

$$r^{ik}_{uv} \in \{0,1\} \qquad \forall k \in K, \forall (u,v) \in A, \forall i \in S_k$$

$$y^{ikf}_u \in \{0,1\} \qquad \forall f \in F, \forall k \in K, \forall u \in V, \forall i \in S_k$$

$$x^f_u \in \mathbb{N} \qquad \forall f \in F, \forall u \in V$$

$$z_u \in \{0,1\} \qquad \forall u \in V$$

The objective (1) of the problem consists in minimizing the whole cost of protecting the service chains against a single failure. The first term refers to the installation cost of a function f in a site u, the second one to the cost of opening a new site u, and finally the last one refers to the cost of re-routing the traffic of section i, of demand k, on the arc (u,v).

The first set of constraints (2) are the flow conservation constraints, to ensure the flow continuity on the back up routes for all the demands on all the network nodes. The constraints (3) are to ensure that the sum of the flows routed on each arc are less or equal to the bandwidth limit capacity. The constraints (4) are the node capacity constraints and (5) the latency constraints all along the backup routes of all the demands. The maximal flow rates that can be processed by a function f on a node u is expressed in (6) : the capacity of a function being the capacity of the new installed function instances in addition to the residual capacity of the functions f already installed for the nominal chains.

The placement constraints (7) are to ensure that each nominal function f has a back up function installed on another node, while constraints (8) are to ensure the disjunction between nominal and backup paths. The constraints (9) are to define the opening of a site : if at least one function f is installed on a site $u \in V\setminus N$, ie a site where no function has been installed yet. Finally, constraints (10) are to link the backup functions to the backup routes for all the functions.

2.3 A variant with protection sharing

A shared protection model can easily be obtained from this model, considering that just one VNF failure can occur at a given time, in the same geographical area. The shared protection consists

Table 1: Indexes, Sets, Constants, and Variables

Indexes	
f	Virtual Network Function types
u	Nodes
k	Service Function Chains
i	Sections of a service chain
(u, v)	Arcs
Sets	
V	Set of nodes
A	Set of arcs
F	Set of VNF types
K	Set of service chains to protect
S	Set of all sections of all chains
N	Set of open nodes i.e. hosting at least one VNF
Constants	
Graph notation	
C_u	Max number of VNF that can be hosted on u
m_u	Cost of opening node u
l_{uv}	Latency of arc (u, v)
B_{uv}	Bandwidth capacity of arc (u, v)
e_{uv}	Cost per traffic unit on arc (u, v)
Service chains	
P_k	Nominal path of service chain k
o_k	Source node for service chain k
d_k	Destination node service chain k
b_k	Bandwidth necessary for service chain k
F_k	Set of VNFs in service chain k
S_k	the set of sections in path P_k
t_f	Maximum rate of f
h_u^f	Cost of installing f in node u
Service chain sections	
s_k^i	Source node of the section i of service chain k
S^f	The set of sections containing function f
d_k^i	Destination node of the section i of service chain k
n_k^i	Backup node for backup section i of service chain k
f_k^i	VNF type in the section i of service chain k
L_k^i	Max. latency for section i of service chain k
p_u^{ikf}	Binary indicator of nominal VNFs placement
q_{uv}^{ik}	Binary indicator of nominal arc usage

in reserving, in each node, a free capacity space to run only the VNF with the largest capacity requirement, so that any single VNF could be run.

Thus, the capacity constraint Equation (6) becomes :

$$\max_{\substack{k \in K \\ i \in S_k}}(b_k y_u^{ikf}) \leq t_f x_u^f + \sum_{\substack{k \in K \\ i \in S_k}} \sum_{\substack{k' \in K \setminus k \\ i \in S_{k'}}} p_u^{ik'f}(t_f - b_{k'})$$
$$\forall u \in V, \forall f \in F$$

And the bandwidth constraint (3) becomes :

$$\max_{\substack{k \in K \\ i \in S_k}}(b_k \, r_{uv}^{ik}) \leq B_{uv} \quad \forall (u, v) \in A$$

This alternate formulation is referred to as PS-VNFR (Protection Sharing VNFR) in the following.

Table 2: Compact formulation

$\lvert V \rvert$	$\lvert A \rvert$	$\lvert F \rvert$	$\lvert S \rvert$	Time (s)	Var.	Constr.
30	198	20	400	1820	319,830	892,258
50	538	40	500	>3600	1,271,050	3,568,138
100	2030	40	400	>3600	2,416,100	6,474,630

2.4 Problem Formulation Analysis

These formulations have been solved to optimality using the commercial solver Cplex 12 ([7]). The largest generated instances, described in section 4.1, couldn't be solved within one hour. An illustration of the size of the formulation (number of variables and constraints) of VNFR is given in Table 2; where $|V|$ and $|A|$ refers to the graph size, $|F|$ to the number of function types, and $|S|$ is the number VNF instances, i.e. the number of chain sections to be protected against failure. From the formulation of the problem, we can compute the number of variables as

$$|S||A| + |S||F||V| + |F||V| + |V| \tag{11}$$

and the number of constraints as

$$|A| + 2|V| + |S| + |V||S| + 2|F||V| + 2|S||A| + 3|V||S||F| \tag{12}$$

We can observe from Equation (11) that for a given graph (i.e. with given V and A), the factor for the number of function types is $|V|$ and the factor for the number of sections is $|A|$. In typical production networks, the number of arcs is much greater than the number of nodes, i.e. $|V| \ll |A|$. The same reasoning applies for the number of constraints, in Equation 12. Therefore, we conclude that the size of the formulation is particularly sensitive to the number of sections, for a given graph.

3 ILP-BASED HEURISTIC

For some instances, solving the ILP (Integer Linear Program) to optimality might not be feasible in a reasonable time. For this reason, we have designed a heuristic called DC-VNFR (Divide and Conquer in the VNFR problem). This heuristic is based on the principle of dividing the original problem into a set of smaller problems. The approach is to process the backups for each type of VNF, one type at a time. More specifically, the set of all the sections S is divided into subsets S^f such that S^f contains all the sections that contain a VNF of type f.

The VNFR problem is then solved for each S^f separately. The order of resolution is by decreasing λ_f, with $\lambda_f = \sum_{(i,k) \in S^f} b_k$. In other words, we determine the backups for the type of VNF with the largest traffic first.

The heuristic DC-VNFR is detailed in Algorithm 1.

As the heuristic consists in fixing a set of variables at each iteration, it may not find a feasible solution for some instances. In that case, the algorithm stores the rejected backups in a set R and carry on the processing. However, over all our numerical experiments the heuristic has always terminated with a feasible solution.

4 RESULTS

4.1 Random Instances Creation

In order to evaluate the performance of our model, we have designed a random instance generator, that allows us to generate arbitrarily large instances. It is implemented with python3 and the networkX package [12].

Algorithm 1: The DC-VNFR heuristic

```
Input  : An instance of the VNFR problem (cf. 2), with
         $S = \bigcup_{f=1}^{|F|} S_f$, with $S^f$ the set of all sections with
         function type $f$
Output : A solution to the VNFR problem (i.e. $\bar{x}_u^f$, $\bar{y}_u^{ikf}$, $\bar{z}_u$,
         $\bar{r}_{uv}^{ik}$), the total cost $ct$, and the set of rejected
         demands $R$
Compute all the $\lambda_f = \sum_{(i,k) \in S^f} b_k$ for $f \in F$
Sort the sets $S^f$ by decreasing $\lambda_f$
for $f \leftarrow 1$ to $|F|$ do
    Solve VNFR to optimality with $S^f$ instead of $S$
    if there is a solution then
        Update variables $\bar{x}_u^f$, $\bar{y}_u^{kf}$, $\bar{z}_u$, $\bar{r}_{uv}^k$
        Update total cost $ct$
        Update capacities of arcs and nodes
        for all $u$ such that $z_u = 1$ do
            $N \leftarrow N \cup \{u\}$
        end
    else
        $R \leftarrow R \cup \{S^f\}$
    end
end
```

The inputs for the random instance generator are the number of nodes in the graph, the number of types of VNF and the number of sections. The other parameters are randomly and uniformly chosen in-between an input interval given in Table 3.

The arcs are added as follows. First a path that connects all the nodes is added, so as to ensure that the graph is connected. Then, arcs are chosen randomly and added until a certain percentage of the maximum number of arcs in the graph is reached. In the following, we name the graphs that have 20% of the maximum number of arcs *type A* and the graphs at 80% *type B* (cf. Table 3). These values were selected in order to assess the impact of the graph density on performance.

Table 3: Parameters for random instances

Intervals	
C_u	[1, 3]
l_{uv}	[1, 20]
B_{uv}	[100, 500]
t_f	[1000, 2000]
b_k	[1, 20]
L_k	[100, 500]
e_{uv}	[1, 5]
h_u^f	[10, 100]
m_u	[100, 300]
Graphs	
type A	20% complete
type B	80% complete

The nominal Service Function Chains are placed as follows: first the origin and destination nodes of each section are randomly selected, then a node is randomly selected on the shortest path to host the VNF. The type of the VNF is also randomly selected.

4.2 Numerical Results

We have generated random instances with varying numbers of nodes and varying number of sections. For a given set of input parameters, we have generated 10 random instances. Then, we have solved all the instances, first with the ILPs (Integer Linear Programs) described in section 2 and then with the DC-VNFR heuristic described in section 3. The ILPs were implemented in Python3 with the Pyomo library [5] and the heuristic was also developed with Python3. We have recorded the execution time in terms of CPU time, and recorded the relative gap between the optimal solution and the solution given by DC-VNFR.

First, it is interesting to note that the two ILPs presented in section 2 yield to the same results (both in terms of objective function and execution time). Therefore, in the following we refer simply to the optimal solution as the ILP. This is due to the fact that, in the random instances we have generated, the parameter t_f (maximum rate processed by a function) was high enough not to need to instantiate two of them in a node. In addition, the capacity of the links were not saturated. In consequence, since both constraints (3) and (6) were not saturated, there is no difference with the model that relaxes them. We could check that, on the instances where one or both these constraints were actually saturated, the objective was lower with the PS-VNFR variant.

Figures 8 and 9 show the CPU Time needed for the resolution of the ILP and the execution of the heuristic, when varying respectively the number of nodes in the graph and the number of sections K. The y-axis scale is logarithmic so it appears from the figures that there is almost always at least an order of magnitude in the performance of the heuristic and the exact resolution.

Figures 10 and 11 represent the relative gap between the DC-VNFR solution and the optimum. The relative gap is defined as $(\beta - \alpha)/\alpha$ with α the minimum cost and β the cost of the solution provided by DC-VNFR. When varying the number of nodes, the relative gap is always below 0.6%.

4.3 Discussion

The results given in section 4.2 first show that the performance gain with the proposed algorithm over the exact resolution is very good, irrespective of the number of nodes in the graph or the number of sections to protect. It is also noteworthy that the type B instances take more time to solve, in average, than the type A instances. This is because the combinatorial exploration is larger when the graph is more connected.

The results also show that the proposed algorithm yields high-quality solutions, in the sense that whatever the number of nodes in the graph, the gap to optimality stays under 0.6%. In fact, the heuristic is always optimal when the node capacities are not saturated. This is due to the fact that, in that case, the problems of finding the backups for each VNF type are independent from one another. Since in DC-VNFR the optimal solution is found for each VNF type separately, the combined solution remains optimal. We have also observed that when the node capacities are less saturated, the solutions of DC-VNFR tends to be closer to the optimal solutions, which seems quite natural.

However, when varying the number of sections, two opposite trends appear. With type A instances, the gap to optimality increases with the number of sections (after 100 sections). This is due to the fact that, for sparse graphs, there is a limited number of nodes to choose from, for the selection of the backup node (because of the latency constraints). In consequence, the node

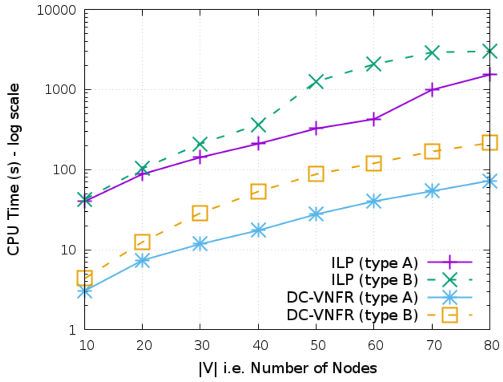


Figure 8: CPU Time vs. Number of Nodes

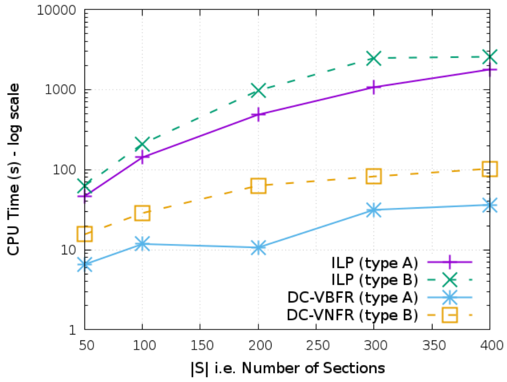


Figure 9: CPU Time vs. Number of Sections

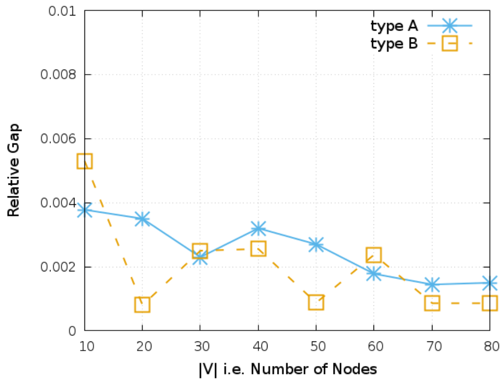


Figure 10: Gap vs. Number of Nodes

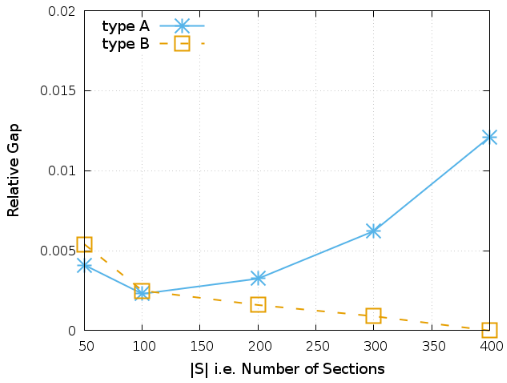


Figure 11: Gap vs. Number of Sections

capacities are more often saturated, leading to sub-optimal placement of backup VNFs. In contrast, with type B instances, the gap decreases (cf. Figure 11). This is because in that case, there is a larger number of choices for the backup node, which means they are less prone to reach their capacity, which allows the heuristic to lead to a near-optimal solution.

In conclusion, the DC-VNFR heuristic allows to take advantage of the mutualization of the backups of each particular VNF types. It is a good compromise between the optimal resolution of the ILP and a reasonable computation time.

5 CONCLUSION

In this work, we have studied how to improve the resiliency of a set of given Service Function Chains, in a practical and cost-effective manner. The aim is to deploy a backup VNF and an associated backup path for each VNF of all the chains. Since the goal is to protect against a single failure, the backups can be mutualized for several nominal VNFs, and also a nominal VNF with spare capacity can be used as backup. The formulation that we proposed allows to solve this problem at minimal cost, and an ILP-based heuristic, relying on a separation of the problem into smaller ones, is provided in order to solve large scale instances. Empirical results on instances representative of real use-cases show the benefits of this approach.

ACKNOWLEDGMENTS

This work is supported by the french Agence Nationale de la Recherche (ANR), Project MAESTRO-5G ANR-18-CE25-0012.

REFERENCES

[1] Z. Allybokus, N. Perrot, J. Leguay, L. Maggi, and E. Gourdin. 2018. Virtual Function Placement for Service Chaining with Partial Orders and Anti-Affinity Rules. *Networks* 71 (2018), 97–106.

[2] C. J. Bernardos, A. Rahman, and A. Mourad. 2018. *Service Function Chaining Use Cases in Fog RAN.* Internet-Draft draft-bernardos-sfc-fog-ran-04. Internet Engineering Task Force. https://datatracker.ietf.org/doc/html/draft-bernardos-sfc-fog-ran-04 Work in Progress.

[3] A. Engelmann and A. Jukan. 2017. A Reliability Study of Parallelized VNF Chaining. *CoRR* abs/1711.08417 (2017). arXiv:1711.08417 http://arxiv.org/abs/1711.08417

[4] ETSI. 2016. Network Functions Virtualisation (NFV) ; Reliability ; Report on Models and Features for End-to-End Reliability. *ETSI GS NFV-REL 003 V1.1.2* (2016).

[5] W. E. Hart, C. D. Laird, J.P. Watson, D. L. Woodruff, G. A. Hackebeil, B. L. Nicholson, and J. D. Siirola. 2017. *Pyomo–optimization modeling in python* (second ed.). Vol. 67. Springer Science & Business Media.

[6] A. Hmaity, M. Savi, F. Musumeci, M. Tornatore, and A. Pattavina. 2016. Virtual Network Function placement for resilient Service Chain provisioning. *2016 8th International Workshop on Resilient Networks Design and Modeling (RNDM)* (2016), 245–252.

[7] IBM ILOG. 2015. *IBM ILOG CPLEX V12.6: User's manual for CPLEX.*

[8] J. Kong, I. Kim, X. Wang, Q. Zhang, H. C. Cankaya, W. Xie, T. Ikeuchi, and J. P. Jue. 2017. Guaranteed-Availability Network Function Virtualization with Network Protection and VNF Replication. *GLOBECOM 2017 - 2017 IEEE Global Communications Conference* (2017), 1–6.

[9] S. Matsushima and R. Wakikawa. 2016. *Stateless user-plane architecture for virtualized EPC (vEPC).* Internet-Draft draft-matsushima-stateless-uplane-vepc-06. Internet Engineering Task Force. https://datatracker.ietf.org/doc/html/draft-matsushima-stateless-uplane-vepc-06 Work in Progress.

[10] J. Napper, M. Stiemerling, D. Lopez, and J. Uttaro. 2018. *Service Function Chaining Use Cases in Mobile Networks.* Internet-Draft draft-ietf-sfc-use-case-mobility-08. Internet Engineering Task Force. https://datatracker.ietf.org/doc/html/draft-ietf-sfc-use-case-mobility-08 Work in Progress.

[11] L. Qu, C. M. Assi, K. B. Shaban, and M. J. Khabbaz. 2017. A Reliability-Aware Network Service Chain Provisioning With Delay Guarantees in NFV-Enabled Enterprise Datacenter Networks. *IEEE Transactions on Network and Service Management* 14 (2017), 554–568.

[12] D. A. Schult. 2008. Exploring network structure, dynamics, and function using NetworkX. In *In Proceedings of the 7th Python in Science Conference (SciPy.* 11–15.

[13] W. Wang and J. Doucette. 2018. Availability optimization in shared-backup path protected networks. *IEEE/OSA Journal of Optical Communications and Networking* 10, 5 (May 2018), 451–460. https://doi.org/10.1364/JOCN.10.000451

Valid constraints for time-indexed formulations of job scheduling problems with distinct time windows and sequence-dependent setup times

Full Paper

Bruno Ferreira Rosa
Federal Center of Technological Education of Minas Gerais
Divinópolis, MG, Brazil
brunorosa@cefetmg.br

Marcone Jamilson Freitas Souza
Department of Computing, Federal University of Ouro Preto
Ouro Preto, MG, Brazil
marcone@ufop.edu.br

Sérgio Ricardo de Souza
Federal Center of Technological Education of Minas Gerais
Belo Horizonte, MG, Brazil
sergio@dppg.cefetmg.br

Zacharie Ales
ENSTA ParisTech/UMA
Paris, France
zacharie.ales@ensta-paristech.fr

Philippe Yves Paul Michelon
University of Avignon
Avignon, France
philippe.michelon@univ-avignon.fr

ABSTRACT

This paper addresses the single machine scheduling problem with distinct time windows, sequence-dependent setup times (SMSPETP) which consists in minimizing the total weighted earliness and tardiness of a set of jobs. We propose a time-indexed mathematical formulation for representing the problem, new valid constraints families for this formulation, as well as separation algorithms. Computational experiments show that the use of these algorithms in a cutting-plane enable to significantly improve the linear relaxation.

1 INTRODUCTION

This paper addresses the single machine scheduling problem with distinct time windows and sequence-dependent setup times. Such problem consists of determining the time at which jobs must be performed in order to minimize the weighted sum of earliness and tardiness penalties, and is hereafter denoted by SMSPETP.

The SMSPETP is a difficult problem which has numerous applications, such as Just-in-Time manufacturing, chemical processing, video on demand services, among others. As a consequence, many resolution algorithms have been introduced to solve this problem [3, 5]. Nevertheless, the job scheduling problem with the characteristics considered in this work has not received the deserved attention. The SMSPETP has mainly been treated by heuristic procedures that divide the problem into two subproblems: (*i*) job sequencing, and (*ii*) determining the optimal time for completion of each job in a given sequence. This work tackles the SMSPETP from a perspective not yet considered in the literature i.e., with a cutting plane algorithm.

The SMSPETP has the following characteristics:

- A single machine must process a set I of n jobs;
- The machine can perform only one job at a time and, once the process is initiated, it cannot be interrupted;
- All jobs are available for processing starting from date 0;
- Between two consecutive jobs x and $y \in I$, a setup time of S_{xy} is required. It is assumed that the time for setting up the machine in order to process the first job in the sequence is equal to 0;
- Idle time between the execution of two consecutive jobs is allowed.
- For each job $x \in I$, there is a processing time P_x and a time window $[E_x, T_x]$ in which the job x should preferably be completed. E_x indicates the earliest due date, and T_x is the tardiest due date;
- If job x is completed before E_x, then there is a cost of α_x per unit of earliness time. In the case that the job is completed after T_x, there is a cost of β_x per unit of tardiness time. Jobs completed within their time windows do not incur costs;

The objective of the problem is to determine the starting dates of the jobs, so that the weighted sum of their earliness and tardiness is minimized, i.e.,

$$\min \sum_{x \in I} (\alpha_x e_x + \beta_x t_x), \tag{1}$$

where C_x represents the completion time of job $x \in I$ and $e_x = \max(0,\ E_x - C_x)$ and $t_x = \max(0, C_x - T_x)$ represent the earliness and tardiness times of x, respectively.

In this paper, a time-indexed formulation for representing the SMSPETP is presented. In addition, five families of valid constraints for time-indexed SMSPETP formulations are proposed in order to obtain better lower bounds.

The rest of this article is organized as follows. The time-indexed formulation for the SMSPETP is presented in Section 2, while the five families of valid constraints for time-indexed SMSPETP formulations are showed in Section 3. Section 4 proposes separations algorithms for these families of constraints. Section 5 presents and discusses the computational results. Finally, Section 6 concludes this work.

2 THE PROPOSED TIME-INDEXED FORMULATION

In [6, 7] were introduced time-indexed formulations of the single machine scheduling problem with distinct deadlines and no

© 2019 Copyright held by the owner/author(s). Published in Proceedings of the 9th International Network Optimization Conference (INOC), June 12-14, 2019, ISBN 978-3-89318-078-3 on OpenProceedings.org.

DOI: 10.5441/002/inoc.2019.09

setup times. We adapt these formulations and use the valid constraints of [1] in order to represent the SMSPETP.

Let $H_x = \{s_x^{LB}, s_x^{LB}+1, \ldots, s_x^{UB}\}$ be the set of possible starting dates of job $x \in I$. Let l_{xh} be decision variables such that $\forall\, x \in I$ and $\forall\, h \in H_x$,

$$l_{xh} = \begin{cases} 1, & \text{if job } x \text{ begins at date } h; \\ 0, & \text{otherwise.} \end{cases}$$

In the rest of this work the following notations are used: $\lfloor \lambda \rfloor_x = \max(s_x^{LB}, \lambda)$ and $\lceil \lambda \rceil_x = \min(\lambda, s_x^{UB})$, for all job $x \in I$ and for all number $\lambda \in R$.

As introduced in [7], the cost incurred by the earliness or tardiness of a job $x \in I$ started at date h can be determined by the function

$$g_x(h) = \alpha_x \cdot \max(E_x - h - P_x, 0) + \beta_x \cdot \max(h + P_x - T_x, 0),\ \forall\, h \in H_x \quad (2)$$

Therefore, a time-indexed formulation for the SMSPETP, denoted by TIF, is given by

$$\text{(TIF)} \qquad \min \sum_{x \in I} \sum_{h \in H_x} g_x(h) \cdot l_{xh}$$

$$\text{s.t.} \qquad \sum_{h \in H_x} l_{xh} = 1, \quad \forall x \in I \quad (3)$$

$$\sum_{k=\lfloor h-P_x-S_{xy}+1 \rfloor_x}^{\lceil h \rceil_x} l_{xk} + \sum_{k=\lfloor h-P_y-S_{yx}+1 \rfloor_y}^{\lceil h \rceil_y} l_{yk} \le 1, \quad \forall x, y \in I, x \neq y,$$
$$\forall h \in H_x \cup H_y \quad (4)$$

$$l_{xh} \in \{0,\ 1\}, \quad \forall x \in I, \forall h \in H_x \quad (5)$$

The objective function seeks to minimize the weighted sum of the earliness and tardiness. Constraints (3) assure that each job will be executed only once. Constraints (4) ensure that there is sufficient time to execute a job and prepare the machine before starting the next job. Note that Constraints (4) assume the validy of the triangle inequality given by

$$S_{xy} \le S_{xz} + P_z + S_{zy}, \quad \forall\ x,\ y,\ z \in I,\ x \neq y,\ x \neq z \text{ and } y \neq z. \quad (6)$$

3 NEW VALID CONSTRAINTS

This section introduces new families of valid constraints for time-indexed formulations to the SMSPETP.

Before presenting the new valid constraints, it is observed that the constraints given by Proposition 3.1 of [1] are also valid for the time-indexed formulations of SMSPETP. These constraints are used to prove the validity of the first family of valid constraints.

Proposition 3.1 ([1]). *Given a subset $I' \subseteq I$ such that $|I'| \ge 2$, for all $h \in \bigcup_{x \in I'} H_x$, we have*

$$\sum_{x \in I'} \sum_{k=\left\lfloor h-P_x-\min\limits_{y \in I' \setminus \{x\}} S_{xy}+1 \right\rfloor_x}^{\lceil h \rceil_x} l_{xk} \le 1. \quad (7)$$

Note that Constraints (4) are obtained from Constraints (7) by considering only the subsets $I' \subset I$ such that $|I'| = 2$.

The first family of valid constraints is inspired by [6]. In this work, the authors propose the set of Constraints (8) for time-indexed formulations of scheduling problems without setup time between jobs:

$$\sum_{k=\lfloor h-P_x+1 \rfloor_x}^{\lceil h+\Delta-1 \rceil_x} l_{xk} + \sum_{y \in I \setminus \{x\}: P_y \ge \Delta} \ \sum_{k=\lfloor h-P_y+\Delta \rfloor_y}^{\lceil h \rceil_y} l_{yk} \le 1, \quad \forall x \in I,$$
$$\forall h \in \bigcup_{y \in I \setminus \{x\}} H_y,\ \forall \Delta \in \{2, 3, \ldots, \max_{y \in I \setminus \{x\}} P_y\} \quad (8)$$

Proposition 3.2 generalizes Constraints (8) to scheduling problems with setup times between jobs. The family of constraints that satisfies Proposition 3.2 is named here "Family 1".

Proposition 3.2 (Family 1). *Let $I' \subseteq I$ be a subset of jobs such that $|I'| \ge 2$. Given a job $x \in I'$, for all $h \in \bigcup_{y \in I' \setminus \{x\}} H_y$ and all $\Delta \in \{2 - P_x - \min_{y \in I' \setminus \{x\}} S_{xy}, \ldots, \max_{y \in I' \setminus \{x\}} (P_y + S_{yx})\}$, we have*

$$\underbrace{\sum_{k=\lfloor h-P_x-S_x^{min}+1 \rfloor_x}^{\lceil h+\Delta-1 \rceil_x} l_{xk}}_{\varepsilon_x} + \sum_{y \in I^*} \underbrace{\sum_{k=\lfloor h-P_y-S_y^{min}+\Delta \rfloor_y}^{\lceil h \rceil_y} l_{yk}}_{\epsilon_y} \le 1, \quad (9)$$

where:

- $I^* = \{y \in I' \setminus \{x\} \mid P_y + S_{yx} \ge \Delta\}$,
- $S_x^{min} = \min_{y \in I^*} S_{xy}$ *and*
- $S_y^{min} = \min\left(S_{yx}, \Delta - 1 + \min_{z \in I^* \setminus \{y\}} S_{yz}\right)$ *for all $y \in I^*$ (if $I^* = \{y\}$, then $S_y^{min} = S_{yx}$).*

Proof. As job x must be processed once, we have $\varepsilon_x \le 1$. Moreover, we have from Proposition 3.1 that $\sum_{y \in I^*} \epsilon_y \le 1$. Suppose there is a feasible scheduling π of I in which $\varepsilon_x = 1$ and $\sum_{y \in I^*} \epsilon_y = 1$ for a given h' and a given Δ'. Thus, there is a job $y' \in I^*$ such that $\epsilon_{y'} = 1$. Consequently, $h' - P_x - S_{xy'} + 1 \le s_x^\pi \le h' + \Delta' - 1$ and $h' - P_{y'} - S_{y'x} + \Delta' \le s_{y'}^\pi \le h'$, where s_x^π is the starting date of job x in scheduling π, i.e., there is overlap between jobs x and y' in scheduling π. □

Note that Constraints (7) are contained in Family 1. In fact, they are obtained by only setting $\Delta = 1$ in Family 1.

The following lemma provides a new set of valid constraints.

Lemma 3.3. *For any subset $I' \subseteq I$, we have*

$$\sum_{x \in I'} \underbrace{\left(\sum_{k=\lfloor h \rfloor_x}^{\lceil h + \min\limits_{y \in I' \setminus \{x\}} (P_y + S_{yx}) - 1 \rceil_x} l_{xk} \right)}_{\epsilon_x} \le 1, \quad \forall\, h \in \bigcup_{x \in I'} H_x. \quad (10)$$

Proof. As every job must be processed once, we have $\epsilon_x \le 1$, $\forall x \in I'$ and $\forall h \in \bigcup_{x \in I'} H_x$. Suppose there is a feasible scheduling π of I such that $\sum_{x \in I'} \epsilon_x > 1$ for a given h'. Therefore, there are two jobs $x_1, x_2 \in I'$ such that $\epsilon_{x_1} = \epsilon_{x_2} = 1$. Consequently, $h' \le s_{x_1}^\pi \le h' + P_{x_2} + S_{x_2,x_1} - 1$ and $h' \le s_{x_2}^\pi \le h' + P_{x_1} + S_{x_1,x_2} - 1$, where s_x^π is the starting date of job x in scheduling π, that is, the machine performs jobs x_1 and x_2 in scheduling π simultaneously. This contradicts the fact that π is a feasible scheduling of I. □

Proposition 3.4 provides another family of valid constraints, named "Family 2". This family contains Constraints (10).

Proposition 3.4 (Family 2). *Let $I' \subseteq I$ be a subset of jobs such that $|I'| \ge 2$. Given a job $x \in I'$, for all $h \in \bigcup_{y \in I' \setminus \{x\}} H_y$ and all $\Delta \in \{2 - \min_{y \in I^*} (P_y + S_{yx}), \ldots, P_x + \max_{y \in I' \setminus \{x\}} S_{xy}\}$, we have:*

$$\underbrace{\sum_{k=\lfloor h-\Delta+1 \rfloor_x}^{\lceil h+PS_x^{min}-1 \rceil_x} l_{xk}}_{\varepsilon_x} + \sum_{y \in I^*} \underbrace{\sum_{k=\lfloor h \rfloor_y}^{\lceil h+PS_y^{min}-\Delta \rceil_y} l_{yk}}_{\epsilon_y} \le 1, \quad (11)$$

where:

- $I^* = \{y \in I' \setminus \{x\} | P_x + S_{xy} \ge \Delta\}$,
- $PS_x^{min} = \min_{y \in I^*} (P_y + S_{yx})$ *and*
- $PS_y^{min} = P_x + S_{xy}$, *if $I^* = \{y\}$, or $PS_y^{min} = \min\left(P_x + S_{xy}, \Delta - 1 + \min_{z \in I^* \setminus \{y\}} (P_z + S_{zy})\right)$ for all $y \in I^*$, otherwise.*

Proof. Since job x must be performed only once, we have $\varepsilon_x \leq 1$. Besides it, from Constraints (10) we have $\sum_{y \in I^*} \epsilon_y \leq 1$. Suppose there is a schedule π from I such that $\varepsilon_x = 1$ and $\sum_{y \in I^*} \epsilon_y = 1$ to a given date h' and a given Δ'. Therefore, there is $y' \in I^*$ such that $\epsilon_{y'} = 1$. Consequently, $h' - \Delta' + 1 \leq s_x^\pi \leq h' + P_{y'} + S_{y'x} - 1$ and $h' \leq s_{y'}^\pi \leq h' + P_x + S_{xy'} - \Delta'$, where s_x^π is the starting date of job x in scheduling π, that is, the machine performs jobs x and y' in schedule π simultaneously. This contradicts the fact that π is a feasible schedule of I. □

Constraints (10) are obtained from Family 2 when considering $\Delta = 1$. Propositions 3.5, 3.6 and 3.7 provide three more families of valid constraints, which will be named "Family 3", "Family 4" and "Family 5" respectively.

Proposition 3.5 (Family 3). *For any subset $I' \subseteq I$ such that $|I'| \geq 2$, we have:*

$$\sum_{x \in I'} \underbrace{\sum_{k = s_x^{LB}}^{\left\lceil \min\limits_{y \in I' \setminus \{x\}} (s_y^{LB} + P_y + S_{yx}) - 1 \right\rceil_x} l_{xk}}_{\epsilon_x} \leq 1. \tag{12}$$

Proof. According to what has already been discussed, $\epsilon_x \leq 1, \forall x \in I'$. Suppose there is a schedule π of I such that $\sum_{x \in I'} \epsilon_x > 1$. Therefore, there are two jobs x_1,$x_2 \in I'$ such that $\epsilon_{x_1} = \epsilon_{x_2} = 1$. Consequently, $s_{x_1}^{LB} \leq s_{x_1}^\pi \leq s_{x_2}^{LB} + P_{x_2} + S_{x_2,x_1} - 1$ and $s_{x_2}^{LB} \leq s_{x_2}^\pi \leq s_{x_1}^{LB} + P_{x_1} + S_{x_1,x_2} - 1$, where s_x^π is the starting date of job x in scheduling π, i.e., the machine performs jobs x_1 and x_2 in scheduling π simultaneously. This contradicts the fact that π is a feasible scheduling of I. □

Proposition 3.6 (Family 4). *Given a subset of jobs $I' \subseteq I$ such that $|I'| \geq 2$, if $TPT_{\min}^{I'}$ denotes the lowest total time required to process all jobs in I' and $s_{I'}^{LB} = \min_{x \in I'} s_x^{LB}$, then*

$$\underbrace{\sum_{x \in I'} \sum_{h = \left\lfloor s_{I' \setminus \{x\}}^{LB} + TPT_{\min}^{I' \setminus \{x\}} + \min\limits_{y \in I \setminus \{x\}} S_{yx} \right\rfloor_x}^{s_x^{UB}} l_{xh}}_{\epsilon} \quad \geq \quad 1. \tag{13}$$

Proof. Suppose there is a feasible scheduling π of I such that ϵ is equal to 0. Let s_x^π be the starting date of job x in scheduling π and $x' \in I'$ be the last job processed in π. Thus, $s_{x'}^\pi < s_{I' \setminus \{x\}}^{LB} + TPT_{\min}^{I' \setminus \{x'\}} + \min_{y \in I \setminus \{x'\}} S_{yx'}$ and there is a scheduling of $I' \setminus \{x'\}$ whose total processing time is lower than $TPT_{min}^{I' \setminus \{x'\}}$. This contradicts the fact that $TPT_{\min}^{I' \setminus \{x'\}}$ is the lowest total time required to process all jobs in $I' \setminus \{x'\}$. □

Proposition 3.7 (Family 5). *Given a pair of distinct jobs x and y in I, for all $h \in \left\{ \max\left(s_x^{LB}, s_y^{LB}\right), \ldots, \min\left(s_x^{UB}, s_y^{LB} + P_y + S_{yx} - 1\right) \right\} \cap H_x$, we have*

$$\underbrace{\sum_{k = h + P_x + S_{xy}}^{s_y^{UB}} l_{yk}}_{\epsilon} \geq l_{xh}. \tag{14}$$

Proof. Suppose there is a feasible scheduling π of I such that $\epsilon < l_{xh}$ for a given h (i.e., such that $l_{xh} = 1$ and $\epsilon = 0$). So, $s_x^\pi = h$ and $s_y^{LB} \leq s_y^\pi < h + P_x + S_{xy}$, where s_x^π denotes the starting date of job x in scheduling π. Since $s_y^{LB} \leq h \leq s_y^{LB} + P_y + S_{yx} - 1$, it follows that $s_x^\pi + P_x + S_{xy} > s_y^\pi$ and $s_y^\pi + P_y + S_{yx} > s_x^\pi$, contradicting the fact that π is a feasible scheduling of I. □

4 SEPARATION ALGORITHMS FOR THE FAMILIES OF PROPOSED CONSTRAINTS

Except Family 5, all families of constraints proposed in Section 3 have an exponential number of constraints (2^n or greater). This fact makes it impossible to fully include these families in the formulations. However, they can be used in cutting-plane algorithms [8]. In short, cutting-plane algorithms are procedures that start from the solution of the linear relaxation of a formulation in which a limited number of constraints are considered and iteratively adds new valid constraints to the problem and solve it, until one stopping criterion is satisfied.

Let *PPM* be the mathematical programming problem based on time-indexed variables that is updated iteratively in a given cutting-plane algorithm. Consider that $l^\star$ represents an optimal solution of the linear relaxation of the current *PPM*. Note that $l^\star$ consists of an array of values assigned to the variables l_{xh}, $\forall x \in I$ and $\forall h \in H_x$. Due to the large number of constraints in Families 1–4, the simple fact of checking which constraints are violated by $l^\star$ is still an impractical process. The problem of finding, in a set of constraints, those that are violated by $l^\star$ is called a "separation problem".

The separation problem associated with Family 5 is solved exactly by checking all constraints, one by one. On the other hand, the separation problems of Families 1–4 are solved heuristically. Moreover, the separation algorithms seek only the constraint which are the most violated by $l^\star$. Constraints whose violation by $l^\star$ is small are discarded. A constraint of type $A \times l \leq b$ is violated by $l^\star$ for at least $\delta > 0$ units if $A \times l^\star \geq b + \delta$.

The algorithms proposed to solve the separation problems associated with the families of constraints presented in Section 3 are described in the following subsections. Let δ represents the minimum violation accepted. Given a solution $l^\star$ for the current *PPM*, let $h_x^{\min} = \min\{h \in H_x : l_{xh}^\star > 0\}$ and $h_x^{\max} = \max\{h \in H_x : l_{xh}^\star > 0\}$, $\forall x \in I$. Furthermore, for each job $x \in I$, let $I_x = \left\{y \in I \setminus \{x\} : h_y^{\max} + P_y + S_{yx} > h_x^{\min} \text{ or } h_x^{\max} + P_x + S_{xy} > h_y^{\min}\right\}$.

4.1 Separation Heuristic for Family 1

The proposed separation heuristic algorithm for Family 1 is described in Algorithm 1. In this algorithm, Ω_1 represents the set of constraints violated by $l^\star$. $lhs_{x,h,I',\Delta}(l^\star)$ represents the numerical value of the expression $\varepsilon_x + \sum_{y \in I^*} \epsilon_y$ related to Proposition 3.2 applied to $l^\star$, for the respective x, h, I' and Δ.

The jobs are sorted in ascending order by the values of $h_x^{\min}$. The maximum number of constraints returned by the separation heuristic of Family 1 is given by $\sum_{x \in I}(h_x^{\max} - h_x^{\min} + 1)$.

4.2 Separation Heuristic for Family 2

Let Ω_2 be a subset of Family 2 composed of constraints which are violated by $l^\star$. If $lhs_{x,h,I',\Delta}(l^\star)$ represents the numerical value of the expression $\varepsilon_x + \sum_{y \in I^*} \epsilon_y$ of Proposition 3.4 applied to $l^\star$, then the proposed heuristic algorithm for the separation problem of Family 2 is analogous to Algorithm 1. The only differences are the range of Δ and the function $lhs_{x,h,I',\Delta}(l^\star)$, which, in this case, are based on Proposition 3.4.

```
Input: $I$; $h_x^{\max}$, $h_x^{\min}$, $I_x$, $lhs_{x,h,I',\Delta}(l^\star)$ $\forall x \in I$; $l^\star$; $\delta \in R$.
$\Omega_1 \leftarrow \emptyset$;
for $x \in I$ do
    for $h = h_x^{\max}, h_x^{\max} - 1, \cdots, h_x^{\min}$ do
        $I' \leftarrow \{x\}$;
        $lhs_0 \leftarrow -\infty$;
        Update ← FALSE;
        while $I' \neq I_x \cup \{x\}$ do
            $y^* \leftarrow -1$;
            $lhs^* \leftarrow -\infty$;
            for $y \in I_x \setminus I'$ do
                $I' \leftarrow I' \cup \{y\}$;
                for $\Delta = P_y + S_{yx}, P_y + S_{yx} - 1, \cdots, 2 - P_x - \min_{z \in I \setminus \{x\}} S_{xz}$ do
                    if $lhs_{x,h,I',\Delta}(l^\star) \geq 1 + \delta$ then
                        $\Omega_1 = \Omega_1 \cup \{lhs_{x,h,I',\Delta}(l) \leq 1\}$;
                        Update ← TRUE;
                        Exit the current loop;
                    if $lhs_{x,h,I',\Delta}(l^\star) > lhs^*$ then
                        $lhs^* \leftarrow lhs_{x,h,I',\Delta}(l^\star)$;
                        $y^* \leftarrow y$;
                if Update = TRUE then
                    Exit the current loop;
                else
                    $I' \leftarrow I' \setminus \{y\}$;
            if Update = TRUE then
                Exit the current loop;
            if $lhs^* > lhs_0$ then
                $lhs_0 \leftarrow lhs^*$;
            else
                Exit the current loop;
            $I' \leftarrow I' \cup \{y^*\}$;
Return $\Omega_1$;
```

Algorithm 1: Separation Heuristic for Family 1.

4.3 Separation Heuristic for Family 3

The proposed heuristic algorithm for the separation problem of Family 3 is detailed in Algorithm 2. Ω_3 represents the set of constraints violated by $l^\star$ found by this algorithm. Let $lhs_{I'}(l^\star)$ represents the numerical value of the expression $\sum_{x \in I'} \epsilon_x$ of Proposition 3.5 applied to $l^\star$, for the respective subset $I' \subseteq I$.

As in the separation heuristics of Families 1 and 2, the order of investigation of the jobs $x \in I$ is always given in the increasing order of $h_x^{\min}$. The maximum number of constraints returned by the separation heuristic of Family 3 is equal to n.

4.4 Separation Heuristic for Family 4

Before presenting the proposed separation for Family 4, it is observed Constraint (13) of Proposition 3.6 is equivalent to Constraint (15) for all subset $I' \subseteq I$.

$$\underbrace{\sum_{x \in I'} \sum_{h = s_x^{LB}}^{\left\lceil s_{I' \setminus \{x\}}^{LB} + TPT_{\min}^{I' \setminus \{x\}} + \min_{y \in I \setminus \{x\}} S_{yx} - 1 \right\rceil_x} l_{xh}}_{\epsilon'_{I'}} - |I'| \leq -1. \qquad (15)$$

Let Ω_4 be a subset of Family 4 composed of constraints that are violated by $l^\star$. Let $lhs_{I'}(l^\star)$ be the numerical value of expression $\epsilon'_{I'} - |I'|$ of Constraint (15) applied to $l^\star$, for the respective subset $I' \subseteq I$.

```
Input: $I$; $I_x$, $lhs_{I'}(l^\star)$ $\forall x \in I$; $l^\star$; $\delta \in R$.
$\Omega_3 \leftarrow \emptyset$;
for $x \in I$ do
    $I' \leftarrow \{x\}$;
    $lhs_0 \leftarrow -\infty$;
    Update ← FALSE;
    while $I' \neq I_x \cup \{x\}$ do
        $y^* \leftarrow -1$;
        $lhs^* \leftarrow -\infty$;
        for $y \in I_x \setminus I'$ do
            $I' \leftarrow I' \cup \{y\}$;
            if $lhs_{I'}(l^\star) \geq 1 + \delta$ then
                $\Omega_3 = \Omega_3 \cup \{lhs_{I'}(l) \leq 1\}$;
                Update ← TRUE;
                Exit the current loop;
            if $lhs_{I'}(l^\star) > lhs^*$ then
                $lhs^* \leftarrow lhs_{I'}(l^\star)$;
                $y^* \leftarrow y$;
            $I' \leftarrow I' \setminus \{y\}$;
        if Update = TRUE then
            Exit the current loop;
        if $lhs^* > lhs_0$ then
            $lhs_0 \leftarrow lhs^*$;
        else
            Exit the current loop;
        $I' \leftarrow I' \cup \{y^*\}$;
Return $\Omega_3$;
```

Algorithm 2: Separation Heuristic for Family 3.

The heuristic algorithm proposed for the separation problem of Family 4 is similar to that of Family 3. The only difference is in the function $lhs_{I'}(l^\star)$, which, in this case, is based on Constraint (15). In addition, instead of the exact value of $TPT_{\min}^{I'}$, a lower bound is used for that value. The lower bound used is provided by Corollary 4.1, which follows from the results proposed in [4].

Corollary 4.1. *For every subset $I' \subseteq I$, the shortest total time required to perform all jobs of I', that is $TPT_{min}^{I'}$, is such that*

$$TPT_{\min}^{I'} \geq \sum_{x \in I'} P_x + \max\left(\sum_{x \in I'} \min_{y \in I' \setminus \{x\}} S_{yx} - \max_{x \in I'} \min_{y \in I' \setminus \{x\}} S_{yx}, \right.$$
$$\left. \sum_{x \in I'} \min_{y \in I' \setminus \{x\}} S_{xy} - \max_{x \in I'} \min_{y \in I' \setminus \{x\}} S_{xy} \right).$$

4.5 Separation Algorithm for Family 5

The separation of Family 5 is solved exactly.

The proposed algorithm for the separation problem of Family 5 is detailed in Algorithm 3. Ω_5 represents the set of constraints violated by $l^\star$ found by this algorithm.

5 COMPUTATIONAL RESULTS

This Section presents the computational results obtained with the time-indexed formulation for the SMSPETP presented in Section 2, as well as with the different families of constraints proposed in Section 3. The separation algorithms described in Section 4 are used in a cutting plane framework in order to experiment how much they enable to improve the linear relaxation.

The mathematical formulations were implemented and solved through the C++ Concert Technology tool and the IBM ILOG CPLEX Optimization Studio 12.6.2 solver. The separation heuristics used for testing the proposed families of constraints were

```
Input: $I$;  $s_x^{LB}$, $s_x^{UB}$ $\forall x \in I$;  $l^\star$;  $\delta \in R$.
$\Omega_5 \leftarrow \emptyset$;
for $x \in I$ do
    for $y \in I \setminus \{x\}$ do
        for $h = \max(s_x^{LB}, s_y^{LB}), \max(s_x^{LB}, s_y^{LB}) + 1, \cdots, \min(s_x^{UB}, s_y^{LB} + P_y + S_{yx} - 1)$ do
            if $\sum_{k=h+P_x+S_{xy}}^{s_y^{UB}} l^\star_{yk} \leq l^\star_{xh} - \delta$ then
                $\Omega_5 = \Omega_5 \cup \{ \sum_{k=h+P_x+S_{xy}}^{s_y^{UB}} l_{yk} \geq l_{xh} \}$;
Return $\Omega_5$;
```

Algorithm 3: Separation Algorithm for Family 5.

also implemented in C++ language. The experiments were realized on a computer Intel® Xeon(R) CPU E5620 @ 2.40GHz × 16, with 48 GB of RAM and CentOS Linux 7 operation system. CPLEX was configured to use only one thread and the other parameters were not changed. In addition, the algorithms were not optimized for multiprocessing.

A set of instances of [5], involving up to 20 jobs and satisfying the triangle inequality, was used in order to test the proposed formulations. This set contains 16 instances of each value of n. For each job $x \in I$, the bounds s_x^{UB} and s_x^{LB} used for determining the parameter values of each mathematical formulation are the same than in [4].

The cutting-plane algorithm described in Algorithm 4 was used in order to obtain lower bounds to the SMSPETP. The strategy that was used is based on the Variable Neighborhood Descent – VND [2] procedure. It uses a subsequencing of m separation algorithms proposed in Section 4, where $1 \leq m \leq 5$,.

```
$PPM \leftarrow PPM_0$;
$l^\star \leftarrow$ solution of PPM;
$\delta \leftarrow 0.8$;
while $\delta \geq 0.1$ do
    $i \leftarrow 1$;
    while $i \leq m$ do
        Solve the separation problem related to the $i$-th family of constraints for $l^\star$ and $\delta$;
        if there are constraints that are violated by $l^\star$ then
            Add these constraints to the current $PPM$;
            $l^\star \leftarrow$ solution of the current $PPM$;
            Eliminate from the current $PPM$ the constraints satisfied by $l^\star$ with non-zero slack;
            $i \leftarrow 1$;
        else
            $i \leftarrow i + 1$;
    $\delta \leftarrow \delta \div 2$;
Return $l^\star$;
```

Algorithm 4: Lower Bound obtained with m families of constraints.

In Algorithm 4, the initial PPM is provided by the PPM_0 formulation, defined by Equations (16)–(18).

$$(PPM_0) \quad \min \sum_{x \in I} \sum_{h \in H_x} g_x(h) \cdot l_{xh} \tag{16}$$

$$\text{s.t.} \quad \sum_{h \in H_x} l_{xh} = 1 \quad \forall\, x \in I \tag{17}$$

$$l_{xh} \in [0, 1] \quad \forall x \in I \text{ and } \forall h \in H_x \tag{18}$$

Equation (16) represents the objective function of SMSPETP. Constraints (17) ensure that each job must be executed once.

Given an instance of the problem, the gap of a given lower bound LB with respect to a given integer solution value $f^\star$ is determined by Equation (19):

$$gap = \frac{f^\star - LB}{f^\star} \times 100. \tag{19}$$

The lower the value of the gap, the better the lower bound LB is. We consider the best integer solutions from [5] to compute the gaps.

The results are reported in Table 1. In this table, the first column indicates the number of jobs of each set consisting of 16 instances. Columns "TIF" present the results using the linear relaxation of the proposed time-indexed formulation. Columns "Family 1", "Family 2", …, "Family 5" report the results by applying Algorithm 4 with the corresponding separation algorithm. Columns "Family 1–5" show the results by applying the Algorithm 4 with the five proposed separation algorithms in this order: Families 1, 2, 4, 3 and 5. For each set of instances, columns "$\overline{gap}$" and "$\overline{\text{time}}$" show, respectively, the average gap of the lower bounds (in %) and the average time, in seconds, required for each strategy over the 16 corresponding instances.

According to Table 1, the smallest average gaps obtained with only one family of constraints are Family 1, followed by Families 2, 4, 3 and 5, in this order (this justifies the choice of this sequence of separation algorithms when using all the constraint families). The difference between the average gaps of the lower bounds obtained with Family 1 and the average gaps obtained with Family 2 is relevant. The same happens with the difference between the average gaps of the lower bounds constructed with Families 2 and 4. The larger average times were also observed when using Family 1, followed by the average times required with Family 2. The average times required by Families 3, 4 and 5 were less than 2 seconds. However, the average gaps of the lower bounds constructed with these families of constraints were greater than or equal to 72.00 %.

Also according to Table 1, the average times required to obtain the lower bounds with the Families 1–5 were always higher than the average time required for solving the linear relaxation of the TIF formulation. However, the average gaps of the lower bounds resulting from the application of Algorithm 4 are significantly lower than the average gaps obtained with linear relaxation. The lower gaps of the average gaps obtained with the linear relaxations of the TIF formulation are greater than 37%, while the average gaps obtained by Families 1–5 are less than 6%. The average gap of the lower bounds obtained with the families 1–5 for the instances with 6 jobs are null, that is, the Algorithm 4 has found the optimal whole solutions of these problems. Although it is not shown in Table 1, the Algorithm 4 has found the optimal integer solutions of a total of 87 instances, among them an instance with 20 jobs.

6 CONCLUSIONS

In this work a time-indexed formulation, named TIF, for solving the Single Machine Scheduling Problem with distinct time windows and sequence-dependent setup times (SMSPETP) is proposed. Five new families of valid constraints for time-indexed formulations as well as separation algorithms for these families are also introduced.

Table 1: Results obtained when applying the Algorithm 1 in the instances.

n	TIF		Family 1		Family 2		Family 3		Family 4		Family 5		Families 1-5	
	$\overline{gap}$ (%)	time (s)	$\overline{gap}$ (%)	time (s)	$\overline{gap}$ (%)	time (s)	$\overline{gap}$ (%)	time (s)	$\overline{gap}$ (%)	time (s)	$\overline{gap}$ (%)	time (s)	$\overline{gap}$ (%)	time (s)
06	37.85	0.25	0.06	0.68	1.59	1.49	84.90	0.06	72.00	0.08	87.83	0.08	0.00	0.65
07	49.03	0.47	0.17	2.95	4.26	5.31	83.77	0.10	74.81	0.10	87.87	0.09	0.02	2.79
08	55.46	0.67	0.44	7.08	8.26	7.86	83.07	0.11	73.74	0.09	86.03	0.12	0.24	7.15
09	56.69	1.01	1.58	14.70	13.01	11.51	89.06	0.13	80.10	0.13	91.92	0.14	1.29	14.88
10	58.47	1.82	0.87	25.63	9.34	27.92	90.86	0.20	82.12	0.18	92.29	0.20	0.46	26.66
11	64.28	2.17	2.60	47.07	14.72	39.07	92.97	0.25	85.59	0.25	94.44	0.27	1.90	50.40
12	68.74	2.83	3.26	77.21	18.58	55.36	89.98	0.39	81.83	0.36	92.37	0.38	2.21	89.28
13	63.79	3.66	3.25	83.64	22.02	54.27	88.43	0.50	83.52	0.44	90.95	0.48	2.80	94.84
14	64.79	6.12	2.16	153.44	17.84	94.49	91.16	0.58	85.64	0.56	92.87	0.57	1.71	171.17
15	70.10	7.35	4.20	195.47	24.44	127.44	91.35	0.81	87.23	0.82	93.87	0.84	3.18	231.52
16	71.55	8.41	5.42	362.04	25.90	187.69	90.84	0.71	87.15	0.74	92.87	0.82	4.94	372.63
17	73.08	11.43	5.24	415.57	26.62	253.60	90.91	0.97	86.98	1.06	92.56	1.07	4.88	470.41
18	69.07	15.26	4.22	516.20	24.91	279.38	92.66	1.26	88.95	1.33	93.84	1.44	3.92	578.00
19	71.70	16.32	4.23	726.58	25.83	397.62	92.51	1.61	89.85	1.63	94.06	1.70	3.79	829.96
20	74.54	23.81	6.34	887.28	26.65	558.24	93.51	1.79	90.08	1.86	94.87	1.99	5.89	1031.59

CPLEX solver was used to solve the linear relaxation of the proposed mathematical formulation applied to instances with up to 20 jobs.

The main contribution of this work is the proposition of five families of valid constraints for SMSPETP formulations based on time-indexed variables. The proposed separation heuristics for these families were also used to obtain lower bounds for instances with up to 20 jobs. The lower bounds obtained with these heuristics are significantly better than those obtained with the linear relaxation of the mathematical formulation presented in this work. Although the times required to generate such lower bounds are greater than those required by CPLEX to solve linear relaxation, the lower bounds obtained are close, or even equal, to the values of the optimal integer solutions.

It is important to note that the valid constraints proposed for the time-indexed SMSPETP formulations can also be used in many other types of scheduling problems involving sequence-dependent setup times.

ACKNOWLEDGEMENTS

The authors thank *Coordenação de Aperfeiçoamento de Pessoal de Nível Superior* - Brazil (CAPES) - Finance Code 001, *Fundação de Amparo à Pesquisa do Estado de Minas Gerais* (FAPEMIG), *Conselho Nacional de Desenvolvimento Científico e Tecnológico* (CNPq), *Centro Federal de Educação Tecnológica de Minas Gerais* (CEFET-MG) and *Universidade Federal de Ouro Preto* (UFOP) for supporting this research.

REFERENCES

[1] Pasquale Avella, Maurizio Boccia, Carlo Mannino, and Igor Vasilyev. 2016. Valid Inequalities for Time-Indexed Formulations of the Runway Scheduling Problem. In *Supplementary Proceedings of the 9th International Conference on Discrete Optimization and Operations Research and Scientific School (DOOR 2016), Vladivostok, Russia, September 19 - 23, 2016.* 787–790. http://ceur-ws.org/Vol-1623/paperapp14.pdf

[2] P. Hansen and N. Mladenović. 2001. Variable neighborhood search: Principles and applications. *European Journal of Operational Research* 130, 3 (2001), 449–467. https://doi.org/10.1016/S0377-2217(00)00100-4

[3] A. Janiak, W. A. Janiak, T. Krysiak, and T. Kwiatkowski. 2015. A survey on scheduling problems with due windows. *European Journal of Operational Research* 242, 2 (2015), 347 – 357. https://doi.org/10.1016/j.ejor.2014.09.043

[4] B. F. Rosa, , M. J. F. Souza, S. R. de Souza, P. Y. P. Michelon, and Z. Ales. 2016. Mathematical formulations for the job scheduling problem with due windows and setup times (in portuguese). In *Proceedings of the XLVIII Brazilian Symposium of Operational Research – XLVIII SBPO.* Vitória, Brazil, 4140–4151.

[5] B. F. Rosa, M. J. F. Souza, S. R. de Souza, M. F. de França Filho, Z. Ales, and P. Y. P. Michelon. 2017. Algorithms for job scheduling problems with distinct time windows and general earliness/tardiness penalties. *Computers & Operations Research* 81 (2017), 203 – 215. https://doi.org/10.1016/j.cor.2016.12.024

[6] J. P. Sousa and L. A. Wolsey. 1992. A time indexed formulation of non-preemptive single machine scheduling problems. *Mathematical Programming* 54, 1 (1992), 353–367. https://doi.org/10.1007/BF01586059

[7] Shunji Tanaka. 2012. An Exact Algorithm for the Single-Machine Earliness-Tardiness Scheduling Problem. In *Just-in-Time Systems*, Roger Z. Ríos-Mercado and Yasmín A. Ríos-Solís (Eds.). Springer Optimization and Its Applications, Vol. 60. Springer New York, 21–40. https://doi.org/10.1007/978-1-4614-1123-9_2

[8] L. A. Wolsey. 1998. *Integer programming.* Wiley-Interscience, New York, NY, USA.

Smart Grid Topology Designs[*]

Paula Carroll[†]
College of Business
Dublin, Ireland
paula.carroll@ucd.ie

Cristina Requejo
Universidade de Aveiro
Aveiro, Portugal
crequejo@ua.pt

ABSTRACT

This paper addresses supports for evolving design demands of electricity low voltage networks in urban areas. Innovations in how electricity is generated and supplied are required to support transformation of energy systems in response to climate change. We describe a MIP model to support grid upgrade decisions in the context of an energy community in an existing urban setting. We evaluate the MIP model on an adaption of an IEEE radial network benchmark instance augmented with geographic data. We present interesting computational results which suggest additional arcs to be added. Our results highlight potential research opportunities for the network optimisation community to facilitate the desired energy systems transformation challenge.

1 INTRODUCTION

The methods of electrical energy production and distribution are changing in response to climate change concerns and as technological innovations create new opportunities. Consumers can now generate electricity through rooftop photovoltaic (PV) panels, and small rooftop wind turbines [2]. End-users equipped with renewable energy generation are turning pro-active in the distribution system and becoming a so called “prosumer”. In future electricity distribution models, any member of the network could potentially generate electricity. We consider the context of an energy community, a geographically close grouping in an urban setting, who wish to collaborate together to share electricity in their local area.

Many challenges and opportunities exist to achieving a transformation of the energy system. In this paper we focus on the problem of deciding how to upgrade an existing local low voltage network to facilitate the operation of the energy community. We contribute a MIP formulation to determine which additional edges could be added to upgrade a distribution system tree topology to form a meshed topology.

We evaluate our model on a 37 node IEEE radial test feeder system [7] under a number of scenarios. We augment the test system with geographic information to create realistic renewable energy test instances. Our results show that the problem becomes more challenging as more prosumers participate in the energy community.

2 ENERGY SYSTEM TRANSFORMATION

The EU Commission’s “Clean Energy for All Europeans” package aims to drive a transformation of the energy system to ensure clean, secure and efficient energy in response to the needs for climate change mitigation actions [4]. Demand for electricity by an end user is currently managed in many jurisdictions through their relationship with an electricity (energy) supplier. Suppliers meet their own total needs by buying from a centrally managed pool. Electricity generators sell the output of their plants to the pool, the electricity can be generated by renewable sources such as wind, or from fossil or nuclear fuels. The electricity is transported from the generators’ sites over the transmission system and finally over the distribution system to the end-users buildings. Approximately 8 - 15% of the power generated is lost through heat loss during transport and distribution This motivates the desire to locate generation nearer to demand sites. The move to more sustainable practices further motivates the focus on increasing the use of Renewable Energy Sources (RES), and decreasing the reliance on fossil and nuclear fuels.

The future decentralised distribution network will be required to facilitate new market practices where certain end-users become electricity generators. Therefore if formerly all the end-users were consumers, now some of them are becoming prosumers. One concept being explored is that of an energy collective to allow participants a more proactive role in power system operation. An energy collective can be viewed as a community-based electricity market structure where prosumers are allowed to share energy at community level [9]. Prosumers may generate more electricity than their needs at some times and may wish to make their excess electricity available to either to their supplier, or in this case, to the local energy community network. At other times they may be self satisfied, or may not produce enough and need to buy electricity from their supplier, or preferably to buy the excess renewable electricity of their neighbours in the energy community network.

The connection topology of traditional centrally controlled electricity grids are generally tree distribution networks. Figure 1 shows the IEEE 37 node radial test feeder topology. The symbol adjacent to node 799 is a type of transformer which acts as on/off switch. The symbol between nodes 709 and 775 is a transformer to control voltage levels. We have added a compass rose to show how we interpret the direction orientation of the test network.

As community energy collectives evolve, upgrade of the local low voltage distribution network may be warranted. The evolution of electricity grid tree topologies mirrors that of telecommunications networks, when connectivity constraints were added to meet reliability concerns. In turn ring bounds can be considered to limit flow (and loss) in network cycles [3, 6]. Many of the (telecommunications) network design models and solution techniques are transferable to address the needs of smart grid topology design. Similar ideas in adapting network topology design models are used in wind farm cabling problems in [5].

An additional challenge to understanding the requirements of future local electricity networks is the move toward the electrification of heat and transport in climate change mitigation actions. These demands will push the demand for electricity upwards

[*] Low Voltage Smart Grid Topologies in urban Areas
[†] UCD Energy Institute & UCD Centre for Business Analytics

ISBN 978-3-89318-079-0 on OpenProceedings, ISSN 2510-7437

DOI: 10.5441/002/inoc.2019.10

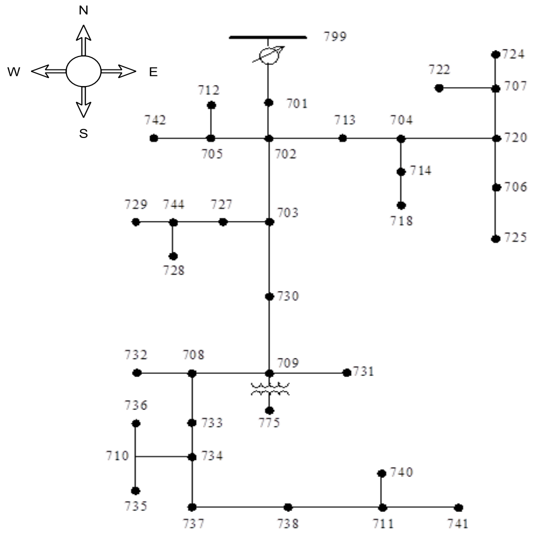


Figure 1: IEEE 37 node radial test network.

and may require significant network reinforcement. In contrast, retrofitting of building with modern (thermo) efficient materials and the use of more efficient white goods counter balance somewhat to decrease electricity (and total energy) demand. Estimates of the uptake of RES further complicate estimates of future network needs. An understanding of potential electricity flow in energy collectives will provide increased understanding for transmission and distribution system operators.

3 SMART GRID LOW VOLTAGE UPGRADE MIP MODEL

Consider an existent electricity low voltage distribution network in an urban area modelled as $G = (N, A)$ for a set of n locations $N = \{1, \dots, n\}$ such that the topology is a tree rooted at a substation n_0. Electricity flows according to the laws of physics and can be controlled by controller devices. Historically electricity flowed from the substation in response to consumer demand so that graphs were considered to be directed. We make some simplifying assumptions to handle nonlinear alternating current flow. Smart wire technology in development may make these assumptions realistic in the near future [8].

Consider that the set $N \backslash n_0$ is partitioned into sets C of the end-users who remain consumers and set P of the new prosumers. Electricity can flow from the prosumer back into the distribution network without the need for additional arcs, the flow can be controlled and monitored by switching devices. Hence we can assume the existent network is modelled by $\bar{G} = (N, E)$

We consider a time horizon T. Each end-user $i \in N \setminus n_0$ consumes a certain amount EC_i^t of energy at time t. We treat the substation root node as a prosumer in the sense that they can both provide and accept electricity. Each prosumer $i \in P$ generates a certain amount EG_i^t of energy at time t. At each time t, the energy demand Q_i^t for each consumer $i \in C$ is $Q_i^t = -EC_i^t$. At each time t, the energy demand Q_i^t for each prosumer $i \in P$ is $Q_i^t = EG_i^t - EC_i^t$.If this value is zero the prosumer is self satisfied. If $Q_i^t > 0$ the prosumer has an excess of electricity and sells electricity to the community network. If $Q_i^t < 0$ the prosumer has insufficient electricity and buys electricity, preferably from the community network but otherwise from their supplier.

We assume that the community network needs can be satisfied. Therefore at each time, the substation node n_0 either provides or accepts energy:

$\sum_{i \in N \backslash \{n_0\}} Q_i^t < 0$ or $\sum_{i \in N \backslash \{n_0\}} Q_i^t > 0$, respectively.

Set $Q_{n_0}^t = \sum_{i \in N \backslash \{n_0\}} Q_i^t$. Set $\bar{Q} = \max_{t \in T} \sum_{i \in N} |Q_i^t|$ to be the maximum amount of electricity transported in any connection.

We take possible energy losses into account. The overall losses between the substation and consumers can be modelled by a percentage loss factor $L \in [8, 15]\%$.

To obtain the MILP model consider the following decision variables:

Topological binary integer variables x_{ij} indicate whether the arc (i, j) is selected to be included in the new decentralised network. Flow variables y_{ij}^t indicate the amount of electricity transported from location i to location j at time t. Let the constants a_{ij} take value 1 if arc $(i, j) \in E$, meaning that it is already installed and belongs to the distribution network, or take value 0 if the arc $(i, j) \notin E$, it is not installed.

$$\min \sum_{(i,j) \in A} c_{ij} x_{ij} + \sum_{t \in T} \sum_{(i,j) \in A} y_{ij}^t \tag{1}$$

subject to

$$\sum_{i \in N} (a_{ij} + x_{ij}) \geq 1 \qquad j \in C \tag{2}$$

$$\sum_{i \in N} (a_{ij} + x_{ij}) \geq 1, \qquad j \in P \tag{3}$$

$$\sum_{i \in N} (a_{ji} + x_{ji}) \geq 1, \qquad j \in P \tag{4}$$

$$\sum_{i \in N} y_{ij}^t + (1 + L) Q_j^t = \sum_{i \in N} y_{ji}^t, \qquad j \in N, \ t \in T \tag{5}$$

$$y_{ij}^t \leq \bar{Q}(a_{ij} + x_{ij}), \qquad (i, j) \in A \tag{6}$$

$$a_{ij} + x_{ij} + a_{ji} + x_{ji} \leq 1, \qquad i, j \in N \tag{7}$$

$$x_{ij} \in \{0, 1\}, \qquad (i, j) \in A \tag{8}$$

$$y_{ij}^t \geq 0, \qquad (i, j) \in A, \ i \in T \tag{9}$$

Let c_{ij} be the cost of installing additional arc $ij \in A$ in the upgrade network. Eq (1) is the objective function which minimises the cost of additional edges in the upgrade as well as minimising the overall flow of electricity. This will have the effect of fostering flow between geographically close neighbours, which in turn reduces transmission losses. Inequality (2) ensures all consumers are connected to the network to receive energy over an existing arc, and possibly through an additional new arc. Inequalities (3) and (4) ensure all prosumers are connected to the network by an existing arc and possibly through an additional new arc. Prosumers have the possibility of both receiving electricity, and of offering their excess to the network. Equalities (5) are the flow conservation constraints and take into account possible energy losses by a percentage factor L, $0.08 \leq L \leq 0.15$. Inequalities (6) are the variables linking constraints and limit for a maximum flow in any connection of the network. Inequalities (7) say we do not install a new arc between two locations if there is an existing link in the network, this means we restrict to one the number of connections between any two locations. Finally constraints (8) and (9) define the variables domain.

In addition, we can add clique inequalities for any subset of consumers. For any clique of size three, $C_3 = \{i, j, k\} \subseteq C$, the following is valid:

$\sum_{\{l,m\} \in C_3} (a_{lm} + x_{lm}) \leq |C_3| - 1 = 2.$

These inequalities say that for any clique $C_c \subset C$ the number of connections is restricted to $|C_c| - 1$. Restricting the number of locations in the clique avoids cycles between any set of consumers. The clique inequalities are inserted for subsets of consumers where the existing arcs are sufficient to ensure the energy flow distribution. Recall that the existing topology is a tree. New arcs are added to improve the energy flow mainly for prosumers that must have the opportunity to distribute their energy in the network. In the case of prosumers, a cycle is allowed in the solution.

4 TEST INSTANCES AND SCENARIOS

We augment the IEEE 37 node test instances with geographic information for two locations; Dublin, Ireland and Aveiro, Portugal. We simply overlay the IEEE system on geographic maps and extract GPS coordinates of the locations. This give us two test instances where we can estimate distances between nodes using the haversine formula as a proxy for c_{ij}.

Dublin, Ireland lies at latitude 53.4°C N and longitude 6.3°C W. It has a temperate climate with pronounced variation between the number of hours of daylight in winter and summer. Hence the amount of electricity generated by PV per season is quite variable [1]. In addition, the east-west orientation of some buildings offers less potential than those with southerly facing aspects. We evaluate two potential seasonal scenarios for the Dublin location; one in summer (with daylight hours 7.00 - 20.00), and the other in winter (with daylight hours 9.00 - 16.00). We create representative load profiles for Dublin using data from the Retail Market Design Service [10] allowing a proportion of the nodes to act as prosumers. Sample profiles are shown in Figure 2.

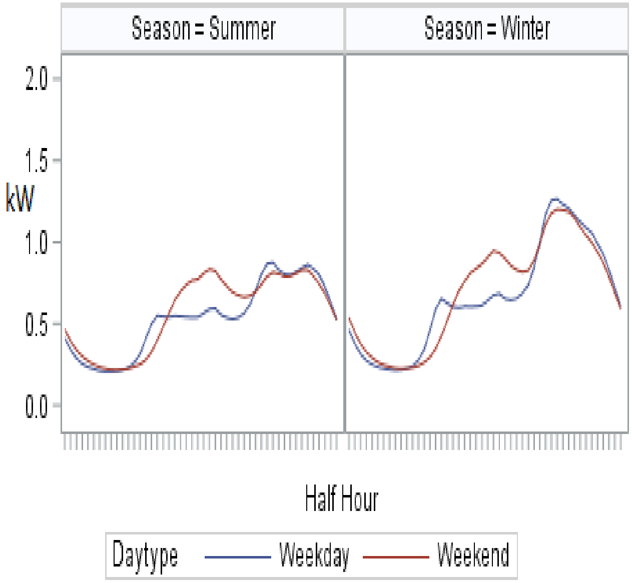


Figure 2: Reference Load Profiles for Ireland.

We follow a similar approach for Aveiro located at 40.6°C N and 8.6°C W and generate sample load profiles informed by [11]. Aveiro offers more consistent daylight hours and sunlight than Dublin, so we test just one reference load profile.

5 RESULTS AND ANALYSIS

The MIP model was implemented in Mosel and computational tests were run using XpressMP 8.5 on a Dell 64 bit Windows 8 machine with Intel i5 3.2 GHz processor and 8GB of Ram. We test the instances varying the estimated transmissions losses L, and the proportion of prosumers in the community. We assume a 2kW PV panel for each prosumer with generation during daylight hours of diminishing output depending on the prosumer's orientation.

We performed computational experiments to assess the performance of the compact model and the quality of the obtained solutions with and without the clique constraints for sets of three consumers. The use of these valid inequalities greatly improves the solution quality, but at a slight expense in computational time. For example, for the instance Aveiro with $L = 8\%$ and $P = 25\%$ the $Gap = (BestMIP - BestBound)/BestMIP$ improves from 0.39 to 0.17. Therefore we use the IP model with the clique valid inequalities for all sets of consumers of size three. We allowed a maximum run time of 3 hours for the more challenging instances.

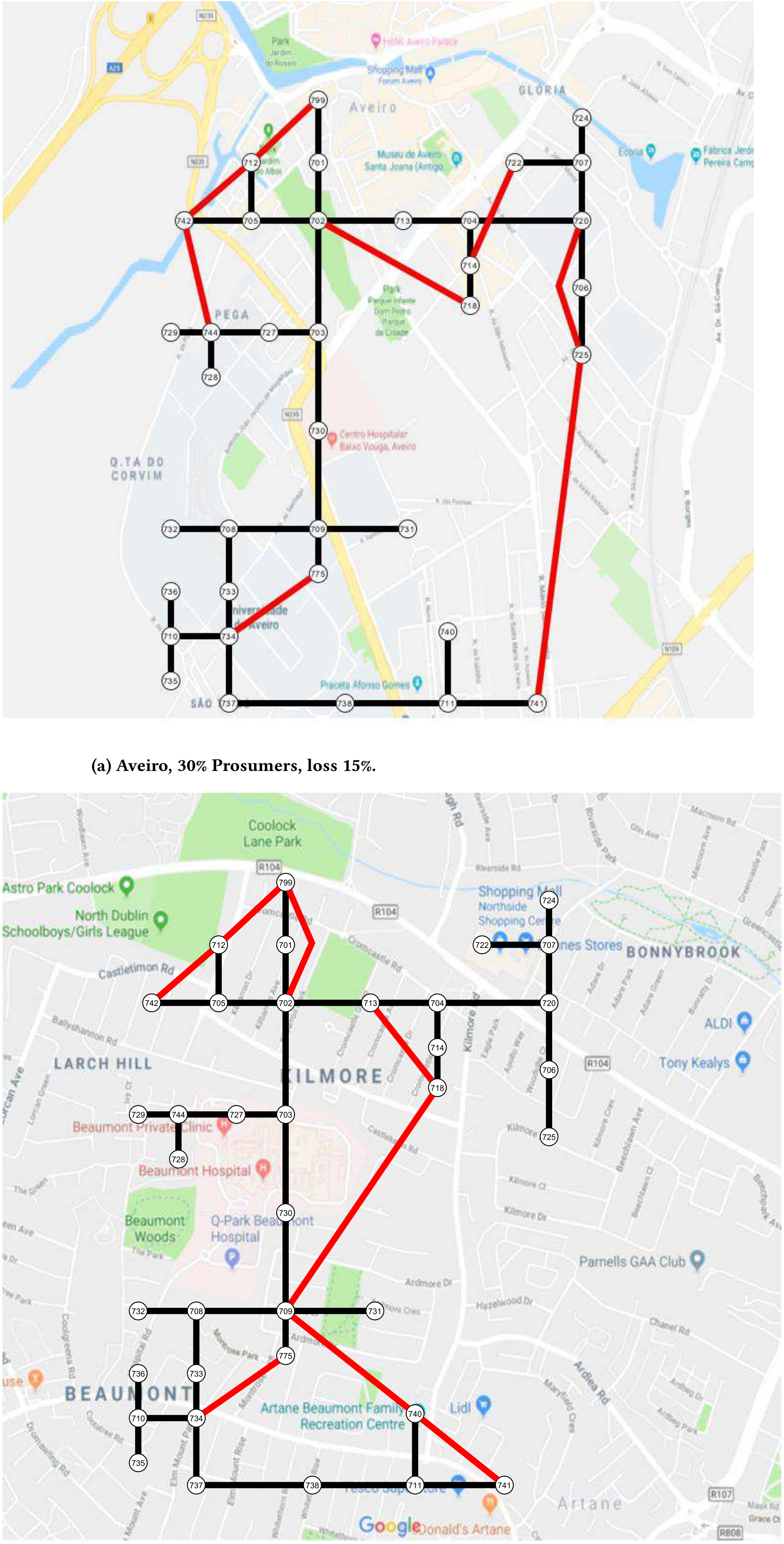
Table 1 shows sample results. From left to right we show the information about the problem instance (Name of the instance, Loss percentage, Prosumers percentage), followed by details of the IP model B&B search obtained for a time limit of three hours (problem status, BestBound value corresponds to the best lower bound obtained, BestMIP value corresponds to the best feasible integer solution, the corresponding Gap value, the number of the nodes in the B&B search procedure, the computational time in seconds and the number of new arc flows to be installed determined by the best MIP solution).

Figure 3 shows sample solution topologies. Existing edges are show in black, and proposed additional arcs in red. We see the evolution from tree to more resilient meshed networks. Figure 3a shows the best solution found for Aveiro with 30% Prosumers, and a loss factor of 15%. Figure 3b shows the best solution found for Dublin with 25% Prosumers, and a loss factor of 8%.

We see that problem instances with a low percentage of prosumers are solved to optimality in relatively short run times. As the proportion of prosumers increases the test instances become more difficult to solve. There is an increase in the solution values as the loss factor increases. The initial LP relaxation is quite weak. The Dublin Winter instances are solved to optimality, and reflect the low availability of excess electricity from prosumers. In contrast, the Aveiro and Dublin Summer instances are more challenging problems when excess electricity from prosumers is availably to satisfy consumers in the community, or to return to the grid via the substation root node.

6 CONCLUSIONS

In this paper we have described a smart grid topology problem which focuses on augmenting the grid topology in order to take into account new demands as some energy consumers become energy producers: prosumers. We propose a MIP model to augment the existent grid topology and identify which potential arcs could be added to support new electricity flows. The computational results show that the run times are short when the



(a) Aveiro, 30% Prosumers, loss 15%.

(b) Dublin, 25% Prosumers, loss 8%

Figure 3: Sample Smart Grid Solutions.

Table 1: Smart Grid Computational Results

Name	Loss%	%Prosumer	Status	Bestbound	BestMIP	Gap	Nodes	Time (s)	♯New Flows
Aveiro	8	20	Optimum	2115	2115	0.00	1	69	2
Aveiro	15	20	Optimum	2219	2219	0.00	0	74	2
Aveiro	8	25	Unfinished	3575	4333	0.17	7548	10885	7
Aveiro	15	25	Unfinished	3673	4320	0.15	9271	10876	5
Aveiro	8	30	Unfinished	3927	5263	0.25	5532	10838	7
Aveiro	15	30	Unfinished	4034	4883	0.17	6700	11200	8
DublinSummer	8	20	Optimum	2305	2305	0.00	265	250	3
DublinSummer	15	20	Optimum	2416	2416	0.00	217	279	3
DublinSummer	8	25	Unfinished	3520	3983	0.12	16750	10836	8
DublinSummer	15	25	Unfinished	3592	4085	0.12	19103	10872	8
DublinSummer	8	30	Unfinished	3731	4367	0.15	14798	10982	13
DublinSummer	15	30	Unfinished	3813	4684	0.19	10216	11095	10
DublinWinter	8	20	Optimum	2994	2994	0.00	1157	956	3
DublinWinter	15	20	Optimum	3159	3159	0.00	2213	1223	3
DublinWinter	08	25	Optimum	4141	4141	0.00	1815	1579	7
DublinWinter	15	25	Optimum	4298	4298	0.00	1929	1786	7
DublinWinter	8	30	Optimum	4351	4351	0.00	3803	5025	8
DublinWinter	15	30	Optimum	4506	4506	0.00	5231	6291	8

percentage of prosumers and Loss factor are low. The problem instances get harder as these parameter values are increased.

Our MIP model yields interesting results that could be used by distribution system operators and energy collectives to explore the potential of solar PV to meet RES targets and sustainability objectives. Our models could be used to perform cost benefit analysis of upgrades, or to understand potential electricity exchange flows in the network, and to understand where devices to control and record the electricity flows may need to be added.

There is potential for future work to improve the MIP model with additional valid inequalities. Other variants of the model could focus on rewarding prosumers with a higher price for excess electricity shared among the energy community, compared with excess returned to the grid via the substation root node, or alternatively new reverse arcs from prosumers to the substation may not be considered. In our computational experiments, we used a distance measure as a proxy for the arc installation costs c_{ij} in Eq (1), and no financial penalty or reward for flows within the community network. Further evaluation of the model could test weightings and alternative costs of the objective function components. In addition, since the arc installation costs substantially exceed network flow costs, a hierarchical model could provide a useful alternative to evaluate potential scenarios.

In our computational tests, we allowed a certain proportion of the nodes to act as prosumers and assumed a 2kW panel/prosumer. In further testing we may choose to only allow those nodes with high potential for solar PV to act as prosumers, i.e., those nodes with south or west facing orientations should be selected to serve the energy community, rather than those with east-west orientations, and decisions on the size of the PV panel could be considered. Such choices are of interest to policy makers and give rise to questions on the social acceptance of energy community designs.

Finally, as noted, the resulting meshed networks are more resilient, but give rise to more complex management problems such as those seen in works on bounded rings in telecommunications networks. A research agenda in the network optimisation community to share and exploit its learnings on network design and evolution could help advance the energy transformation and provides many interesting research opportunities.

ACKNOWLEDGMENTS

The research of C. Requejo was partially supported by the Fundação para a Ciência e a Tecnologia (FCT), project:
UID/MAT/04106/2019 (CIDMA).

REFERENCES

[1] LM Ayompe, Aidan Duffy, SJ McCormack, and Michael Conlon. 2011. Measured performance of a 1.72 kW rooftop grid connected photovoltaic system in Ireland. *Energy conversion and management* 52, 2 (2011), 816–825.

[2] Francesco Balduzzi, Alessandro Bianchini, Ennio Antonio Carnevale, Lorenzo Ferrari, and Sandro Magnani. 2012. Feasibility analysis of a Darrieus vertical-axis wind turbine installation in the rooftop of a building. *Applied Energy* 97 (2012), 921 – 929. https://doi.org/10.1016/j.apenergy.2011.12.008 Energy Solutions for a Sustainable World - Proceedings of the Third International Conference on Applied Energy, May 16-18, 2011 - Perugia, Italy.

[3] Paula Carroll, Bernard Fortz, Martine Labbé, and Seán McGarraghy. 2013. A branch-and-cut algorithm for the ring spur assignment problem. *Networks* 61, 2 (2013), 89–103.

[4] COM. 2016. *Clean Energy For All Europeans.* Technical Report 860. European Union. https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:52011DC0885&rid=4 accessed January 2018.

[5] Martina Fischetti and David Pisinger. 2018. Optimal wind farm cable routing: Modeling branches and offshore transformer modules. *Networks* (2018). https://doi.org/10.1002/net.21804

[6] Bernard Fortz, Martine Labbé, and Francesco Maffioli. 2000. Solving the Two-Connected Network with Bounded Meshes Problem. *Operations Research* 48, 6 (2000), 866–877.

[7] William H Kersting. 1991. Radial distribution test feeders. *IEEE Transactions on Power Systems* 6, 3 (1991), 975–985.

[8] Frank Kreikebaum, Debrup Das, Yi Yang, Frank Lambert, and Deepak Divan. 2010. Smart Wires âĂŤ A distributed, low-cost solution for controlling power flows and monitoring transmission lines. In *2010 IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT Europe).* IEEE, 1–8.

[9] F. Moret and P. Pinson. 2018. Energy Collectives: a Community and Fairness based Approach to Future Electricity Markets. *IEEE Transactions on Power Systems* (2018), 1–1. https://doi.org/10.1109/TPWRS.2018.2808961

[10] Retail Market Design Service (RMDS). 2019. *Standard Load Profiles.* Technical Report. RMDS. https://rmdservice.com/standard-load-profiles/ Accessed February 2019.

[11] Daniel Wiesmann, InÃłs Lima Azevedo, Paulo FerrÃčo, and John E. FernÃąndez. 2011. Residential electricity consumption in Portugal: Findings from top-down and bottom-up models. *Energy Policy* 39, 5 (2011), 2772 – 2779. https://doi.org/10.1016/j.enpol.2011.02.047

On Optimization of Semi-stable Routing in Multicommodity Flow Networks

Artur Tomaszewski
Institute of Telecommunications,
Warsaw University of Technology
Warsaw, Poland
a.tomaszewski@tele.pw.edu.pl

Michał Pióro
Institute of Telecommunications,
Warsaw University of Technology
Warsaw, Poland
m.pioro@tele.pw.edu.pl

Davide Sanvito
Dipartimento di Elettronica,
Informazione e Bioingegneria,
Politecnico di Milano
Milan, Italy
davide.sanvito@polimi.it

Ilario Filippini
Dipartimento di Elettronica,
Informazione e Bioingegneria,
Politecnico di Milano
Milan, Italy
ilario.filippini@polimi.it

Antonio Capone
Dipartimento di Elettronica,
Informazione e Bioingegneria,
Politecnico di Milano
Milan, Italy
antonio.capone@polimi.it

ABSTRACT

Ideally, the network should be dynamically reconfigured as traffic evolves. Unfortunately, even in SDN paradigm, network reconfigurations cannot be too frequent due to a number of reasons related to route stability, forwarding rules instantiation, individual flows dynamics, traffic monitoring overhead, etc.

In this paper, we focus on the fundamental problem of deciding whether, when, and how to reconfigure the network during traffic evolution. We consider a problem of optimizing semi-stable routing in the capacitated multicommodity flow network when one may use at most a given maximum number of routing configurations (called clusters) and when each routing configuration must be used for at least a given minimum amount of time.

We propose a solution method based on cluster generation that provides a good lower bound on the minimum network delay (i.e., the total of link delays) and scales well with the size of the network.

1 INTRODUCTION

The dynamic nature of network traffic caused by daily fluctuations is the origin of a crucial trade-off between routing optimality and frequency of network reconfiguration. Nevertheless, network operators have traditionally privileged routing stability by resorting to approaches, like oblivious routing [1] and robust routing [9, 15, 16], that apply static routing designs based on“worst case” traffic conditions. This unavoidably creates overprovisioning and suboptimal utilisation of network capacity.

Recently, Software-Defined Networking (SDN) has provided tools for making online network reconfiguration a potentially viable solution: dynamic routing reconfigurations can be applied at the network devices to optimize performance as the traffic evolves [4, 7, 8, 12]. However, reconfiguring the network too frequently can in general affect its stability since reprogramming flow rules can take longer than the reconfiguration period.

A group of hybrid approaches [2, 5, 13, 14, 17], often referred to as semi-stable routing, have been recently proposed to combine static and dynamic routing. Considering a limited set of routing configurations, each designed and activated during specific time intervals, allows for reducing the penalty of using the “worst case” traffic conditions, and, simultaneously, for controlling the reconfiguration frequency. As a result, the optimization problem of selecting a sequence of routing configurations, and timepoints when the consecutive routing configurations must be activated, arises.

In this paper we consider the problem of optimizing routing in the capacitated multicommodity flow network, in which demand volumes change periodically over an ordered set of timepoints. Following the semi-stable routing approach, we analyse a specific version of the problem where one may use at most a given maximum number of routing configurations and where each routing configuration must be used for at least a given minimum number of consecutive timepoints, in order to meet the maximum network reconfiguration frequency constraint. Referring to a set of consecutive timepoints as a (timepoint) cluster, we name this problem the semi-stable routing cluster design problem (SSRCDP). In SSRCDP the optimization objective is to minimize the network delay, i.e., the sum of timepoint delays (over all timepoints) where for a single timepoint its delay is defined as the sum of the link delays. Although we have chosen the delay metric, the solution method we propose is general enough to cope with other types of the congestion metric.

The works on semi-stable routing available in the literature usually exhibit one of the following limitations: (i) they ignore the time domain by not providing any limit on the reconfiguration rate [2, 14, 17], (ii) the number of created clusters is limited and reconfiguration timepoints are arbitrary [2, 5]. Other semi-stable approaches have more recently been proposed to overcome these limitations [3, 11]. In particular, the techniques presented there compute a set of routing configurations that can be combined together to generate a routing configuration for a new traffic realization. However, combining multiple configurations may generate a large number of paths and flow split ratios that might not be feasible to handle by network devices.

For SSRCDP we propose a solution method based on cluster generation that delivers provably near-optimal solutions, i.e., it also provides a good lower bound of the network delay. In addition, this method scales well with the size of the network and can be effectively applied to networks of large sizes. The problem formulation, the solution method, and an illustrative realistic numerical example are presented below.

ISBN 978-3-89318-079-0 on OpenProceedings, ISSN 2510-7437

DOI: 10.5441/002/inoc.2019.11

2 PROBLEM FORMULATION

The notation used in the paper, summarized in Table 1, is as follows. Let the capacitated multicommodity flow network be modeled with a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{D})$, where $\mathcal{V}$ is the set of nodes and $\mathcal{E}$ is the set of (directed) links (where $c(e) \geq 0$, $e \in \mathcal{E}$, is the capacity of link e). $\mathcal{D}$ is the set of (directed) demands, where $o(d), t(d)$, $d \in \mathcal{D}$, are the originating and terminating node, respectively, of demand d. Next, let $\mathcal{P}(d)$ be a given set of (routing) paths in graph $\mathcal{G}$ that are admissible for demand d, $d \in \mathcal{D}$, (each path $p \in \mathcal{P}(d)$ connects the demand's origin $o(d)$ with its termination $t(d)$). (Below, $\mathcal{P}$ will denote the set of all admissible paths, i.e., $\mathcal{P} := \bigcup_{d \in \mathcal{D}} \mathcal{P}(d)$.) Additionally, let $Q(e,d) \subseteq \mathcal{P}(d)$, $e \in \mathcal{E}, d \in \mathcal{D}$, denote the set of admissible paths of demand d that use link e. Finally, let $\mathcal{T} := \{0, 1, \ldots, T-1\}$ be the set of consecutive *timepoints*, and let $h(d,t) \geq 0$, $d \in \mathcal{D}, t \in \mathcal{T}$, be the volume of demand d to be realized at timepoint t.

We assume that a *routing configuration* is defined by vector $x := (x_{dp})_{d \in \mathcal{D}, p \in \mathcal{P}(d)}$, where x_{dp} is the fraction (i.e., $x_{dp} \in [0,1]$) of the volume of demand d that is assigned to path p. The following condition must thus hold:

$$\textstyle\sum_{p \in \mathcal{P}(d)} x_{dp} = 1 \quad d \in \mathcal{D}. \tag{1}$$

Then, if routing configuration x is used at timepoint $t \in \mathcal{T}$, the *utilization* $w_e^t(x)$ of link e at t is defined as:

$$w_e^t(x) := \tfrac{1}{c(e)} \textstyle\sum_{p \in Q(e,d)} h(d,t) x_{dp} \quad e \in \mathcal{E}. \tag{2}$$

Note that the quantity $\sum_{p \in Q(e,d)} h(d,t) x_{dp}$ in the right-hand side of definition (2) expresses the load of link e at timepoint t. Further, let $F : [0, +\infty) \to [0, +\infty)$ be an increasing convex piece-wise linear function with $F(0) = 0$. We will call $F(w)$ the *delay function* (see [6, 10]) as it is supposed to measure the packet delay on a link for a given link utilization w. Finally, the quantity

$$z^t(x) := \textstyle\sum_{e \in \mathcal{E}} F(w_e^t(x)) \tag{3}$$

will be called the *timepoint delay* at timepoint t.

We may now introduce the notion of a *cluster* $C(t,l)$ with parameters t (timepoint in which the cluster starts) and l (length of the cluster). Namely, $C(t,l)$ is the set of l consecutive timepoints that starts at timepoint t. Hence, $C(t,l) := \{t, t \oplus 1, \ldots, t \oplus (l-1)\}$, where $\oplus$ denotes addition modulo T (i.e., the timepoints are counted modulo T). For a given cluster $C = C(t,l)$, let $t(C) = t$ and $l(C) = l$ denote, respectively, the start and the length of C.

Suppose that the same routing configuration (denoted by $x(C) = (x(C)_{dp})_{d \in \mathcal{D}, p \in \mathcal{P}(d)})$ is used for all timepoints of cluster C. Then, we will call C a *(stable) routing cluster*. For a routing cluster C and a given routing configuration x, the quantity

$$z(C,x) := \textstyle\sum_{t \in C} z^t(x) \tag{4}$$

will be referred to as *cluster delay* (of cluster C under routing configuration x). The minimum cluster delay (i.e., the value of $z(C,x)$ minimized over all routing configurations x will be denoted by $Z(C)$.

The *semi-stable routing cluster design problem* (SSRCDP) we consider is this: given $\mathcal{G}, \mathcal{P}, \mathcal{D}, \mathcal{T}$, and a pair of positive integer numbers $N \leq T$ and $L \leq T$, find a *partition* $\mathscr{R}$ of the set of timepoints $\mathcal{T}$ into at most N (non-empty) routing clusters, each of length at least L (i.e., $|\mathcal{R}| \geq L$, $\mathcal{R} \in \mathscr{R}$), and find a routing configuration $x(\mathcal{R})$ for each routing cluster $\mathcal{R} \in \mathscr{R}$, so as to minimize the *network delay* $Z(\mathscr{R}) := \sum_{\mathcal{R} \in \mathscr{R}} Z(\mathcal{R})$. In the following, the minimum value of the total maximal network utilization resulting from SSRCDP will be denoted by Z^*. Note that the assumptions on N, L, T imply that $N \leq \frac{T}{L}$, and hence $N \leq \lfloor \frac{T}{L} \rfloor$.

3 SOLUTION METHOD

3.1 The fixed partition subcase

We start with the following observation. If the sets forming a partition $\mathscr{R}$ of set $\mathcal{T}$ were given and fixed, SSRCDP would reduce to finding a routing configuration $x(\mathcal{R})$ minimizing $Z(\mathcal{R})$ for each cluster $\mathcal{R} \in \mathscr{R}$, and this could be done independently for each cluster. Thus, we first analyse the problem of finding an optimal routing configuration for a given cluster. We aim, in particular, at deriving some properties that can be useful in formulating and solving the original semi-stable routing cluster design problem.

Finding an optimal routing configuration for a given set of (not necessarily consecutive) timepoints $\mathcal{U} \subseteq \mathcal{T}$ is identical to a well-known problem of finding an optimal routing configuration for a given set of traffic matrices. Such a *routing problem* (denoted by RP($\mathcal{U}$)) consists in finding a single routing configuration $x(\mathcal{U})$ that minimizes the sum of timepoint delays over $\mathcal{U}$:

Problem RP($\mathcal{U}$)

$$Z(\mathcal{U}) = \min \textstyle\sum_{t \in \mathcal{U}} \big(\sum_{e \in \mathcal{E}} z_e^t \big) \tag{5a}$$

$$\textstyle\sum_{p \in \mathcal{P}(d)} x_{dp} = 1 \qquad d \in \mathcal{D} \tag{5b}$$

$$w_e^t \geq \tfrac{1}{c(e)} \textstyle\sum_{p \in Q(e,d)} h(d,t) x_{dp} \qquad t \in \mathcal{U},\ e \in \mathcal{E} \tag{5c}$$

$$z_e^t \geq a(k) w_e^t + b(k) \qquad t \in \mathcal{U},\ e \in \mathcal{E},\ k \in \mathcal{K} \tag{5d}$$

$$x_{dp} \in [0,1] \qquad d \in \mathcal{D},\ p \in \mathcal{P}(d) \tag{5e}$$

$$z_e^t, w_e^t \in \mathbb{R} \qquad t \in \mathcal{U},\ e \in \mathcal{E}. \tag{5f}$$

Above, variables x_{dp}, $d \in \mathcal{D}$, $p \in \mathcal{P}(d)$, define a routing configuration $x(\mathcal{U})$ common for all timepoints in $\mathcal{U}$, variables w_e^t, $t \in \mathcal{U}$, $e \in \mathcal{E}$, express link utilizations at the timepoints in $\mathcal{U}$, and variables z_e^t, $t \in \mathcal{U}$, $e \in \mathcal{E}$, specify the corresponding link delays. In (5d), parameters $a(k), b(k)$, $k \in \mathcal{K} := \{1, 2, \ldots, K\}$, define the delay function $F(z) := \max\{a(k)z + b(k) : k \in \mathcal{K}\}$, where $b(1) = 0 > b(2) > \ldots > b(K)$, $0 < a(1) < a(2) < \ldots < a(K)$.

Note that RP($\mathcal{U}$) is a linear programming (LP) problem in a non-compact formulation that can be easily solved to optimality (even for large networks) using the column (path) generation approach based on a shortest path algorithm: to generate a new path $p \in \mathcal{P}(d)$ for demand $d \in \mathcal{D}$ and price out a new variable x_{dp} one has to find a shortest path in graph $\mathcal{G}$ between the end nodes of d, with the costs of links equal to $\frac{1}{c(e)} \sum_{t \in \mathcal{U}} h(d,t) \pi_e^t$, $e \in \mathcal{E}$, where π_e^t are optimal dual variables associated with constraint (5c). A path is added to the problem if its cost is less than λ_d – optimal dual variable associated with constraint (5b). Observe that RP($\mathcal{U}$) can alternatively be formulated as an LP problem in a compact way, using the node-link notation with link flows (instead of path flows) that does not require column generation.

We end this section with the following observation.

Remark 1. *For any two sets $\mathcal{U}', \mathcal{U}$ such that $\mathcal{U}' \subseteq \mathcal{U} \subseteq \mathcal{T}$, the inequality*

$$Z(\mathcal{U}') \leq \textstyle\sum_{t \in \mathcal{U}'} z^t(x^*(\mathcal{U})) \tag{6}$$

holds, where $x^*(\mathcal{U})$ and is the optimal routing configuration resulting from RP($\mathcal{U}$), and $z^t(x^*(\mathcal{U}))$, $t \in \mathcal{U}$, are defined by (2)*. The reason is that if $Z(\mathcal{U}')$ would be larger than $\sum_{t \in \mathcal{U}} z^t(x^*(\mathcal{U}))$, then the routing configuration $x^*(\mathcal{U})$, when applied to $\mathcal{U}'$, would decrease the value of $Z(\mathcal{U}')$. Clearly, when $\mathcal{U} = \mathcal{U}'$ then the right hand side of* (6) *is equal to $Z(\mathcal{U}')$.*

Table 1: Notation

Notation	Description
$\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{D})$	network graph, $\mathcal{V}$ – set of nodes, $\mathcal{E}$ – set of (directed) links, $\mathcal{D}$ – set of (directed) demands
$\mathcal{T} = \{0, 1, \dots, T-1\}$	set of timepoints
$c(e)$	capacity of link e ($e \in \mathcal{E}$)
$h(d, t)$	volume of demand d to be realized in timepoint t ($d \in \mathcal{D},\ t \in \mathcal{T}$)
$o(d), t(d)$	originating node and terminating node, respectively, of demand $d \in \mathcal{D}$
$\mathcal{P}(d)$	set of admissible (routing) paths for demand $d \in \mathcal{D}$
$Q(e, d)$	set of paths in $\mathcal{P}(d)$ that contain link e ($e \in \mathcal{E},\ d \in \mathcal{D}$)
$\mathcal{P} = \bigcup_{d \in \mathcal{D}} \mathcal{P}(d)$	set of all admissible paths
$x = (x_{dp})_{d \in \mathcal{D}, p \in \mathcal{P}(d)}$	routing configuration (vector of path flows)
$w_e^t(x), F(w_e^t(x))$	utilization of link e at timepoint t and the corresponding delay
$z^t(x)$	timepoint delay (sum of link delays at timepoint $t \in \mathcal{T}$) implied by routing configuration x
C	(routing) clusters composed of timepoints
$t(C), l(C)$	starting timepoint and length (respectively) of cluster C ($t(C) \in \mathcal{T},\ l(C) \in \{1, 2, \dots, T\}$)
$C(t, l)$	cluster with $t(C) = t,\ l(C) = l,\ C(t, l) = \{t, t \oplus 1, \dots, t \oplus (l-1)\}$ ($\oplus$ denotes addition modulo T)
$\mathscr{C}$	family of (routing) clusters ($C \in \mathscr{C}$)
$x(C)$	routing configuration used in routing cluster C
$z(C, x) = \sum_{t \in C} z^t(x)$	cluster delay for cluster C with routing configuration x
$Z(C)$	cluster delay of C minimized over all routing configurations x ($Z(C)$ is a solution of RP(C))
$Z(C\|\infty) = \sum_{t \in C} Z(\{t\})$	a lower bound for $Z(C)$
$\mathscr{R}$	family of routing clusters forming a partition of the set of timepoints $\mathcal{T}$ into at most N ($1 \le N \le \frac{T}{L}$) routing clusters, each of length at least L ($\|\mathcal{R}\| \ge L,\ \mathcal{R} \in \mathscr{R}$)
$z(\mathscr{R}) = \sum_{\mathcal{R} \in \mathscr{R}} z(\mathcal{R}, x(\mathcal{R}))$	network delay for partition $\mathscr{R}$ (with routing configurations $x(\mathcal{R}),\ \mathcal{R} \in \mathscr{R}$)
$Z(\mathscr{R}) = \sum_{\mathcal{R} \in \mathscr{R}} Z(\mathcal{R})$	minimum network delay for partition $\mathscr{R}$
SSRCDP	semi-stable routing design problem
Z^*	minimum of $Z(\mathscr{R})$ over all partitions $\mathscr{R}$ (Z^* is the optimal solution value of SSRCDP)
RP($\mathcal{U}$)	routing problem for $\mathcal{U} \subseteq \mathcal{T}$ (finding routing configuration realizing $Z(\mathcal{U})$)
APP($\mathscr{C}$)	approximative partitioning problem using control cluster family $\mathscr{C}$
$\mathscr{R}(\mathscr{C})$	family of routing clusters solving APP($\mathscr{C}$)
$Y(\mathscr{C})$	minimum objective value of APP($\mathscr{C}$) (lower bound for SSRCDP)
CGA	cluster generation algorithm
$\mathbb{B}, \mathbb{Z}_+, \mathbb{R}_+$	$\mathbb{B} = \{0, 1\}$, $\mathbb{Z}_+ = \{0, 1 \dots\}$, $\mathbb{R}_+$ non-negative real numbers

3.2 Approximation problem

The suboptimal approach to SSRCDP presented below consists in formulating an optimization problem that determines a suboptimal partition of the set of timepoints $\mathcal{T}$ into a family $\mathscr{R}$ of clusters, where for each $\mathcal{R} \in \mathscr{R}$, an optimal routing configuration $x^*(\mathcal{R})$ will then be found by solving problem RP($\mathcal{R}$).

Let u^t ($t \in \mathcal{T}$) be a binary variable that equals 1 if, and only if, t is a start of a routing cluster, and 0 otherwise, and let y^t ($t \in \mathcal{T}$) be a continuous variable that approximates (from below) the minimum timepoint delay at t. Let $\mathscr{C}$ be a fixed subfamily of the family of all timepoint clusters (below the family $\mathscr{C}$ will be called a *control family* of *control clusters*), and let $Z(C|\infty) := \sum_{t \in C} Z(\{t\})$ for each $C \in \mathscr{C}$.

The *approximate partitioning problem* APP($\mathscr{C}$) of finding a partition $\mathscr{R}$ of the set of timepoints $\mathcal{T}$ into routing clusters that minimizes the *approximated* network delay is as follows:

Problem APP($\mathscr{C}$)

$$\begin{aligned}
& Y(\mathscr{C}) = \min \textstyle\sum_{t \in \mathcal{T}} y^t && && \text{(7a)}\\
& \textstyle\sum_{t \in \mathcal{T}} u^t \le N && && \text{(7b)}\\
& \textstyle\sum_{0 \le k \le L-1} u^{t \oplus k} \le 1 && t \in \mathcal{T} && \text{(7c)}\\
& U^C = \textstyle\sum_{1 \le k < l(C)} u^{t(C) \oplus k} && C \in \mathscr{C} && \text{(7d)}\\
& Y^C = \textstyle\sum_{t \in C} y^t && C \in \mathscr{C} && \text{(7e)}\\
& y^t \ge Z(\{t\}) && t \in \mathcal{T} && \text{(7f)}\\
& Y^C \ge Z(C) + \big(Z(C|\infty) - Z(C)\big) \cdot U^C && C \in \mathscr{C} && \text{(7g)}\\
& u^t \in \mathbb{B},\ y^t \in \mathbb{R}_+ && t \in \mathcal{T} && \text{(7h)}\\
& U^C \in \mathbb{Z}_+,\ Y^C \in \mathbb{R}_+ && C \in \mathscr{C}. && \text{(7i)}
\end{aligned}$$

Constraints (7b) and (7c) guarantee that each feasible binary vector $u := (u^t)_{t \in \mathcal{T}}$ specifies a partition of the set of timepoints $\mathcal{T}$ which contains at most N clusters, each of length at least L. Let us denote such a partition by $\mathscr{R}$. Then, constraint (7d) defines integer variables U^C that specify with how many clusters in family $\mathscr{R}$ a given cluster C from family $\mathscr{C}$ intersects. Note that when $U^C = 0$ then C intersects with only one cluster in $\mathscr{R}$, when $U^C = 1$ then C intersects with exactly two clusters in $\mathscr{R}$, and so on. Additionally, constraint (7e) defines an approximated cluster delay for each control cluster C.

Constraints (7f) and (7g) specify two kinds of *valid inequalities*, i.e., inequalities that are satisfied by the maximal link utilizations $z^t(x(\mathcal{R}))$, $\mathcal{R} \in \mathscr{R}, t \in \mathcal{R}$, determined (through definition (3)) by any partition $\mathscr{R}$ and any set of routing configurations $x(\mathcal{R})$, $\mathcal{R} \in \mathscr{R}$ (satisfying condition (1)).

The inequality in constraint (7f) holds since $Z(\{t\})$, as the optimal solution of RP($\{t\}$), provides the absolute lower bound on the timepoint delay for any given $t \in \mathcal{T}$. Thus, (7f) is a valid inequality. Note also, that (7f) implies that $\sum_{t \in C} y^t \ge Z(C|\infty)$.

Now observe that the right hand side of inequality in (7g) defines an affine function of variable U^C (defined by (7d)). Let us denote this function by A. Since $Z(C) \ge Z(C|\infty)$ (by definition of $Z(C|\infty)$), function A is non-increasing, and in fact strictly decreasing when $Z(C) > Z(C|\infty)$. Since $A(0) = Z(C)$, for $U^C = 0$ the inequality in (7g) reduces to $\sum_{t \in C} y^t \ge Z(C)$. Moreover, condition $U^C = 0$ means that $C \subseteq \mathcal{R}$ for some $\mathcal{R} \in \mathscr{R}$, and hence, by Remark 1, implies inequality $\sum_{t\in C} z^t(x(\mathcal{R})) \geq Z(C)$. This means that for $U^C = 0$ the inequality in (7g) is valid.

Next, since $A(1) = Z(C|\infty)$, for $U^C = 1$, inequality in (7g) reduces to $\sum_{t\in C} y^t \geq Z(C|\infty)$, which, as mentioned above, is already implied by (7f). This means that in this case (7g) is valid as well. Moreover, since A is non-increasing, $A(U) \leq A(1)$ for $U > 1$ and this means that (7g) is valid for all $U^C > 1$. Thus, (7g) is valid for all possible values of U^C, and this finally implies that APP($\mathscr{C}$) is a relaxation of SSRCDP so that its optimal solution value $Y(\mathscr{C})$ is a lower bound for the minimum network delay Z^*.

Observe that the reason for using the particular form of the inequality in (7g) is that it is stronger than inequality

$$\textstyle\sum_{t\in C} y^t \geq Z(C)\left(1 - U^C\right) \qquad C \in \mathscr{C} \tag{8}$$

as far as the linear relaxation of APP($\mathscr{C}$) is concerned.

In order to find a (suboptimal) solution of SSRCDP we can first solve APP($\mathscr{C}$) for a given control family $\mathscr{C}$, for example for the family of all clusters with length not greater than L. Then, we can solve the routing problem RP($\mathcal{R}$) for each $\mathcal{R} \in \mathscr{R}(\mathscr{C})$, where $\mathscr{R}(\mathscr{C})$ denotes the partition of $\mathcal{T}$ resulting from solving APP($\mathscr{C}$), and determine $Z(\mathscr{R}(\mathscr{C}))$, i.e., the minimum of the network delay for partition $\mathscr{R}(\mathscr{C})$. An issue is, however, how to find a way for extending the current family $\mathscr{C}$ in order to decrease the so obtained $Z(\mathscr{R}(\mathscr{C}))$. The following three basic properties of formulation APP($\mathscr{C}$) will help to resolve this issue.

Property 1. *Let $\mathscr{C}$ be an arbitrary family o clusters for the set of timepoints $\mathcal{T}$. For any partition $\mathscr{R}$ of $\mathcal{T}$ into at most N routing clusters with length at least L each, there exists a feasible solution $u = (u^t)_{t\in\mathcal{T}}, y = (y^t)_{t\in\mathcal{T}}$ of problem APP($\mathscr{C}$) that defines the partition $\mathscr{R}$ and such that for each $\mathcal{R} \in \mathscr{R}$, $y^t = z^t(x(\mathcal{R}))$, $t \in \mathcal{R}$, i.e., y^t is equal to the timepoint delay at t implied by the routing scheme $x(\mathcal{R})$ of the routing cluster $\mathcal{R}$.*

Proof. For each $t \in \mathcal{T}$ we put $u^t = 1$ if $t = t(\mathcal{R})$ for some $\mathcal{R} \in \mathscr{R}$; otherwise, we put $u^t = 0$. Clearly, the so obtained vector u satisfies constraints (7b), (7c) and uniquely defines the partition $\mathscr{R}$. Also, the vector y specified in the thesis of the proposition is feasible for APP($\mathscr{C}$) since, as explained above, inequalities (7f) and (7g) are valid for any routing family $\mathscr{R}$ in question. □

Property 2. *Let $\mathscr{R}(\mathscr{C})$ be the family of clusters determined by an optimal solution of APP($\mathscr{C}$), i.e., by u^*. Then,*

$$Y(\mathscr{C}) \leq Z^* \leq Z(\mathscr{R}(\mathscr{C})), \tag{9}$$

where $Y(\mathscr{C}) = \sum_{t\in\mathcal{T}} y^{t*}$ is the optimal objective of APP($\mathscr{C}$), Z^* is the optimal objective of SSRCDP (i.e, the minimum network delay), and $Z(\mathscr{R}(\mathscr{C})) = \sum_{\mathcal{R}\in\mathscr{R}(\mathscr{C})} Z(\mathcal{R})$.

Proof. Inequality $Y(\mathscr{C}) \leq Z^*$ holds because APP($\mathscr{C}$) is a relaxation of SSRCDP. The second inequality ($Z^* \leq Z(\mathscr{R}(\mathscr{C}))$) holds because partition $\mathscr{R}(\mathscr{C})$ with optimized clusters' routing configurations is a feasible solution of SSDRP. □

Property 3. *Let $\mathscr{R}(\mathscr{C})$ denote an optimal partition resulting from APP($\mathscr{C}$) and suppose that $\mathscr{R}(\mathscr{C})$ is a subset of $\mathscr{C}$. Then $Z(\mathscr{R}(\mathscr{C}))$ is an optimal solution of SSRCDP.*

Proof. Consider the vectors u, y defined for partition $\mathscr{R}(\mathscr{C})$ as in Proposition 1, where $x(\mathcal{R})$ is a routing configuration optimized for each routing cluster $\mathcal{R} \in \mathscr{R}(\mathscr{C})$ by means of RP($\mathcal{R}$). By Proposition 1, the solution u, y is feasible for APP($\mathscr{C}$). We will show that it is also optimal. Consider an arbitrary routing cluster $\mathcal{R} \in \mathscr{R}(\mathscr{C})$ and note that among the inequalities in (7g) that involve variables y^t, $t \in \mathcal{R}$, the one corresponding to $C = \mathcal{R}$ is satisfied tightly since, by assumption, $\sum_{t\in\mathcal{R}} y^t = Z(\mathcal{R})$. Since for each $C' \subset \mathcal{R}$ (whether or not C' is in $\mathscr{C}$), the inequality $\sum_{t\in C'} y^t \geq Z(C')$ holds (by Remark 1), we conclude that vector y is optimal for APP($\mathscr{C}$), and hence $Y(\mathscr{C}) = \sum_{t\in\mathcal{T}} y^t = \sum_{\mathcal{R}\in\mathscr{R}} \sum_{t\in\mathcal{R}} y^t = \sum_{\mathcal{R}\in\mathscr{R}} Z(\mathcal{R})$. Thus, by (9), $Z(\mathscr{R}(\mathscr{C})) = Z^*$. □

3.3 Cluster generation algorithm

The above properties suggest the following algorithm for solving SSRCDP.

CGA: cluster generation algorithm

```
    Step 0: Specify an initial family of clusters $\mathscr{C}$.
    Step 1: Solve APP($\mathscr{C}$) to obtain $\mathscr{R}(\mathscr{C})$ and $Y(\mathscr{C})$. Compute
    $Z(\mathscr{R}(\mathscr{C}))$ by solving RP($\mathcal{R}$) for each $\mathcal{R} \in \mathscr{R}(\mathscr{C})$.
    Step 2: If $\mathscr{R}(\mathscr{C}) \subseteq \mathscr{C}$ or $\frac{Z(\mathscr{R}(\mathscr{C}))-Y(\mathscr{C})}{Y(\mathscr{C})} \leq \varepsilon$ then stop:
    $\mathscr{R}(\mathscr{C})$ is suboptimal (or even optimal) family of routing
    clusters solving SSRCDP (where for each $\mathcal{R} \in \mathscr{R}$ its routing is optimized by RP($\mathcal{R}$)).
    Step 3: $\mathscr{C} \leftarrow \mathscr{C} \cup \mathscr{R}(\mathscr{C})$ and go to Step 1.
```

If in Step 2 the condition $\mathscr{R}(\mathscr{C}) \subseteq \mathscr{C}$ is fulfilled then the routing family $\mathscr{R}(\mathscr{C})$ delivered by CGA is optimal and $Z(\mathscr{R}(\mathscr{C}))$ is the optimal objective value. The same is true when $\frac{Z(\mathscr{R}(\mathscr{C}))-Y(\mathscr{C})}{Y(\mathscr{C})}$ equals 0. Clearly, the delivered family can be optimal even when $\mathscr{R}(\mathscr{C}) \setminus \mathscr{C} \neq \emptyset$ and $\frac{Z(\mathscr{R}(\mathscr{C}))-Y(\mathscr{C})}{Y(\mathscr{C})} > 0$ as in this case the optimality will be proven in the next CGA iteration.

Finally observe that CGA will stop even if $\varepsilon = 0$ is assumed (and then return an optimal partition $\mathscr{R}(\mathscr{C})$ for SSRCDP) in a finite number of steps, because the number of all clusters is finite. This, however, can take an excessive computation time.

3.4 An efficient heuristic

In this section we describe a heuristic consisting in solving only one iteration of the CGA algorithm but using a modified version of APP($\mathscr{C}$). Consider a partition $\mathscr{R}$ defined by a binary vector $u = (u^t)_{t\in\mathcal{T}}$ feasible for APP($\mathscr{C}$), i.e., fulfilling (7b) and (7c).

Property 4. *Let $C = C(\tau, l)$ be a control cluster with $l \geq 2$ that has a non-empty intersection with exactly two (neighboring) clusters from $\mathscr{R}$ (i.e., $U^C = 1$). Let us also define the following quantity:*

$$Z(C|1) := \min\nolimits_{1\leq k\leq l-1}\left\{Z(C(\tau,k)) + Z(C(\tau\oplus k, l-k))\right\}. \tag{10}$$

Then the inequality

$$\textstyle\sum_{t\in C} y^t \geq Z(C|1) \tag{11}$$

is valid.

Proof. Suppose that $C \subseteq \mathcal{R}' \cup \mathcal{R}''$, where $\mathcal{R}'$ and $\mathcal{R}''$ are two neighboring (and disjoint) clusters from family $\mathscr{R}$ specified by u. Then $C = C(\tau,k) \cup C(\tau\oplus k, l-k)$ for some $1 \leq k \leq l-1$. Let $C' = C(\tau,k) \cap \mathcal{R}'$ and $C'' = C(\tau\oplus k, l-k) \cap \mathcal{R}''$. Since, by Remark 1, $Z(C') \leq \sum_{t\in C'} z^t(x^*(\mathcal{R}'))$ and $Z(C'') \leq \sum_{t\in C''} z^t(x^*(\mathcal{R}''))$. Thus, $\sum_{t\in C'} z^t(x^*(\mathcal{R}')) + \sum_{t\in C''} z^t(x^*(\mathcal{R}'')) \geq Z(C') + Z(C'') \geq Z(C|1)$, which shows that (11) is a valid inequality. Note that when in an optimal solution of APP($\mathscr{C}$), $C' = \mathcal{R}'$ and $C'' = \mathcal{R}''$ and inequality (11) becomes tight. □

Clearly, for $U^C = 1$, inequality (11) is tighter than the inequality implied by constraint (7g) (recall that $Y^C := \sum_{t\in C} y^t$) since in general $Z(C|1) > Z(C|\infty)$ (see Remark 1). Thus, substituting constraint (7g) in (7) with

$$Y^C \geq Z(C) + \big(Z(C|1) - Z(C)\big) \cdot U^C \qquad C \in \mathscr{C} \qquad (12)$$

will result in a modified version of APP($\mathscr{C}$) (referred to as MAPP($\mathscr{C}$)) with stronger linear relaxation than the original one.

Observe however, that for $U^C \geq 2$, inequality (12) is in general not valid. For example, for $U^C = 2$, the value of $Z(C) + \big(Z(C|1) - Z(C)\big) \cdot 2$ can be greater than the proper value given by the following formula (analogous to (10)):

$$\begin{aligned} Z(C|2) := \min\nolimits_{1 \leq k_1 < k_2 \leq l-1, k_2 - k_1 \geq L} \big\{ & Z(C(\tau, k_1)) + \\ & + Z(C(\tau \oplus k_1, k_2 - k_1)) + Z(C(\tau \oplus k_2, l - k_1 - k_2))\big\}. \end{aligned} \qquad (13)$$

It follows that MAPP($\mathscr{C}$) is correct only when the control family $\mathscr{C}$ is a subfamily of $\mathscr{C}(L+1)$ – the family of all clusters of length at most $L+1$ – since only then it is guaranteed that $U^C \leq 1$ for all $C \in \mathscr{C}$, and thus inequality in (12) is valid. Thus, the modified problem cannot be used in the CGA algorithm, as in general the family $\mathscr{R}(\mathscr{C})$ contains clusters with length larger than $L+1$ and such sets cannot be added to the control cluster family $\mathscr{C}$ when MAPP($\mathscr{C}$) is applied; therefore its use in CGA is limited to just one iteration. As we will see in Section 4, even this (non-iterative) solution gives very good results when applied to SSRCDP.

3.5 Improvements

The efficiency of the CGA algorithm described in Section 3.3 can be improved in two complementary ways.

First, the linear relaxation of formulation (7) can be strengthened (by improving, i.e., increasing, the lower bound delivered by its linear relaxation) in order to speed up the branch-and-bound algorithm (used to solve APP($\mathscr{C}$) in Step 1 of CGA)) and also to decrease the gap $\frac{Z(\mathscr{R}(\mathscr{C})) - Y(\mathscr{C})}{Y(\mathscr{C})}$ between the integer solution and the relaxed solution. The lower bound computed through the linear relaxation of formulation (7) can be increased by improving valid inequalities specified in constraint (7g). In fact, these valid inequalities are tight only for the case $U^C = 0$, i.e., when the control cluster C is contained in a cluster of the constructed family of routing clusters $\mathscr{R}$. (Recall that in this case the inequality in question takes the form $\sum_{t \in C} y^t \geq Z(C)$.) For $U^C \geq 1$ the inequalities implied by (7g) are weaker than the inequality in (7f), which, as already mentioned, implies that $\sum_{t \in C} y^t \geq Z(C|\infty)$, and this inequality is in general not tight.

A tight valid inequality generalizing (7g) can be obtained by constructing, for each $C \in \mathscr{C}$, a piece-wise linear function $G^C(U)$, $0 \leq U \leq M(C)$, where $M(C) := \lceil \frac{l(C)-1}{L} \rceil$ is an upper bound for U^C, and for integer values of the argument U, $G^C(U) = Z(C|U)$, where $Z(C|0) := Z(C)$, $Z(C|1)$ is defined by (10), $Z(C|2)$ by (13) and $Z(C|U)$, $U \geq 3$, are defined analogously. Then, the valid inequality in (7g) should be replaced with the tight valid inequality $Y^C \geq G^C(U)$. (Such an inequality is not linear but can be transformed, using additional binary variables and linear constraints, to a form appropriate for a MIP formulation.)

Second, on top of the family of clusters $\mathscr{R}(\mathscr{C})$ that is added to the control family $\mathscr{C}$ in Step 2 of CGA, we may seek to add extra control sets C' for which constraints (7g) are broken to the largest extent by the the current optimal values y^*.

4 NUMERICAL EXPERIMENT

Below we describe a numerical experiment illustrating the efficiency of the proposed APP($\mathscr{C}$)-based approach for a network linking 47 cities in an European Union country. The network consists of 47 nodes linked with 140 directed links (each of capacity 4 Gbps), and $47 \times 46 = 2162$ traffic demands corresponding to all ordered pairs of nodes. The demand volumes used in the calculations are derived from real traffic measurements (obtained from a network operator) taken every 15 minutes on a selected weekday (a Wednesday in 2018). Thus, the number of considered timepoints equals 96 ($T - 1 = 95$). We set the maximal number of clusters to $N = 8$ and the minimum cluster length to $L = 8$. This means that we accept at most 8 changes of the routing configuration during 24 hours and require that a routing configuration change can occur after the hold-off time of at least 2 hours.

In the experiment reported below, for solving the semi-stable routing cluster design problem (SSRCDP) we used formulation MAPP($\mathscr{C}$) in the way described in Section 3.4. The procedure was implemented using the platform: Lenovo Thinkpad, Intel i7-6500U, 8GB RAM, Windows 10 x64, ILOG CPLEX Studio 12.8, ILOG Concert library, C# language, CPLEX 12.8 solver, 2 threads.

For the control family $\mathscr{C}$ we used all the clusters of length L and $L+1$. There are $2T = 192$ of such clusters, and thus, in the preprocessing phase, for each of them we need to calculate the values $Z(C)$ and $Z(C|1)$ according to formulae (5a) and (11), respectively. For that, the routing problem RP($\mathcal{U}$) (5) is solved $8T = 768$ times, i.e., for all clusters of length between 2 and 9.

In RP($\mathcal{U}$) the delay function $F(z) := \max\{0.1z, z - 0.45, 10z - 8.5\}$ (with $K = 3$ linear pieces) was used, i.e., $b(1) = 0, b(2) = -0.45, b(3) = -8.5$ and $a(1) = 0.1, a(2) = 1, a(3) = 10$. Thus, $F(z)$ grows from 0 to 0.05 in the interval $[0, 0.5]$, from 0.05 to 0.5 in the interval $[0.5, 0.9]$, and from 0.5 to $+\infty$ in the interval $[0.9, +\infty]$.

The results of our experiment are presented in Table 2. For each task of the solution procedure, the corresponding row of the table first gives the determined lower bound (column LB) and the upper bound (column UB) for the optimal objective function value, and the current gap between the two (column GAP). Next, column T shows the total execution time of the task. Then, column N_{CLUSTERS} gives the number of clusters that we analyze in the task, i.e., clusters for which we solve the routing problem, and in brackets, if applicable, the number of clusters that are contained in the control set of the partitioning problem. Finally, column N_{PATHS} first shows (in brackets, with the plus sign) the total number of paths that we have generated while solving routing problems in the task, and (not in brackets) the final size of the set of paths $\mathcal{P}$ obtained in the routing problem.

In the row STATIC ROUTING, the case when only one routing cluster, i.e., $\mathcal{T}$, is applied. Then an optimized single routing scheme gives the optimal objective equal to $Z(\mathcal{T})$ given in the column UB, as this value is the upper bound for the true SSDRP optimal solution value. The row DYNAMIC ROUTING corresponds to the case when each timepoint is considered as a cluster, i.e., the routing scheme is optimized individually for each timepoint. Hence, the column LB in this row indicates $\sum_{t \in \mathcal{T}} Z(\{t\})$ which is clearly the cheapest solution value to SSRCDP (the case when the partition to the routing clusters is unconstrained). The value in column GAP, equal to $\frac{\text{UB-LB}}{\text{LB}} \times 100\%$ (UB taken for STATIC ROUTING and LB taken for DYNAMIC ROUTING), is indicated. The row PREPROCESSING contains information concerning preparation of the control cluster family $\mathscr{C}$ and initial routing paths (recall the RP($\mathcal{U}$) is solved through path generation). Next, the row PARTITIONING LR shows the results of solving the linear relaxation of the modified APP($\mathscr{C}$) formulation, i.e., of problem MAPP($\mathscr{C}$) described in Section 3.4. The so obtained value of LB happens to be the same as for DYNAMIC ROUTING, although in general it could be larger. Further, the solution of the MIP formulation

Table 2: Performance of the solution procedure

TASK	LB	UB	GAP	T	N$_{\text{CLUSTERS}}$	N$_{\text{PATHS}}$
STATIC ROUTING	-	563.65	-	5m7s	1	(+4461) 6623
DYNAMIC ROUTING	545.47	-	3.33%	1m23s	96	(+89) 6712
PREPROCESSING	-	-	-	1h1m16s	768	(+1298) 8010
PARTITIONING LR	545.47	-	3.33%	1s	(192)	-
PARTITIONING MIP	550.50	-	2.43%	2s	(192)	-
ROUTING	-	551.86	0.25%	1m16s	8	(+1) 8011

MAPP($\mathscr{C}$) is described in the row PARTITIONING MIP. The LB value delivered by this solution is increased with respect to the preceding row and hence the GAP value is decreased. Finally, the row ROUTING shows the results for the partitioning $\mathscr{R}(\mathscr{C})$ obtained with the MIP formulation MAPP($\mathscr{C}$) with the routing scheme optimized for each of the resulting routing clusters $\mathcal{R}$. In particular, UB gives the value of $Z(\mathscr{R}(\mathscr{C}))$. Observe that the gap between this feasible SSRCDP solution and the best lower bound obtained with PARTITIONING MIP is very small and equals 0.25%. In the final solution, the optimal routing cluster family $\mathscr{R}(\mathscr{C})$ is composed of five 8-element, one 13-element, one 15-element, and one 28-element clusters.

The results indicate that already the simplified version of the proposed method, without any special tuning, is capable of finding a suboptimal solution of SSRDCP in a reasonable time within the optimality gap as small as 0.25%.

5 CONCLUSIONS

In this paper we propose a scalable solution to the problem of designing clusters of the semi-stable routing in multicommodity flow networks. Although the problem can be approached directly using a compact mixed-integer formulation it cannot be just solved with a solver, even for small-size networks, due to an excessive number of binary variables and poor linear relaxation. Thus we were considering a number of exact and hybrid approaches (as in [13]) that aimed at separating the design of a partition of the time horizon into clusters from the design of traffic routing for those clusters.

Although there are just $O(T^2)$ clusters with length between 1 and T (where T is typically between 96 and 288 as the traffic measurement period is either 5 or 15 minutes), our numerical trials show that in practice we cannot analyze all those clusters. Using a link-path formulation combined with path generation and a warm start for the master problem, it took around k seconds to solve the routing problem for a cluster of length k and a 50-node network. And this time might grow considerably as we aim at networks whose number of nodes approaches 500.

Therefore, leveraging the valid inequalities of an approximate time-horizon partitioning problem, we developed an efficient heuristic algorithm based on cluster preprocessing. Our algorithm is capable of providing the upper and the lower objective function value bounds with very low optimality gaps, well below 0.5%, as shown in the presented numerical study (and some other studies not reported here for the lack of space). It also offers the trade-off between the quality of the solution, and the number of clusters in the control set that influences the preprocessing time, and the size and the solution time of the partitioning problem.

In addition, we have proposed two possible ways for improving the efficiency of the approach that lead to interesting future research. First, we can use a stronger formulation of APP($\mathscr{C}$) equipped with improvements described in Section 3.5. Second, we can either implement a full version of the cluster generation algorithm presented in Section 3.3, or, even better, to incorporate cluster generation into a branch-and-bound procedure of solving the partitioning problem, by analyzing relaxed or incumbent solutions and generating appropriate user cuts. We will also aim at testing the resulting optimization procedure on examples with lower correlation among the traffic matrices, which might feature a more substantial gap between the static and dynamic routing solutions than the 3.33% observed in the current example (which our algorithm nonetheless managed to decrease tenfold).

ACKNOWLEDGMENT

The work of the Polish authors was supported by the National Science Center, Poland, grant no. 2015/17/B/ST7/03910 “Logical tunnel capacity control – a traffic routing and protection strategy for communication networks with variable link capacity”.

REFERENCES

[1] Yossi Azar, Edith Cohen, Amos Fiat, et al. 2003. Optimal oblivious routing in polynomial time. In *ACM Symp. on Theory of Computing*. 383–388.

[2] Walid Ben-Ameur and Mateusz Żotkiewicz. 2011. Robust routing and optimal partitioning of a traffic demand polytope. *Intl. Trans. in Operational Research* 18, 3 (2011), 307–333.

[3] Walid Ben-Ameur and Mateusz Żotkiewicz. 2013. Multipolar routing: where dynamic and static routing meet. *Electronic Notes in Discrete Mathematics* 41 (2013), 61–68.

[4] Theophilus Benson, Ashok Anand, Aditya Akella, and Ming Zhang. 2011. MicroTE: Fine grained traffic engineering for data centers. In *Proc. ACM CoNext*. 8.

[5] Pedro Casas, Lionel Fillatre, and Sandrine Vaton. 2008. Multi Hour Robust Routing and Fast Load Change Detection for Traffic Engineering. In *Proc. IEEE ICC*. 5777–5782.

[6] Bernard Fortz and Mikkel Thorup. 2002. Optimizing OSPF/IS-IS weights in a changing world. *IEEE JSAC* 20, 4 (2002), 756–767.

[7] Chi-Yao Hong, Srikanth Kandula, Ratul Mahajan, et al. 2013. Achieving high utilization with software-driven WAN. In *ACM SIGCOMM CCR*, Vol. 43. 15–26.

[8] Sushant Jain, Alok Kumar, Subhasree Mandal, et al. 2013. B4: Experience with a globally-deployed software defined WAN. *ACM SIGCOMM CCR* 43, 4 (2013), 3–14.

[9] Murali Kodialam, TV Lakshman, and Sudipta Sengupta. 2004. Efficient and robust routing of highly variable traffic. In *Proc. HotNets*.

[10] Michał Pióro and Deep Medhi. 2004. *Routing, Flow, and Capacity Design in Communication and Computer Networks*. Morgan-Kaufmann.

[11] Michael Poss and Christian Raack. 2013. Affine recourse for the robust network design problem: Between static and dynamic routing. *Networks* 61, 2 (2013), 180–198.

[12] Matthew Roughan, Mikkel Thorup, and Yin Zhang. 2003. Traffic engineering with estimated traffic matrices. In *Proc. ACM IMC*.

[13] Davide Sanvito, Ilario Filippini, Antonio Capone, Stefano Paris, and Jeremie Leguay. 2018. Adaptive Robust Traffic Engineering in Software Defined Networks. In *Proc. IFIP Networking*.

[14] Marco Silva, Michael Poss, and Nelson Maculan. 2018. Solving the bifurcated and nonbifurcated robust network loading problem with k-adaptive routing. *Networks* 72, 1 (2018), 151–170.

[15] Vahid Tabatabaee, Abhishek Kashyap, Bobby Bhattacharjee, Richard J La, and Mark A Shayman. 2007. Robust routing with unknown traffic matrices. In *Proc. IEEE INFOCOM*. 2436–2440.

[16] Hao Wang, Haiyong Xie, Lili Qiu, Yang Richard Yang, Yin Zhang, and Albert Greenberg. 2006. COPE: traffic engineering in dynamic networks. In *ACM SIGCOMM CCR*, Vol. 36. 99–110.

[17] Yin Zhang and Zihui Ge. 2005. Finding critical traffic matrices. In *Proc. IEEE DSN*.

The Workforce Routing and Scheduling Problem: solving real-world Instances

Gabriel Volte
LIRMM, University of Montpellier, CNRS
Montpellier, France
gabriel.volte@lirmm.fr

Chloé Desdouits
DecisionBrain
Montpellier, France
chloe.desdouits@decisionbrain.com

Rodolphe Giroudeau
LIRMM, University of Montpellier, CNRS
Montpellier, France
rodolphe.giroudeau@lirmm.fr

ABSTRACT

We propose an efficient method to solve a workforce routing and scheduling problem with working constraints, and a bounded execution time limit. This problem combines two fundamental problems in operations research: routing and scheduling. In such a context, we develop a column generation algorithm, as a set partitioning problem with side constraints, within a branch-and-price framework. The pricing sub-problem is an elementary shortest path with resource constraints modeled with constraint programming. In our branch-and-price framework, we first solve our problem using branch-and-price and a branch-and-bound strategy is proposed on the last restricted master problem, in order to obtain a feasible solution when the time limit is almost reached. However, we show that the developed method leads to better solutions than using constraint programming or large neighborhood search methods. We show the relevance of our method with various-size real instances.

INTRODUCTION

We consider in this paper a hybrid problem in which it is necessary to associate the vehicle optimization problem with an assignment problem for employees to satisfy some specific technical constraints. The study of this problem is motivated by taking into account new business constraints for employees with specific skills. These problems are more and more present in the everyday life of maintenance companies. The main difficulty is to consider the various parameters to respond to real situations.

Workforce Scheduling and Routing Problem (hereafter *WSRP*) represents problems that mobilize workforce to perform tasks for customers. Given a set of employees and a set of tasks to be scheduled, *WSRP* consists in assigning tasks to employees in order to fulfill some constraints while minimizing operational costs.

WSRP combines the complexity of scheduling problems [2, 18]:

- Multi-skill Project Scheduling Problem, MPSP [6, 14, 21] (Technician and Task Scheduling Problem).
- Sequencing and Scheduling Problem, SSP [19],
- Project Scheduling with Resources Constrained Scheduling Problem,

and problems of vehicle routing [20, 25]:

- Vehicle Routing Problem with Time Windows [23],
- Vehicle Routing Problem with Time Windows and Dependencies,

Figure 1 represents the successive generalizations of basic scheduling and routing problems, such as the *TSP*, that lead to the *WSRP* class of problems.

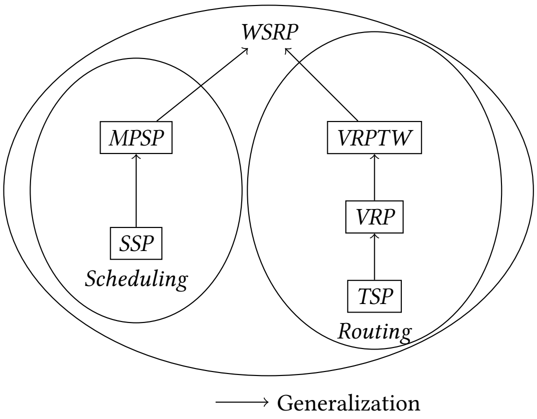


Figure 1: Description of hierarchical complexity class for *WSRP* .

This paper is organized as follows: the section **RELATED WORK** gives an overview of the previous works found in the literature on the *WSRP*, the section **MODELLING** formally describes our problem and a first compact model using integer linear programming (henceforth ILP) is given. The column generation decomposition and the branch-and-price scheme implemented is described in section **THE BRANCH-AND-PRICE FRAMEWORK**. The results and instances are presented in section **TESTS**. The last section concludes the paper and presents some future work.

RELATED WORK

In the next section, we formally define the Workforce Scheduling and Routing Problem class, based on the survey [3]. This survey first presents the common characteristics of technicians and tasks, summarized in Table 1, then reviews known methods to solve problems considered as *WSRP*. The main method used to tackle these problems is a hybrid approach combining exact methods, integer linear programming or constraint programming, and heuristics/meta-heuristics methods, large neighborhood search or tabu search. The branch-and-price approach is also used since this approach is known to be efficient on routing problems and scheduling problems. This survey also gives a detailed computational study outlining the computational difficulties to solve these problems. This study has been carried out on different data sets with different integer linear programming formulations.

We describe some characteristics presented in Table 1. The processing time of the tasks is not negligible compared to the travel time and may depend on the employee. Tasks have required skills to filter employees who can perform them. A task can be processed by one or more employees, in which case all employees must be present before the starting time of the task. Castillo et

ISBN 978-3-89318-079-0 on OpenProceedings, ISSN 2510-7437

DOI: 10.5441/002/inoc.2019.12

al. [4] and Rasmussen et al. [22] define temporal dependencies among tasks. Thus, some tasks admit a priority over others.Tasks can have priority meaning that a task should be performed before others.

Some tasks can be outsourced. In addition, the schedule of the employees can vary: it can be daily, weekly, etc. In general, *WSRP* instances are too big to be exactly solved. They are usually divided into smaller geographical areas to prevent an employee from working far from home but also to reduce the size of instances making it easier to solve.

Employee	Task
Means of transport	Processing time
Starting Position	Position
Ending position	Temporal dependence
Working Hours	Opening hours
Team	Required Skills
Skills	Priority
	Outsourcing

Table 1: Characteristics of employees and tasks.

For example, we may consider a set of employees who have to execute a set of tasks. Employees can travel by car, bicycle or public transport to perform the tasks. Employees are allowed to start and end their day from home. In the literature, there are numerous works of surveys aimed at characterizing and classifying the various problems belonging to the *WSRP* class. Based on an extension of the classic notation scheme $\alpha \mid \beta \mid \gamma$ proposed by [13], Desrochers [7] develop a classification of *WSRP*.

An extensive overview of time constraint routing and scheduling problems during the last decades is given in Desrosiers et al. [8]. They detail ILP models and algorithms (column generation and dynamic programming) for each variation of problems (TSPTW, SDVRPTW, MDVRPTW, etc), focus their work on optimization methods for practical size instances. Although, they also present heuristic methods to solve complex problems or large-scale instances when optimal solutions are too difficult to obtain. The survey [24] outlines the research on different routing problems with time windows (M-TSPTW, SPPTW, etc) and give hints for future works on these problems.

To solve problems belonging to *WSRP*, we observe in literature many methods such as exact methods (constraint programming or integer linear programming), meta-heuristics (simulated annealing, tabu search, genetics, ...) or hybrid methods. Regarding exact methods, one can find ILP models and column generation using Dantzig-Wolfe decomposition. The master problem corresponds to a set partitioning problem [1] and the sub-problem to an Elementary Shortest Path Problem with Time Windows [12, 15] which is known to be $\mathcal{NP}$-hard [10].

MODELLING

The goal of the project is to assign maintenance tasks to technicians in order to build daily schedules while optimizing some criterion such as quality of service, travel time, productivity and efficiency. The time limit is bounded to at most one hour for the biggest instances. The number of tasks is too large to schedule all of them in one day, thus tasks can be postponed. Thus the set of tasks is updated every day according to previous schedules.

The problem can be stated as follows: let us define $\mathcal{P}$ the set of technicians and $\mathcal{T}$ the set of tasks. For each technician, we add two artificial tasks: one for the starting point (0^p) and the other for the ending point (n^p). Therefore, we define the set $\mathcal{T}^p = \mathcal{T} \cup \{0^p, n^p\}$ for each technician p.

Let p_j be the processing time of task j, and let d_j be the due date of task j, ω_j is the weight (or revenue) of task j.

Let q be the number of skills and l the number of level of skills. Consider $\alpha^p = (\alpha_1^p, \alpha_2^p, ..., \alpha_q^p)$ be the skill vector of technician p and $\beta^j = (\beta_1^j, \beta_2^j, ..., \beta_q^j)$ the skill vector of task j. For each $i \in \{0, \ldots, q\}$, α_i^p and β_i^j indicate the level (value in $\{0, \ldots, l\}$) of the ith skill in the vector.

Each task possesses a location and each technician have a starting location and ending location. Let M be the distance matrix where $m_{i,j}$ represents the distance between locations i and j.

Let $\overline{\mathcal{K}}^p$ (resp. $\mathcal{K}^p$) be the set of unavailable (resp. available) periods of technician p. The previous notation is extended to task j with $\overline{\mathcal{K}}^j$ and $\mathcal{K}_j$. $\mathcal{K}_j^p$ denotes the set of time windows where technician p and task j are both available, $\mathcal{K}_j^p = \mathcal{K}_j \cap \mathcal{K}^p$. We define $[a_k, b_k] \in \mathcal{K}_j^p$ the k^{th} time window of the set. A task cannot overlap unavailable period (no task should start or end during an unavailable period).

The beginning (resp. ending) of the workday of a technician is given by the starting time (resp. end time) of his working hours. Moreover, the technician cannot travel before his starting hours or after his ending hours. Lastly, if a technician arrives early to a customer waiting is allowed.

Our problem can be formulated as integer linear program given below. The routing variables x_{ijk}^p take value 1 if the technician $p \in \mathcal{P}$ travels from task $i \in \mathcal{T}^p$ to task $j \in \mathcal{T}^p$ in the time window $k \in \mathcal{K}_i^p$, 0 otherwise; the scheduling variables t_i^p correspond to the time the technician $p \in \mathcal{P}$ starts the task $i \in \mathcal{T}^p$; the covering variables y_i take value 1 if the task $i \in \mathcal{T}$ is unscheduled/uncovered and 0 if the task i is performed by a technician; the tardiness variables D_i correspond to the lateness of the task $i \in \mathcal{T}$. First, we introduce an ILP representing the backbone of our problem, then we will add the specific constraints (*same technician constraints* and *appointment constraints*).

$(\pi_1, \pi_2, \pi_3, \pi_4)$ are the weights of the different criteria in the objective function (Equation (1)). The first criterion corresponds to the number of unscheduled tasks, the second minimizes the technician's travel distances, the third computes the sum of the tasks tardiness and finally the fourth maximizes the skill gap between technicians and tasks. $\mathcal{W}$ is the total of the weight of all tasks. The first criterion maximizes the weighted sum of tasks scheduled but we choose to minimize the weighted sum of unscheduled tasks, the weight of all tasks minus the weight of all unscheduled tasks gives the weight of all scheduled tasks.

We denote by *Prec*, *Same* and *App*, the set of pairs $(i, j) \in \mathcal{T}\times\mathcal{T}$ for which a *precedence constraint*, a *same technician constraint*, and *appointment constraint* exists, respectively. *Precedence constraints* are defined below. *Same technician constraint* corresponds to a pair $(i, j) \in$ *Same*, if technician p executes task i (resp j) thus he is the only one who can perform task j (resp i).

Appointment constraints enforce a task to be performed by a technician at a fixed time. These constraints appear when the customers require a specific technician or a specific time to perform a job. There are three kinds of appointment constraints:

- When task j is assigned to technician p (even if he does not have the required skills to perform it): when p should perform j?

- When task j is assigned to time t: which technician should perform it?
- When task j is assigned to technician p and to time t: is j scheduled for p at time t?

Model M1: Compact formulation

$$\textbf{Maximize}\ \ \pi_1(\mathcal{W} - \sum_{t \in \mathcal{T}} \omega_i y_i) - \pi_2 \sum_{p \in \mathcal{P}} \sum_{i,j \in \mathcal{T}^p} \sum_{k \in \mathcal{K}_i^p} x_{ijk}^p m_{i,j}$$

$$-\pi_3 \sum_{j \in \mathcal{T}} \omega_j^{'} D_j + \pi_4 \sum_{p \in \mathcal{P}} \sum_{i,j \in \mathcal{T}^p} \sum_{k \in \mathcal{K}_i^p} \sum_{s=1}^{q} x_{ijk}^p (\alpha^p(s) - \beta^i(s)) \tag{1}$$

$$\textbf{s.t.}\ \ \sum_{p \in \mathcal{P}} \sum_{j \in \mathcal{T}^p} \sum_{k \in \mathcal{K}_i^p} x_{ijk}^p + y_i = 1\ \forall i \in \mathcal{T} \tag{2}$$

$$\sum_{j \in \mathcal{T}^p} \sum_{k \in \mathcal{K}_{0^p}^p} x_{0^p jk}^p = 1\ \forall p \in \mathcal{P} \tag{3}$$

$$\sum_{j \in \mathcal{T}^p} \sum_{k \in \mathcal{K}_j^p} x_{jn^p k}^p = 1\ \forall p \in \mathcal{P} \tag{4}$$

$$\sum_{i \in \mathcal{T}^p} \sum_{k \in \mathcal{K}_i^p} x_{ihk}^p - \sum_{j \in \mathcal{T}^p} \sum_{k \in \mathcal{K}_h^p} x_{hjk}^p = 0\ \forall p \in \mathcal{P},\ \forall h \in \mathcal{T} \tag{5}$$

$$\sum_{h \in \mathcal{T}^p} \sum_{k \in \mathcal{K}_j^p} x_{jhk}^p (\alpha^p(i) - \beta^j(i)) \geq 0\ \forall j \in \mathcal{T},\ \forall p \in \mathcal{P},\ \forall i \in [1..q] \tag{6}$$

$$B_j y_i + \sum_{p \in \mathcal{P}} t_i^p + p_i \leq \sum_{p \in \mathcal{P}} t_j^p + B_i y_j \quad \forall (i,j) \in Prec \tag{7}$$

$$\sum_{j \in \mathcal{T}} \sum_{k \in \mathcal{K}_i^p} x_{ijk}^p a_k \leq t_i^p \leq \sum_{j \in \mathcal{T}} \sum_{k \in \mathcal{K}_i^p} x_{ijk}^p b_k \quad \forall i \in \mathcal{T},\ \forall p \in \mathcal{P} \tag{8}$$

$$\sum_{j \in \mathcal{T}} \sum_{k \in \mathcal{K}_i^p} x_{ijk}^p a_k \leq t_i^p + (\sum_{j \in \mathcal{T}} \sum_{k \in \mathcal{K}_i^p} x_{ijk}^p).p_i \leq \sum_{j \in \mathcal{T}} \sum_{k \in \mathcal{K}_i^p} x_{ijk}^p b_k \quad \forall i \in \mathcal{T},\ \forall p \in \mathcal{P} \tag{9}$$

$$t_i^p + x_{ijk}^p (m_{i,j} + p_i) \leq t_j^p + (1 - x_{ijk}^p).B_k \quad \forall p \in \mathcal{P},\ \forall i,j \in \mathcal{T}^p,\ \forall k \in \mathcal{K}_i^p \tag{10}$$

$$D_j \geq \sum_{p \in \mathcal{P}} t_j^p + p_j - d_j \quad \forall j \in \mathcal{T} \tag{11}$$

$$x_{ijk}^p \in \{0,1\}, \quad \forall p \in \mathcal{P},\ \forall i,j \in \mathcal{T}^p,\ \forall k \in \mathcal{K}_i^p \tag{12}$$

$$t_i^p \in \mathbb{N}, \quad \forall p \in \mathcal{P},\ \forall i \in \mathcal{T}^p \tag{13}$$

$$y_i \in \{0.1\}, \quad \forall i \in \mathcal{T} \tag{14}$$

$$D_j \in \mathbb{N}, \quad \forall j \in \mathcal{T} \tag{15}$$

The Model M1 is inspired by the MIP model given by a home nursing problem [22] and VRPTW [9]. Our problem differs on a few constraints: the skills/qualifications of employees, the working hours of employees and precedence constraints (they have temporal dependencies constraints). In addition, one additional dimension is needed on the routing decision variables (x_{ijk}^p) because it is necessary to know on which time window the task is scheduled. The relevance of this compact formulation is discussed in Section **TESTS** and presented in Table 3.

Constraints (2) ensure that each task is scheduled at most once. If a task i is not scheduled (i.e. $\sum_{p \in \mathcal{P}} \sum_{j \in \mathcal{T}^p} \sum_{k \in \mathcal{K}_j^p} x_{ijk}^p = 0$) then to satisfy the constraint, task i must be covered by $y_i = 1$ and the task is postponed to a another day. The flow constraints for each technician depicted in (3), (4) and (5) control that technicians must start (resp. finish) their shift at their start (resp. end) location and that the flow conservation is respected (i.e. if a technician arrives at a customer he musts leave it). The skill constraints (6) restrict the set of tasks allowed to be performed by a given technician. A task can only be performed by a technician who has the required skills. In the current implementation of these constraints, we force variables x_{ijk}^p to 0 when $\alpha^p(s) - \beta^i(s) > 0$, $s \in [0,..,q]$. Constraints (7) give the precedence constraints among the tasks: let $(i,j) \in Prec$, task j can be performed only if task i is executed before. The temporal constraints (8) and (9) and (10) verify that the availability periods of tasks and technicians are respected in the schedule (no overlap with unavailability or travel periods). If a task j is not executed by technician p, constraint (8) forces t_j^p to 0.

Constraints (16)-(19) correspond to problem-specific constraints.

Constraints (16) model the same technician constraints: they ensure that if a technician performs one of the two tasks, then the second is either performed by the same technician or the task is not scheduled. Constraints (17), (18) and (19) represent pre-assignment constraints describe above. Constraints (17) ensure that the right technician performs the task or the task is not scheduled. Constraints (18) ensure that the task is scheduled at the right time or not at all. Constraints (19) ensure that the task is assigned to the right technician at the right time or that otherwise task is not performed.

$$\sum_{h \in \mathcal{T}^p} \sum_{k \in \mathcal{K}_j^p} x_{ihk}^p + y_i = \sum_{h \in \mathcal{T}^p} \sum_{k \in \mathcal{K}_j^p} x_{jhk}^p + y_j \quad \forall (i,j) \in Same,\ \forall p \in \mathcal{P} \tag{16}$$

$$\sum_{j \in \mathcal{T}} \sum_{k \in \mathcal{K}_j^p} x_{ijk}^p - (1 - y_i) = 0 \quad \forall (i,p) \in App \tag{17}$$

$$\sum_{p \in \mathcal{P}} t_i^p - (1 - y_i).t_i = 0 \quad \forall (i, t_i) \in App \tag{18}$$

$$t_i^p - (1 - y_i).t_i = 0 \quad \forall (i,p,t_i) \in App \tag{19}$$

THE BRANCH-AND-PRICE FRAMEWORK

In this section, we will introduce a branch-and-price framework. First, we use a Dantzig-Wolfe decomposition on the compact formulation in order to model it as a set partitioning problem with side constraints. In a branch-and-price framework the problem is split into a master problem (hereafter MP) and a pricing sub-problem (henceforth PSP). The PSP generates new feasible schedules/routes for each technician. Given the set of all feasible technician schedules, the MP assigns a schedule to each technician such that a maximum of tasks is processed (c.f. the first criterion of the objective function). Since the set of feasible technician schedules can be very large, we restrict the MP to a subset of schedules to obtain a reasonable size problem (called Restricted Master Problem denoted by RMP). A feasible route for a technician begins at his starting location and ends at his ending location, and respects all constraints mentioned in Model M1.

Master Problem

Consider $\mathcal{S}^p$ the set of all feasible schedules for technician p (this set will be generated successively by the PSP). Let $a_{is}^p = 1$ if task i is in Schedule s of technician p and 0 otherwise; let t_{is}^p be the starting time of Task i in schedule s of technician p. The constant c_s^p represents the cost of the schedule s for technician p. In the compact formulation (cf. Model M1), the aim is to minimize delays, distances and maximize the skill gap between tasks and technicians. This cost is a variation of the objective function of the compact formulation (Equation (1)). The first criterion of the compact formulation objective function is separated from the others in the RMP objective function to enhance the linear relaxation of the RMP.

$$c_s^p = -\pi_2 \sum_{i,j \in \mathcal{T}} \sum_{k \in \mathcal{K}_j^p} x_{ijk}^p m_{i,j} - \pi_3 \sum_{j \in \mathcal{T}} \omega_j^{'} D_j$$

$$+ \pi_4 \sum_{i,j \in \mathcal{T}} \sum_{k \in \mathcal{K}_j^p} \sum_{s=1}^{q} x_{ijk}^p (\alpha^p(s) - \beta^i(s))$$

We introduce binary variables for the RMP: the scheduling variables λ_s^p take value 1 if schedule s is chosen for technician p and 0 otherwise; the covering variables γ_i take value 1 if task i is uncovered/unscheduled and 0 otherwise.

Model M2: Restricted Master Problem

$$\textbf{Max} \sum_{p \in \mathcal{P}} \sum_{s \in \mathcal{S}^p} \lambda_s^p c_s^p + \pi_1 \sum_{i \in \mathcal{T}} (1 - \gamma_i) w_i \tag{20}$$

$$\textbf{s.t.} \sum_{p \in \mathcal{P}} \sum_{s \in \mathcal{S}^p} a_{is}^p \lambda_s^p + \gamma_i = 1 \ \ \forall i \in \mathcal{T} \tag{21}$$

$$\sum_{s \in \mathcal{S}^p} \lambda_s^p \leq 1 \ \ \forall p \in \mathcal{P} \tag{22}$$

$$\gamma_i + \sum_{s \in \mathcal{S}^p} a_{is}^p \lambda_s^p = \sum_{s \in \mathcal{S}^p} a_{js}^p \lambda_s^p + \gamma_j \quad \forall (i,j) \in Same, \ \forall p \in \mathcal{P} \tag{23}$$

$$B_j \gamma_i + \sum_{p \in \mathcal{P}} \sum_{s \in \mathcal{S}^p} t_{is}^p \lambda_s^p + p_i \leq \sum_{p \in \mathcal{P}} \sum_{s \in \mathcal{S}^p} t_{js}^p \lambda_s^p + B_i \gamma_j \quad \forall (i,j) \in Prec \tag{24}$$

$$\lambda_s^p \in [0,1] \ \ \forall p \in \mathcal{P}, \ \forall s \in \mathcal{S}^p \tag{25}$$

$$\gamma_i \in [0,1] \ \ \forall i \in \mathcal{T} \tag{26}$$

Constraints (21) express the fact that each task must be executed or covered. Constraints (22) ensure that only one schedule is associated with a technician. The *same technician constraints* are modeled by constraints (23). The *same technician constraints* are only in the RMP because in the PSP these constraints are always checked (a PSP is solved for each technician). Constraints (24) model the *precedence constraints.* As *precedence constraints* among tasks are independent of the set of technicians, these constraints must be present in the RMP and the PSP. Constraints (25) (resp. (26)) indicate the domain of λ_s^p variables (resp. γ_i).

For any primal solution of the RMP, we obtain a dual solution $[\mathbf{u}, \mathbf{z}, \mathbf{l}, \mathbf{w}]$, where $\mathbf{u} = (u_i)_{i \in \mathcal{T}}$; $\mathbf{z} = (z_p)_{p \in \mathcal{P}}$; $\mathbf{l} = ((l_i, l_j))_{(i,j) \in Prec}$; $\mathbf{w} = ((w_{ip}, w_{jp}))_{(i,j) \in Same,\ p \in \mathcal{P}}$ are the dual variables of constraints (21), (22), (24), (23) respectively. These dual variables are used in the PSP (cf. Equation (36)) to generate new improving routes for each technician.

Pricing subproblem

In our case, the sub-problem generates feasible schedules/routes (that respect the constraints) for each technician, thus these routes are added to the RMP. The sub-problem aims to find feasible routes for a technician which improve the solution obtained in the RMP. We cannot consider technicians as a fleet of vehicles (they have almost no similar characteristics), thus we must solve a sub-problem for each technician. The PSP is solved using constraint programming with the ILOG IBM Scheduler constraints and variables (for more information on those constraints and variables please refer to [16, 17]). The Pricing Sub-Problem (hereafter PSP) is the elementary shortest path problem with time windows (ESPPTW). It focuses on finding an improved schedule for a particular technician. Recall that ESPPTW is $\mathcal{NP}$-hard in a strong sense [10] (there is no hope to develop dynamic programming).

Since our problem is a maximization problem if the PSP objective function $Z_{PSP} < 0$ (cf. Equation (35)) then the corresponding route is not improving the current solution. Adding it in the solution of the RMP would decrease the value of the objective function. So we add in the RMP all tours with a reduced cost (Z_{PSP}) strictly positive to potentially increase the value of the objective function. Since the PSP is hard to solve, the optimization is terminated as soon as a tour with a strictly positive reduced cost is found. Thanks to the constraints propagators, constraint programming is effective to find a good feasible solution in a short time.

For any technician p, we construct the following constraint programming model. We introduce the interval variables X_i^I, $\forall i \in \mathcal{T}^p$ to model tasks scheduling time. The domain of these variables is either $\{\perp\}$ (task is not processed) or the scheduling horizon (the scheduling time of the task). Let X_p^S refers to the sequence variable of technician p, the domain of this variable is a permutation of tasks interval variables: $D(X_p^S) = \text{perm}(\{X_{p-i}^I \mid \forall i \in \mathcal{T}\} \cup \{X_{p0}^I, X_{pn}^I\})$.

Model M3: Constraint programming

$$NoOverLap(X_p^S, M) \tag{27}$$

$$first(X_p^S, X_{p0}^I) \tag{28}$$

$$last(X_p^S, X_{pn}^I) \tag{29}$$

$$\begin{cases} pOf(X_j^I) \leq pOf(X_i^I) \\ EndBeforeStart(X_i^I, X_j^I) \end{cases} \forall (i,j) \in Prec \tag{30}$$

$$\begin{cases} ForbidExtent(X_i^I, \mathcal{K}_p^i) \\ ForbidStart(X_{p0}^I, \mathcal{K}^p) \quad \forall i \in \mathcal{T}, \\ ForbidEnd(X_{pn}^I, \mathcal{K}^p) \end{cases} \tag{31}$$

$$X_T = \sum_{i \in \mathcal{T}} max(0, d_i - EndOf(X_i^I)) \tag{32}$$

$$X_D = \sum_{(i,j) \in X_p^S} m_{i,j} \tag{33}$$

$$X_{SG} = \sum_{i \in \mathcal{T}} pOf(X_i^I) \times \sum_{s \in [1..q]} (\alpha^p(s) - \beta^i(s)) \tag{34}$$

$$\text{Max } Z_{PSP} = X_{SG} - X_T - X_D - f(\mathbf{u}, \mathbf{z}, \mathbf{l}, \mathbf{w}) \tag{35}$$

Equation (27) ensures that the tasks performed by p are not overlapping and respects the travel time matrix M. Equations (28) and (29) enforce the route to begin (resp. end) at the starting (resp. ending) location. The constraint pOf (meaning $presenceOf$, is used to know if a task is executed) and the constraint $EndBeforeStart$ are both used to assure that precedence constraints (30) are satisfied. Equations (31) prevent tasks to be performed outside the technician and task time windows. The constraint $ForbidExtend$ ensures that tasks are not overlapping an unavailability period (given by $\mathcal{K}_p^i or \mathcal{K}^p$). The constraint $ForbidStart$ (resp. $ForbidEnd$) ensures that tasks begin before (resp. end after) an unavailability period. We restrict the domain of the variables to satisfy appointment constraints. In the objective function (cf. Equation (35)), the variable X_{SG} computes the skill gap between technicians and tasks (cf. Equation (34)), the variable X_T computes the tasks tardiness (cf. Equation (32)) and the variable X_D computes the travel time/distance (cf. Equation (33)). The function $f(\mathbf{u}, \mathbf{z}, \mathbf{l}, \mathbf{w})$ is dedicated to the cost associated with the dual variables $[\mathbf{u}, \mathbf{z}, \mathbf{l}, \mathbf{w}]$ defined above.

$$f(\mathbf{u}, \mathbf{z}, \mathbf{l}, \mathbf{w}) = \sum_{i \in \mathcal{T}} pOf(X_i^I) \times u_i + z_p + \sum_{(i,j) \in Prec} (l_j \beta_i - l_i \beta_j)$$
$$- \sum_{(i,j) \in Same} (w_{jp} \times pOf(X_j^I) - w_{ip} \times pOf(X_i^I)) \tag{36}$$

Branching strategies

We based our branching strategy on the ones presented in [11]. This paper presents two rules for branching. The first one, «standard strategy» consists in branching on decision variables λ_s^p, this branching is not effective because it leads to an unbalanced branching tree. The second one, is the «natural strategy» consists in branching on flow variables x_{ij} (cf. Model M1) and decision variables y_i, we opt for this strategy because it leads to a more balanced tree and an easier PSP.

Branch-and-price is usually used to obtain optimal solution but with a lack of resources and because of the large-scale instances this method is neglected. Because of the time limit and the large-scale highly constrained instances, finding an optimal solution can be difficult. Our algorithm is based on the one used in [5]. The authors propose a column generation to obtain an optimal non-integer solution. Therefore, a branch-and-bound algorithm is applied to obtain an integer solution.

We enhance this method adding the branch-and-price framework to generate more heterogeneous routes. We solve the problem with the following branch-and-price scheme (cf. Figure 2). We first solve the problem using a standard branch-and-price framework, at each node of the branching tree RMP is solved using column generation. If an integer solution is found, the bound (the best feasible solution found) is updated, else we add a branching node. At the end of the time limit, if we reach it, we use branch-and-bound method on the last RMP to obtain an integer solution. If this solution is better than the best one found in the branch-and-price algorithm we keep it.

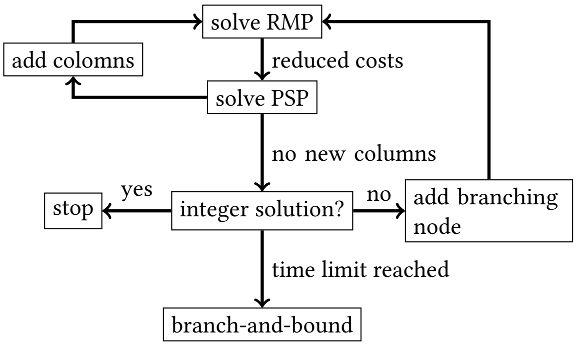


Figure 2: Illustration of our branch-and-price framework.

TESTS

We have access to many instances of two Decisionbrain customers. Table 2 gives statistics for each instance. The name #30#256 (first column) means that the instance has 30 technicians and 256 tasks, the column $|Prec|$ gives the number of precedence constraints, $|Same|$ shows the number of same technician constraints, $|App|$ represents the number of precedence constraints, q denotes the number of skills (length of the skill vector), *loc* indicate the number of task and technician locations, $\mathcal{K}^{\mathcal{P}}$ (resp. $\mathcal{K}_{\mathcal{T}}$) shows the mean of technician time windows (resp. task time windows).

instance	$\|Prec\|$	$\|Same\|$	$\|App\|$	q	loc	$\mathcal{K}^{\mathcal{P}}$	$\mathcal{K}_{\mathcal{T}}$
#30#256	0	0	0	154	162	0.96	1
#30#305	30	5	0	137	162	0.9	1
#30#2781	341	101	37	137	514	0.9	0.92
#144#1377	0	0	0	154	163	0.875	1
#145#4568	0	0	0	183	544	0.87	1

Table 2: The set of instances.

We are going to compare the branch-and-price method described here with the proposed ILP model solved using the software CPLEX and four other methods developed by Decision-Brain. The «ILP» corresponds to the integer linear program implemented and tested with Cplex. The «CP» corresponds to constraint programming using ILOG IBM Scheduler constraints and variables and tested with CPOptimizer. The «H» corresponds to a heuristic, kept confidential. The «H+X» method corresponds to start the X optimization with a first solution computed by the heuristic. The «LNS» corresponds to Large Neighborhood Search using the best insert algorithm on different neighborhood operators. The «BP» column corresponds to our branch-and-price scheme.

Instance	Model	Obj.	CPU(s)	Gap
#30#256	ILP	1.1379E7	1802	105%
#30#256	CP	2.1393E7	1803	9.9%
#30#256	H	2.1475E7	17	9.5%
#30#256	H+ILP	2.1475E7	1806	9.5%
#30#256	H+CP	2.1475E7	1803	9.5%
#30#256	H+LNS	2.1687E7	1643	8.5%
#30#305	ILP	2364770.0	1801	525%
#30#305	CP	1.3094E7	1802	17%
#30#305	H	1.1269E7	21	35%
#30#305	H+ILP	1.1278E7	1807	35%
#30#305	H+CP	1.3102E7	1802	16.9%
#30#305	H+LNS	1.3314E7	1678	15.1%

Table 3: Results for instances with 30 technicians and 256 tasks and 30 technicians and 305 tasks with a resolution time of 30 minutes.

The ILP model does not scale for the medium and large size instances, we obtain a high gap on the medium size instances (100% for instance#30#256 and 525% for instance#30#305). The CP model scales, and in some cases, achieves better results than the heuristic and meta-heuristic resolution method (H + LNS). One can see that the behavior of the CP is very close to the behavior of heuristics. Indeed, the CP obtains a good quality solution in a short time thanks to solver constraint propagators by cutting non-solution domain values. It is interesting to note that the heuristic gives a good solution in just a few seconds.

Now, we display the results obtained with the branch-and-price scheme. Table 5 gives the results for the different instances. In this table, CP is used to solve the PSP and the adopted tree traversal strategy is the Best-first search strategy in order to converge quickly towards a good solution.

Instance	Model	Obj.	CPU(s)	Status
#144#1377	CP	1.0995E7	3610	Feasible
#144#1377	H	1.4247E7	58	Feasible
#144#1377	H+CP	1.4360E7	3591	Feasible
#144#1377	H+LNS	1.4738E7	2877	Feasible
#145#4568	H	1.0421E8	221	Feasible
#145#4568	H+LNS	1.0645E8	3472	Feasible
#30#2781	CP	1.7181E7	3650	Feasible
#30#2781	H	2.3314E7	20	Feasible
#30#2781	H+CP	2.3343E7	3584	Feasible
#30#2781	H+LNS	2.3314E7	3636	Feasible

Table 4: Results for large instances with a resolution time of 1 hour.

The column «nodes» refers to the number of nodes browsed in the branch-and-price tree. The column «#col» represents the total number of columns in the master problem.

Instance	Model	Obj.	CPU(s)	nodes	#col	gap
#30#256	BP	2.1690E7	607.0	7	436	8.4%
#30#256	H+BP	2.1599E7	600.0	9	421	8.9%
#30#305	BP	1.3322E7	612.0	13	484	15%
#30#305	H+BP	1.3232E7	601.0	14	424	15,8%
#30#256	BP	2.0802E7	1812.0	15	565	13%
#30#256	H+BP	2.1712E7	1808.0	17	544	8.3%
#30#305	BP	1.3263E7	1809.0	13	606	15.5%
#30#305	H+BP	1.3251E7	1816.0	23	558	15.6%

Table 5: Results for branch-and-price on medium-sized instances with a resolution time limit of 10 and 30 minutes using constraint programming for the PSP.

One can observe that solutions obtained with the branch-and-price are better than solutions obtained with the constraint programming model and even than solutions computed by heuristic followed by the local search. However, as the ILP model, the branch-and-price does not scale. Large instances are too substantial to be treated by our branch-and-price scheme in a reasonable time. These results nevertheless show the interest in using the hybridization between column generation and constraint programming.

CONCLUSION

In this paper, we propose a branch-and-price scheme dedicated to solving a *WSRP* problem in the presence of large-scale highly constrained real-world instances when the time limit is bounded. With this method, we were able to obtain good results, better than LNS or CP in some instances. However this method is not scalable, therefore results for large instances are missing.

Using a dynamic programming label algorithm to solve the sub-problem should speed the solving process up by adding multiples improving routes in the master problem at each step of the column generation algorithm while decreasing memory usage of each sub-problem. On the column generation phase, we solve a sub-problem for each technician, therefore solving all the sub-problem is time-consuming. One could try to group technicians that have some similar characteristics to reduce the time spent solving sub-problem.

REFERENCES

[1] C. Barnhart, E.L. Johnson, G.L. Nemhauser, M.W.P Savelsbergh, and P.H. Vance. 1998. Branch-and-price: Column generation for solving huge integer programs. Operations research 46, 3 (1998), 316–329.

[2] J. Blazewicz, J.K. Lenstra, and A.H.G.R. Kan. 1983. Scheduling subject to resource constraints: classification and complexity. Discrete Applied Mathematics 5, 1 (1983), 11–24. DOI:http://dx.doi.org/10.1016/0166-218X(83)90012-4

[3] J.A. Castillo-salazar. 2014. Qu , Rong (2014) Workforce scheduling and routing problems : literature survey and computational study . Annals of Operations Research . pp . 1-29 . ISSN 1572-. 239 (2014), 1–29.

[4] J.A. Castillo-Salazar, D. Landa-Silva, and R. Qu. 2015. A greedy heuristic for workforce scheduling and routing with time-dependent activities constraints. International Conference on Operations Research and Enterprise Systems (ICORES 2015) (2015).

[5] E. Choi and D.W. Tcha. 2007. A column generation approach to the heterogeneous fleet vehicle routing problem. Computers & Operations Research 34, 7 (2007), 2080–2095.

[6] J.F. Cordeau, G. Laporte, F. Pasin, and S. Ropke. 2010. Scheduling technicians and tasks in a telecommunications company. Journal of Scheduling 13, 4 (2010), 393–409. DOI:http://dx.doi.org/10.1007/s10951-010-0188-7

[7] M. Desrochers, J.K. Lenstra, and M.W.P. Savelsbergh. 1990. A classification scheme for vehicle routing and scheduling problems. European Journal of Operational Research 46, 3 (1990), 322–332. DOI:http://dx.doi.org/10.1016/0377-2217(90)90007-X

[8] J. Desrosiers, Y. Dumas, M. Solomon, and F. Soumis. 1995. Time Constrained Routing and Scheduling. Handbooks in Operations Research and Management Science 8, C (1995), 35–139. DOI:http://dx.doi.org/10.1016/S0927-0507(05)80106-9

[9] A. Dohn, M.S. Rasmussen, and J. Larsen. 2011. The vehicle routing problem with time windows and temporal dependencies. Networks 58, 4 (2011), 273–289.

[10] M. Dror. 1994. Note on the Complexity of the Shortest Path Models for Column Generation in VRPTW. 42 (10 1994), 977–978.

[11] D. Feillet. 2010. A tutorial on column generation and branch-and-price for vehicle routing problems. 4OR: A Quarterly Journal of Operations Research 8, 4 (2010), 407–424. https://hal-emse.ccsd.cnrs.fr/emse-00505959

[12] D. Feillet, P. Dejax, M. Gendreau, and C. Gueguen. 2004. An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. Networks 44, 3 (2004), 216–229.

[13] R.L. Graham, E.L. Lawler, J.K. Lenstra, and A.H.G. Rinnooy Kan. 1979. Optimization and Approximation in Deterministic Sequencing and Scheduling: a Survey. Annals of Discrete Mathematics 5 (1979), 287–326.

[14] C. A. J. Hurkens. 2009. Incorporating the strength of MIP modeling in schedule construction. RAIRO - Operations Research 43 (2009), 409–420. DOI:http://dx.doi.org/10.1051/ro/2009026

[15] S. Irnich and G. Desaulniers. 2005. Shortest path problems with resource constraints. In Column generation. Springer, 33–65.

[16] P. Laborie and J. Rogerie. 2008. Reasoning with Conditional Time-Intervals. Proceedings of the Twenty-First International Florida Artificial Intelligence Research Society Conference, May 15-17, 2008, Coconut Grove, Florida, USA (2008), 555–560.

[17] P. Laborie, J. Rogerie, P. Shaw, and P. Vilim. 2009. Reasoning with Conditional Time-Intervals. Part II: An Algebraical Model for Resources. FLAIRS Conference (2009), 201–206. http://www.aaai.org/ocs/index.php/FLAIRS/2009/paper/viewPDFInterstitial/60

[18] B. J. Lageweg, J.K. Lenstra, E.L. Lawler, and A.H.G. Rinnooy Kan. 1982. Computer-Aided complexity classification of combinational problems. *Commun. ACM* 25, 11 (1982), 817–822. DOI:http://dx.doi.org/10.1145/358690.363066

[19] E.L. Lawler, J.K. Lenstra, A.H.G Rinnooy Kan, and D.B. Shmoys. 1993. Sequencing and scheduling: Algorithms and complexity. (1993), 445–522. DOI:http://dx.doi.org/10.1016/S0927-0507(05)80189-6

[20] J.K Lenstra. 2013. Personnel scheduling: A literature review. (2013). DOI:http://dx.doi.org/10.1016/j.cam.2008.10.038

[21] S. Pokutta and G. Stauffer. 2009. France Telecom workforce scheduling problem: A challenge. RAIRO - Operations Research 43, 4 (2009), 375–386. DOI:http://dx.doi.org/10.1051/ro/2009025

[22] M.S. Rasmussen. 2010. The Home Care Crew Scheduling Problem: Preference-Based Visit Clustering and Temporal Dependencies. May (2010).

[23] M. Solomon. 1987. Algorithms for the vehicle routing and scheduling problems with time window constraints. Operations research 35, 2 (1987), 254–265.

[24] M. Solomon and J. Desrosiers. 1988. Survey Paper — Time Window Constrained Routing and Scheduling Problems. March 2014 (1988).

[25] J.N. Tsitsiklis. 1992. Special cases of traveling salesman and repairman problems with time windows. Networks 22, 3 (1992), 263–282. DOI:http://dx.doi.org/10.1002/net.3230220305

Distributionally robust airline fleet assignment problem

Marco Silva
LIA, Université d'Avignon et des Pays du Vaucluse
Avignon
marco.costa-da-silva@univ-avignon.fr

Michael Poss
LIRMM, Université de Montpellier 2
Montpellier
michael.poss@lirmm.fr

ABSTRACT

In this work we consider the airline fleet assignment problem and we experiment with a robust solution where passenger demand is uncertain. To mitigate conservativeness of the classical robust optimization we consider a two-stage distributionally robust objective formulation. Our main contribution with respect to the airline fleet management problem literature lies in the modeling characteristics of our proposal.

We benchmark against current deterministic and robust fleet assignment formulations and verify solution performance results through simulation.

KEYWORDS

Distributionally robust optimization, Affine decision rules, Integer Programming, Airline fleet assignment

1 INTRODUCTION

Once an airline decides when and where to fly (flight legs) by developing its flight schedule, the next decision is determining the type of aircraft, or fleet, that should be used on each of the flight legs defined within the flight schedule. This process is called fleet assignment and its purpose is to assign fleet types to flight legs, subject to an available number of aircrafts and conservation of aircraft flow requirements, such as to maximize profits with respect to captured passenger demand. This decision needs to be made well in advance of departures when passenger demand is still highly uncertain. The factors that influence schedulers when assigning fleet types to various flights are: passenger demand, seating capacity, operational costs, and availability of maintenance at arrival and departure stations. One important requirement of the fleet assignment is that the aircraft must circulate in the network of flights. These so-called balance constraints are enforced by using time lines to model the activities of each fleet type. The period for which the assignment is done is normally one day for domestic flights.

Profit maximization is normally defined in terms of unconstrained revenue minus assignment cost. Unconstrained revenue of a flight leg is the maximum attainable revenue for that particular flight regardless of assigned capacity. Assignment cost, a function of the assigned fleet type, includes the flight operating cost, passenger carrying related cost and spill cost. Spill cost on a flight is the revenue lost when the assigned aircraft for that flight cannot accommodate every passenger. The result is that either the airline spills some passengers to other flights in its own network (in which case these passengers are recaptured by the airline), or they are spilled to other airlines.

In [12] the authors develop a two-stage stochastic programming model for integrated flight scheduling and fleet assignment where the fleet family assigned to each scheduled flight leg is decided at the first-stage. Then, the fleet type to assign to each flight leg is decided at the second-stage based on demand and fare realization. Sample average approximation (SAA) algorithm is then used to solve the problem and provide information on the quality of the solution.

In [9] the authors propose a new model based on itinerary grouping to mitigate the effect of demand uncertainty. Their itinerary group fleet assignment model deals with the difficulties caused by itinerary forecast by replacing them with aggregated demand forecasts. The authors affirm that an itinerary-based representation of demand (see Section 2 for details) has led to a high granularity of demand, making it hard to predict.

In this work, as an alternative to previous works presented, we propose a two-stage data-driven distributionally robust optimization model to address the question of airline robust planning for the fleet assignment problem. Our main contribution with respect to the airline fleet management problem literature lies in this novel modeling approach.

We adopt the concept of robust optimization as defined in [5] and [6] in that the demand uncertainty belongs to a known deterministic uncertainty set. In fact, we consider this uncertainty set as the support for the family of probability distributions associated with our random passenger demand parameter. We consider a data-driven approach by which this uncertainty set is constructed from available historical data. We assume that historical unconstrained (not subject to capacity issues) itinerary demand data is available and that we can use this historical data to predict future demand. By constructing the uncertainty set from historical data we are able to capture correlations between demands of different itineraries and thus mitigate the granularity demand effect as pointed out in [9].

On the other hand, we consider different modeling alternatives to mitigate conservatism of a robust approach. Since fleet assignment is a repetitive process, where fleet assignment decisions are made on daily basis, we mitigate the conservatism of the worst-case objective of classical robust optimization and consider a distributionally robust optimization approach on which we optimize the worst case expected performance on a set constituted by an infinite number of probability distributions, named ambiguity set (see [10] for main concepts). We also propose a two-stage model, as introduced in [4] where, although all the fleet assignments decisions are first stage, the calculation of lost revenue (spill) is only done after realization of uncertainty.

To facilitate handling large-scale fleet assignment problems, we propose the use of principal component analysis techniques to reduce dimension of the uncertainty set and the use of affine decision rules for our two-stage problem as approximations to improve time performance of our algorithms.

2 FLEET ASSIGNMENT FORMULATIONS

The fleet assignment model is typically formulated as a mixed-integer program. One can see the work in [16] for a survey of different modeling approaches for the problem.

ISBN 978-3-89318-079-0 on OpenProceedings, ISSN 2510-7437 DOI: 10.5441/002/inoc.2019.13

In [1] the author first introduced for the fleet assignment problem a time-space network model to represent the availability of the fleet at each airport in the course of time. The proposed model resulted in a linear program that could either maximize profit or minimize operations cost.

In [11] the authors use the time-space network model and develop a large-scale integer program for fleet assignment. They propose several preprocessing techniques, namely node aggregation and isolated islands at stations, in order to reduce problem complexity.

In these two works demand is expressed for a specific flight leg and, therefore, these works do not capture demand dependencies between legs. This is because demand is defined for airline itineraries that can be comprised by multiple flight legs. Variations of demand in one itinerary flight leg will affect the others legs. This is called the network effect and it was taken into consideration in the model defined in [2]. There, the authors use the time-space network model and consider the effect of recapturing, where passengers spills from one itinerary can be redirected to alternative itineraries. In their model demand is deterministic.

The above model is reference for our work, with the difference that we do not consider recapturing. We replicate here the itinerary based formulation as presented in [9] where the authors also explicitly deal with itinerary fare classes to better capture the revenue dimension by favoring higher classes instead of considering all the fare classes at the same level. We present notations and formulation used.

Sets

P : the set of itinerary fare classes, indexed by p
A : the set of airports, indexed by o
L : the set of flight legs, indexed by i
K : the set of fleet types, indexed by k
T : the sorted set of all relevant event times (leg departures or aircraft availability) at all airports, indexed by t
$CL(k)$: the set of flight legs that pass the count time when flown by fleet type k
$I(k, o, t)$: the set of inbound flight legs to node (k, o, t)
$O(k, o, t)$: the set of outbound flight legs from node (k, o, t)

Decision variables

t_p : the number of passengers requesting itinerary fare class p and spilled by the model because of the capacity limit.
f_{ki} : binary variable equal to 1 if fleet type k is assigned to flight leg i, 0 otherwise.
y_{kot^+} : the number of fleet type k that are on the ground at airport o immediately after time t.
y_{kot^-} : the number of fleet type k that are on the ground at airport o immediately before time t. If $t^{'}$ is the time of the first event occurring after t, then $y_{kot} = y_{kot'^-}$

Data

$SEATS_k$: the number of seats available on aircraft of fleet type k.
c_{ki} : the cost of operating leg i with fleet type k.
N_k : the number of aircraft in fleet type k.
D_p : the unconstrained demand for itinerary fare class p.
$fare_p$: the fare class for itinerary p.
δ_i^p : a binary flag equal to 1 if itinerary fare class includes flight leg i, 0 otherwise.
$count\ time$: the time at which a snapshot of fleet utilization is taken to ensure consistency with the available fleet.
t_m : the last event before the $count\ time$, $t_m = count\ time^-$.

$(IFAM)$

$$\text{min} \quad \sum_{i\in L, k\in K} c_{ki} f_{ki} + \sum_{p\in P} fare_p\, t_p \tag{1}$$

$$\text{s.t.} \quad \sum_{k\in K} f_{ki} = 1 \qquad \forall i \in L \tag{2}$$

$$\sum_{i\in I(k,o,t)} f_{ki} + y_{kot^-} = \sum_{i\in O(k,o,t)} f_{ki} + y_{kot^+}, \quad \forall k \in K, o \in A, t \in T \tag{3}$$

$$\sum_{o\in A} y_{kot_m} + \sum_{i\in CL(k)} f_{ki} \le N_k \qquad \forall k \in K \tag{4}$$

$$\sum_{p\in P} \delta_i^p D_p - \sum_{p\in P} \delta_i^p t^p \le \sum_{k\in K} f_{ki}\, SEATS_k \qquad \forall i \in L \tag{5}$$

$$t_p \le D_p \qquad \forall p \in P \tag{6}$$

$$f_{ki} \in \{0, 1\}, y_{kot} \in \{0, 1\}, t_p \ge 0, \quad \forall p \in P, k \in K, i \in L, o \in A, t \in T$$

The objective function (1) minimizes the total cost of operations plus the cost related to spilled itinerary fare class demand. This minimization is equivalent to profit maximization. Constraints (2) are the leg coverage constraints. Each flight leg has to be operated by exactly one aircraft type. The flow conservation constraint related to each single event is ensured through constraints (3). The limited size of each fleet is respected through constraints (4). The count time can be seen as a fixed time where a cut is applied on the network to ensure that the total aircraft of each fleet type k on the ground at all airports plus those flying at the time must not exceed the total aircraft N_k available for type k. The capacity constraints (5) ensure that satisfied demand fits with the number of seats available on any given leg. Last, constraints (6) ensure that spill does not exceed unconstrained demand for any given itinerary fare class.

We now propose a two-stage distributionally robust optimization formulation derived from *IFAM* formulation to incorporate the random nature of passenger demand vector D. Distributionally robust optimization is an emerging and effective method to address the inexactness of probability distributions of uncertain parameters.

We formulate our problem assuming that passenger demand spill decision variable is a second-stage variable. This way passenger demand spill is only defined after realization of uncertainty and we represent this dependency defining it as a function map $t_p(D)$. We also assume the uncertainty of vector D is represented through a probability distribution $\mathbb{P}$ that belongs to a family of distributions $\mathcal{D}$.

We present the formulation developed.

($DIFAM$)

$$\min \quad \sum_{i\in L,k\in K} c_{ki} f_{ki} + \sup_{\mathbb{P}\in\mathcal{D}} \mathbb{E}_{\mathbb{P}}[Q(f,D)] \tag{7}$$

s.t.

$$\sum_{k\in K} f_{ki} = 1 \qquad \forall i \in L \tag{8}$$

$$\sum_{i\in I(k,o,t)} f_{ki} + y_{kot^-} = \sum_{i\in O(k,o,t)} f_{ki} + y_{kot^+}, \quad \forall k \in K, o \in A, t \in T \tag{9}$$

$$\sum_{o\in A} y_{kot_m} + \sum_{i\in CL(k)} f_{ki} \leq N_k \qquad \forall k \in K \tag{10}$$

$$f_{ki} \in \{0,1\}, y_{kot} \in \{0,1\}, \quad \forall i \in L, k \in K, o \in A, t \in T$$

where

$Q(f,D) =$

$$\min \quad \sum_{p\in P} fare_p \, t_p(D) \tag{11}$$

$$\sum_{p\in P} \delta_i^p D_p - \sum_{p\in P} \delta_i^p t^p(D) \leq \sum_{k\in K} f_{ki}\, SEATS_k \qquad \forall i \in L \tag{12}$$

$$t_p(D) \leq D_p \qquad \forall p \in P \tag{13}$$

$$t_p \geq 0, \forall p \in P$$

The cost $\sum_{i\in L,k\in K} c_{ki} f_{ki}$ incurred during the first stage is deterministic. In progressing to the second-stage, the random passenger demand vector D is realized. We can then determine the cost incurred at the second-stage. For a given first stage fleet type assignment decision, f, and a realization of the random passenger demand vector, D, we evaluate the second-stage cost via the linear optimization problem, $Q(f,D)$. Since the fleet type assignment is a repetitive daily process and the true probability distribution of D is unknown and belong to a family of distributions set $\mathcal{D}$ we are interested in the worst case expectation $\sup_{\mathbb{P}\in\mathcal{D}} \mathbb{E}_{\mathbb{P}}[Q(f,D)]$. Note that it is a relatively complete recourse problem because any first-stage solution leads to a feasible second-stage solution.

In order to be able to deal with large scale problems, our two-stage distributionally robust optimization formulation must admit a tractable reformulation. The reformulation is closely related to the choices of ambiguity set that we make. On the other hand these choices must correctly reflect properties of historical data available.

In the next section we show that defining ambiguity sets by linear relationships of uncertainty parameters and approximating second-stage variables as affine functions of uncertainty parameters yields a tractable problem. We will also use techniques of uncertainty dimensionality reduction as a further compromise between optimality and tractability.

3 TWO-STAGE DISTRIBUTIONAL REFORMULATION

3.1 Dimensionality reduction

In real case examples, the dimension of the random passenger demand vector D can be in the range of thousands of itineraries. This can impact performance of the formulation $DIFAM$. Employing dimensionality reduction techniques to reduce the number of random variables under consideration can improve performance of our formulations.

Here we assume there is a set $\mathcal{W}'$ of N demand data samples available, $\mathcal{W}' = \{D^{(i)}\}_{i=1}^N$, based on historical data, and use this set to calculate the mean vector $\bar{D}$, variance vector $\hat{D}$ and covariance matrix $cov(D^{(i)})$.

A linear technique for dimensionality reduction, principal component analysis, performs a mapping of the data to a lower-dimensional space in such a way that the variance of the data in the low-dimensional representation is maximized. Intuitively, we change the system of coordinates and define this system by new vectors Y, but we select only some of them, therefore reducing dimension of the system. The new system of coordinates, vectors $\{Y^c\}_{c=1}^C$, are in fact normalized eigenvectors of the covariance matrix $cov(D^{(i)})$, where c is the index of the selected eigenvectors. We refer to [17] for more details on principal component analysis (PCA).

We execute a procedure to express the passenger demand vectors, $D^{(i)}$, in the new system of coordinates, but before we normalize the vectors $D^{(i)}$, using $\bar{D}$ and $\hat{D}$. Therefore we define $D^{(i)'} = (D^{(i)} - \bar{D})./\hat{D}$, where ./ is a component wise division of vectors.

We then compute the coordinates, $X_c^{(i)}$, in the system of coordinates of the principal components vectors, $\{Y^c\}_{c=1}^C$, where the principal component vector has dimension $|P|$. The value of $X_c^{(i)}$ results from the expression:

$$X_c^{(i)} =< D^{(i)'}, Y^c >, \tag{14}$$

where $<,>$ is a dot product.

In the following we need the random vector to be nonnegative, which may not be the case of components $X_c^{(i)}$. Hence, we introduce a new random vector $\xi^{(i)}$ where each component will vary in the nonnegative interval $[0,1]$. Each component of $\xi^{(i)}$ is defined as

$$\xi_c^{(i)} = (X_c^{(i)} - \max_i (X_c^{(i)}))/(\min_i (X_c^{(i)}) - \max_i (X_c^{(i)})), \tag{15}$$

where $\max_i (X_c^{(i)})$, $\min_i (X_c^{(i)})$ are, respectively, the maximum and minimum projection component values along each vector Y^c considering all instances, $i \in \{1,\ldots,N\}$. $X_c^{(i)}$ varies in the interval $[\min_i (X_c^{(i)}), \max_i (X_c^{(i)})]$ and as consequence $\xi_c^{(i)}$ will vary in the interval $[0,1]$.

Using the above definition of $\xi^{(i)}$, we can define the components of each demand vector $D^{(i)}$ as

$$D_p^{(i)} = D1_p + \sum_{c=1}^{C} D2_{pc} \xi_c^{(i)}, \tag{16}$$

where

$$D1_p = \bar{D}_p + \hat{D}_p \sum_{c=1}^{C} \min_i (X_c^{(i)}) Y_p^c, \tag{17}$$

$$D2_{pc} = \hat{D}_p (\max_i (X_c^{(i)}) - \min_i (X_c^{(i)})) Y_p^c \tag{18}$$

This is an important step in order to guarantee positive definite matrices in the algorithm developed in Section 4 for our distributionally robust ambiguity set.

3.2 Ambiguity set and first-order deviation moment functions

The tractability of a distributionally robust linear optimization problem is dependent on the choice of the ambiguity set. Several ambiguity sets have been proposed in the literature. In particular, moment-based uncertainty sets assume that all distributions in the distribution family share the same moment information. By leveraging conic duality many distributionally robust optimization problems with moment-based ambiguity sets can, in general, be reformulated equivalently as convex problems. Although these problems can be solved theoretically in polynomial time, they are not efficient for large-scale instances.

In [8], a moment-based second-order conic representable ambiguity set, $\mathcal{D}$, is defined as

$$\mathcal{D} = \left\{ \mathbb{P} \in \mathcal{P}_0(\mathbb{R}^{|P|}) \;\middle|\; \begin{array}{c} D \in \mathbb{R}^{|P|} \\ \mathbb{E}_{\mathbb{P}}[GD] = \mu \\ \mathbb{E}_{\mathbb{P}}[g_i(D)] \le \gamma_i \quad \forall i \in I \\ \mathbb{P}(D \in U) = 1 \end{array} \right\}.$$

We assume random passenger demand vector D, but the same results can be derived substituting for dimensional reduced random vector ξ derived in the previous section. $\mathcal{P}_0(\mathbb{R}^{|P|})$ represents the set of all probability distributions in $\mathbb{R}^{|P|}$ and new parameters are defined as $G \in \mathbb{R}^{n_1 \times |P|}$, $\mu \in \mathbb{R}^{n_1}$, $\gamma \in \mathbb{R}^{|I|}$, SOC (second-order conic) representable support set $U \in \mathbb{R}^{|P|}$ and SOC representable functions $g_i \in \mathbb{R}^{|P| \times 1}$.

$\mathcal{D}$ only contains valid distributions supported over the support set U and moment information of uncertainties are characterized via functions g_i. The equality expectation expression allow the modeler to specify the mean values of D.

The authors of [8] further reformulate the ambiguity set $\mathcal{D}$ as a projection of an extended ambiguity set $\bar{\mathcal{D}}$ by introducing an I-dimensional auxiliary random vector u in

$$\bar{\mathcal{D}} = \left\{ \mathbb{Q} \in \mathcal{P}_0(\mathbb{R}^{|P|} \times \mathbb{R}^{|I|}) \;\middle|\; \begin{array}{c} (D,u) \in \mathbb{R}^{P} \times \mathbb{R}^{|I|} \\ \mathbb{E}_{\mathbb{Q}}[GD] = \mu \\ \mathbb{E}_{\mathbb{Q}}[u] \le \gamma_i \quad \forall i \in I \\ \mathbb{P}((D,u) \in \bar{U}) = 1 \end{array} \right\}$$

where $\bar{U}$ is the lifted support set defined as

$$\bar{U} = \left\{ (D,u) \in \mathbb{R}^{|P|} \times \mathbb{R}^{|I|} \;\middle|\; \begin{array}{c} D \in U \\ g_i(D) \le u_i \quad \forall i \in I \end{array} \right\}$$

They observe that the lifted ambiguity set has only linear expectation constraints and show that the adaptive distributionally robust optimization problem can be reformulated as a classical robust optimization problem with uncertainty set $\bar{U}$.

To be able to reformulate adequately our fleet assignment problem *DIFAM* we must then define an ambiguity set $\mathcal{D}$ that will lead to a polyhedron lifted support set $\bar{U}$.

In [15] the authors define first-order deviation moment-based functions $g_i(.)$ that are second-order conic representable as piecewise linear functions

$$g_i(D) = \max\{h_i^T D - q_i, 0\} \quad \forall i \in I.$$

They can be understood as the first-order deviation of uncertain parameters along a certain projection h_i truncated at q_i. We apply these moment-based functions to our problem and also assume that the support set U is a polyhedron. We then define our ambiguity set $\mathcal{D}$ as

$$\mathcal{D} = \left\{ \mathbb{P} \in \mathcal{P}_0(\mathbb{R}^{|P|}) \;\middle|\; \begin{array}{c} D \in \mathbb{R}^{|P|} \\ \mathbb{E}_{\mathbb{P}}[\max\{h_i^T D - q_i, 0\}] \le \gamma_i \quad \forall i \in I \\ \mathbb{P}(D \in U) = 1 \end{array} \right\}$$

and the lifted support set $\bar{U}$ will be a polyhedron given as

$$\bar{U} = \left\{ (D,u) \in \mathbb{R}^{|P|} \times \mathbb{R}^{|I|} \;\middle|\; \begin{array}{c} D \in U \\ 0 \le u_i \quad \forall i \in I \\ h_i^T D - q_i \le u_i \quad \forall i \in I \end{array} \right\}$$

3.3 Affine decision rules

With the above definition of lifted support set we can apply, to our *DIFAM* formulation, the reformulation proposed by [8] for the adaptive distributionally robust optimization problem, approximating second-stage variables t_p as affine functions of the lifted support set parameters (D,u), $t_p(D,u) = t_p^0 + \sum_{i \in P} t_{pi}^1 D_i + \sum_{i \in I} t_{pi}^2 u_i$.

This reformulation is based on the dualization of the inner problem of *DIFAM*, $\sup_{\mathbb{P} \in \mathcal{D}} \mathbb{E}_{\mathbb{P}}[Q(f,D)]$, and by introducing Lagrangian multipliers r and β to it (alternatively see [15] for a summarized proof of this reformulation). This leads to the following classical robust optimization problem:

$$\begin{array}{lll}
(RRIFAM) & & \\
\min & \sum_{i \in L, k \in K} c_{ki} f_{ki} + r + \sum_{i \in I} \gamma_i \beta_i & \\
\text{s.t.} & & \\
& r + \sum u_i \beta_i \ge \sum_{p \in P} fare_p t_p(D,u), & \\
& \qquad \forall (D,u) \in \bar{U} & \\
& \sum_{p \in P} \delta_i^p D_p - \sum_{p \in P} \delta_i^p t^p(D,u) \le \sum_{k \in K} f_{ki}\, SEATS_k, & \\
& \qquad \forall i \in L,\ \forall (D,u) \in \bar{U} & \\
& t_p(D,u) \le D_p, & \\
& \qquad \forall (D,u) \in \bar{U} & \\
& \sum_{k \in K} f_{ki} = 1, & \\
& \qquad \forall i \in L & \\
& \sum_{i \in I(k,o,t)} f_{ki} + y_{kot^-} = \sum_{i \in O(k,o,t)} f_{ki} + y_{kot^+}, & \\
& \qquad \forall k \in K, o \in A, t \in T & \\
& \sum_{o \in A} y_{kot_m} + \sum_{i \in CL(k)} f_{ki} \le N_k, & \\
& \qquad \forall k \in K & \\
& t_p(D,u) = t_p^0 + \sum_{i \in P} t_{pi}^1 D_i + \sum_{i \in I} t_{pi}^2 u_i, & \\
& \qquad \forall p \in P & \\
& r \in \mathbb{R}, \beta_i \ge 0, f_{ki} \in \{0,1\}, y_{kot} \in \{0,1\}, t_p \ge 0, & \\
& \qquad \forall p \in P, k \in K, i \in L, o \in A, t \in T &
\end{array}$$

4 DATA-DRIVEN AMBIGUITY SET

A desirable ambiguity set should flexibly adapt to the intrinsic structure behind real data, thereby well characterizing $\mathbb{P}$ and attempting to reduce natural conservatism of robust solutions. In face of complicated distributional geometry, making prior assumptions on the form of probability distribution or using classical uncertainty sets to describe their support have limited modeling power. With this in mind we adopt a data-driven methodology to construct and define parameters of the support set and moment-based functions associated with our ambiguity set.

4.1 Support Set

In what follows, we use the technical approach of [14] to construct a support set U from data samples of the random variable ξ. We assume there is a set $\mathcal{W}$ of N data samples available, $\mathcal{W} = \{\xi^{(i)}\}_{i=1}^{N}$, and this set is constructed from the sample of demand vectors, $D^{(i)}{}_{i=1}^{N}$, as explained in Subsection 3.1.

In [14] the authors propose piecewise linear kernel-based support vector clustering (SVC) as a machine learning technique tailored to data-driven robust optimization. They explore the SVC's secondary effect that evolves the data samples inside a sphere in a high-dimensional space [3]. They use this sphere to characterize the uncertainty set. This mapping of data points to a high-dimensional space is done by means of a kernel function. Using well known techniques of machine learning they define a linear kernel that, in turn, is used to define a polyhedral region evolving the data in the original space.

Using these techniques, we define our data-driven support set U as the region inside or in the borders of this sphere. This sphere is given by the expression

$$U = \{\xi \mid K(\xi,\xi) - 2\sum_{i=1}^{N} \alpha_i K(\xi,\xi^{(i)}) + \\ + \sum_{i=1}^{N}\sum_{k=1}^{N} \alpha_i \alpha_k K(\xi^{(i)},\xi^{(k)}) \leq R^2\}$$

Parameters α and R are derived by applying Lagrangian relaxation to the original formulation and the linear kernel is given by

$$K(\xi^{(i)},\xi^{(j)}) = \sum_{k=1}^{N} l_k - |\xi^{(i)} - \xi^{(j)}|_1,$$

where $l_k = \max_{1\leq i\leq N} \xi_k^{(i)} - \min_{1\leq i\leq N} \xi_k^{(i)}$. We refer to [3] for details.

4.2 Moment-based functions

For moment-based functions we adopt the work of [15] where the authors define a two-step procedure for determining parameters h_i and q_i of our piecewise linear functions in order to capture meaningful information from available data.

The directions h_i are based on principal component analysis (PCA) such that the data space becomes decorrelated along each direction and the information overlap between different directions is slight. Since our random vector ξ already comprises decorrelated components we adopt vector h_i as standard unit vectors e_i.

After that, several truncation points $\{q_i\}$ are set along each direction h_i. For each direction h_i we choose $2J+1$ well-distributed truncation points. The first truncation point is set as the mean value $\bar{\xi}_i$ and the remaining $2J$ ones around the mean $\bar{\xi}_i$ symmetrically based on a fixed step-size given as the variance $\hat{\xi}_i$ along the i-th direction.

In this way, we will have $C(2J+1)$ piecewise functions $g_i(.)$ in total in the ambiguity set.

Intuitively, the parameter J can be deemed as the "size" of the ambiguity set, which can be manipulated to adjust the conservatism of the model. The more truncation points we have, the more statistical information will be incorporated, which leads to a smaller ambiguity set as well as a less conservative solution.

After determining the value of h_i and q_i, the next step is to estimate the parameters γ_i empirically based on N available data samples:

$$\gamma_i = \frac{1}{N}\sum_{j=1}^{N} \max(h_i^T \xi^{(j)} - q_i, 0)$$

Intuitively, with the values of size parameters γ_i increasing, the DRO model becomes more conservative.

5 IMPLEMENTATION AND RESULTS

5.1 Implementation details

We report on experiments conducted with the formulations proposed for the airline fleet assignment problem. Our objective is to verify the performance of each solution in a long run operation since fleet assignment is a daily repetitive process.

For our purposes, we create a small-sized hub-and-spoke airline instance, in which a unique major airport serves as a central point for coordinating flights to and from other airports. This way all our itineraries are composed of a maximum of two flight legs. We consider a structure of 9 airports, 3 fleet types and 24 daily itineraries based on three fare classes. A flight schedule with 21 flight legs is created and they are used to compose the daily itineraries.

We test this operation under four different problem formulations: *IFAM*, *RRIFAM*, as already presented in this study and two other formulations *RIFAM* and *RIFAM*2. Formulation *RIFAM* is a standard two-stage robust formulation where the objective is given as the worst case performance and dimensionality reduction is performed the same way as for *RRIFAM*. Formulation *RIFAM*2 is the same as *RIFAM* where no dimensionality reduction is performed.

We randomly generate a set of 400 demand vectors. They are designed in a way that many itinerary demands are highly correlated.

We use 100 demand vectors as historical training data to create the ambiguity set of formulation *RRIFAM* and the 300 others to simulate the airline operating period.

We use naive approaches to determine demand vectors for formulations *IFAM*, *RIFAM* and *RIFAM*2. For formulation *IFAM* we consider three demand scenarios of low, medium and high total demand and consider the average of these three scenarios as input to our *IFAM* formulated problem. For formulation *RIFAM* and *RIFAM*2 we consider maximum and minimum demand values for each leg and consider a box uncertainty set where each demand component varies within this interval.

With the solution of formulations *IFAM*,*RIFAM* and *RRIFAM* we simulate an airline operating period of 300 days and calculate an objective of total operating costs plus total loss revenue. We compare simulation results of the three formulations where our focus is on analyzing objective value and time performance.

Conservatism regulation parameters of our ambiguity set are fixed as $v = 0.6$ and $J = 0$. With $v = 0.6$, 100% of demand vectors were considered inside or in the border of the support set (no outliers). We calculate parameter C so that the sum of variances in the direction of each principal component considered sums

	IFAM	*RIFAM*	*RIFAM*2	*RRIFAM*
Objective value	335747	638817	651081	488135
Total Time (s)	4.5	239.77	10950.47	9041.14
Number of variables	789	982	1366	1181
Number of constraints	450	-	-	-
Number of iterations	-	32	49	81
Simulation Total cost*	1.38e8	1.35e8 (2.2%)	-	1.32e8 (4.3%)

*In parenthesis percentage gain when compared to worst result

Table 1: Implementation and performance comparison of fleet assignment formulations

up to a minimum of 90% of the sum of variances considering all principal components. For the instance we created, $C = 8$ of 24.

To solve formulations *RIFAM*, *RIFAM*2 and *RRIFAM* we use a master and adversarial problem approach where, at each iteration, we use the adversarial problems to search for a demand scenario instance that invalidates the master problem solution. See [7] for more details on this solution approach.

Algorithms were coded in Julia [13] using JuMP and Cplex 12.7. All algorithms were run in an Intel CORE i7 CPU 3770 machine.

5.2 Comparative performance of the formulations

Table 1 presents the results of the implementation and solution for the four different formulations. The relation between objective values are as expected since formulation *IFAM* is optimizing against a specific demand scenario, formulations *RIFAM* and *RIFAM*2 are optimizing against a worst case scenario and formulation *RRIFAM* is optimizing an expected performance (worst-case). *RIFAM* is designed to be a lower bound of *RIFAM*2 since it considers less constraints (restricted uncertainty set), but the results show that *RIFAM* is a reasonable approximation of *RIFAM*2. Since we use affine decision rules, formulations *RIFAM*, *RIFAM*2 and *RRIFAM* are themselves upper bound approximations of the true optimal worst-case or worst-case expected performance. Since we use auxiliary variables to compose affine decision rules for formulation *RRIFAM*, it leads to more flexible results than affine decision rules use original demand uncertainty.

The total time performance result is in direct link with the number of variables of each formulation, although the number of iterations for each of the robust formulations varies. In terms of time performance, dimensionality reduction has been effective to reduce total time. On the other hand, since the size of our airline instance is small, additional measures should be put in place to be able to deal with real large airline instances.

The simulation results are also as expected since the formulation *RRIFAM*, in the long run, leads to the less costly total solution. We note that there are no guarantees, in terms of the mathematical model proposed, on how formulations *IFAM* and *RIFAM* would perform in the long simulation run. We also note that formulation *RRIFAM* is an approximation of the true optimal result. Even though we would expect that, in the long simulation run, result of worst-case expected performance of formulation *RRIFAM* would out perform the two other formulations, and that is the case.

6 CONCLUSION

Initial computational results have shown that our proposed model can improve over other more traditional approaches. A further study can analyze the quality of the approximations performed, using real life data and comparing with data-driven stochastic optimization approximation algorithms.

REFERENCES

[1] Abara, J. Applying integer linear programming to the fleet assignment problem. *Interfaces 19*, 4 (1989), 20–28.

[2] Barnhart, C., Kniker, T. S., and Lohatepanont, M. Itinerary-based airline fleet assignment. *Transportation Science 36*, 2 (2002), 199–217.

[3] Ben-Hur, A., Horn, D., Siegelmann, H. T., and Vapnik, V. Support vector clustering. *J. Mach. Learn. Res. 2* (2002), 125–137.

[4] Ben-Tal, A., Goryashko, A., Guslitzer, E., and Nemirovski, A. Adjustable robust solutions of uncertain linear programs. *Mathematical Programming 99*, 2 (2004), 351–376.

[5] Ben-Tal, A., and Nemirovski, A. Robust convex optimization. *Math. Oper. Res. 23*, 4 (1998), 769–805.

[6] Ben-Tal, A., and Nemirovski, A. Robust solutions of uncertain linear programs. *Operations Research Letters 25*, 1 (1999), 1–13.

[7] Bertsimas, D., Dunning, I., and Lubin, M. Reformulation versus cutting-planes for robust optimization. *Computational Management Science 13*, 2 (Apr 2016), 195–217.

[8] Bertsimas, D., Sim, M., and Zhang, M. Adaptive distributionally robust optimization. *Management Science* (2018), online.

[9] Boudia, M., Delahaye, T., Gabteni, S., and Acuna-Agost, R. Novel approach to deal with demand volatility on fleet assignment models. *Journal of the Operational Research Society 69*, 6 (2018), 895–904.

[10] Delage, E., and Ye, Y. Distributionally robust optimization under moment uncertainty with application to data-driven problems. *Operations Research 58*, 3 (2010), 595–612.

[11] Hane, C. A., Barnhart, C., Johnson, E. L., Marsten, R. E., Nemhauser, G. L., and Sigismondi, G. The fleet assignment problem: Solving a large-scale integer program. *Math. Program. 70* (1995), 211–232.

[12] Kenan, N., Jebali, A., and Diabat, A. An integrated flight scheduling and fleet assignment problem under uncertainty. *Computers and Operations Research 100* (2018), 333 – 342.

[13] Lubin, M., and Dunning, I. Computing in Operations Research using Julia. *CoRR abs/1312.1431* (2013).

[14] Shang, C., Huang, X., and You, F. Data-driven robust optimization based on kernel learning. *Computers and Chemical Engineering 106* (2017), 464 – 479.

[15] Shang, C., and You, F. Distributionally robust optimization for planning and scheduling under uncertainty. *Computers and Chemical Engineering 110* (2018), 53 – 68.

[16] Sherali, H. D., Bish, E. K., and Zhu, X. Airline fleet assignment concepts, models, and algorithms. *European Journal of Operational Research 172* (2006), 1–30.

[17] WOLD, S., ESBENSEN, K., and GELADI, P. Principal component analysis. *Chemometrics and Intelligent Laboratory Systems 2* (1987), 37–52.

Routing and Slot Allocation in 5G Hard Slicing

Nicolas Huin
Huawei Technologies
nicolas.huin@huawei.com

Jérémie Leguay
Huawei Technologies
jeremie.leguay@huawei.com

Sébastien Martin
Huawei Technologies
sebastien.martin@huawei.com

Paolo Medagliani
Huawei Technologies
paolo.medagliani@huawei.com

Shengming Cai
Huawei Technologies
caishengming@huawei.com

ABSTRACT

5G networks will enable the creation of network slices to serve very different user requirements. Flex Ethernet (FlexE) is a standard technology that provides strict isolation between slices, also called hard slicing, by allocating capacity slots of physical links to slices. The resulting resource allocation problem is called Routing and Slot Allocation problem (RSA). We first prove that this problem is NP-hard and cannot be approximated. Then, we develop two matheuristics to efficiently solve the problem, by leveraging on a combination of Column Generation and Gauss Seidel procedures. The numerical evaluation, carried out by comparing the two matheuristics against a greedy algorithm over a realistic IP-RAN networks, shows an optimality gap smaller than 7%, while reducing the reservation cost by 4% compared to the greedy algorithm.

1 INTRODUCTION

The deployment of next generation 5G networks is paving the road for custom and personalized network services. In particular, due to the improvement in terms of end-to-end network capacity, latency and reliability, it is now possible to envision the decomposition of the physical network into several virtual sub-networks with very different requirements. Each sub-network, also called a *slice*, is independent from each other, and operated by different players, often referred to as *tenants*. The partitioning of network resources aims at guaranteeing that the requirements of tenants are met in all slices.

The importance of network slicing relies on the fact that these virtual networks can be designed to guarantee different Quality of Service (QoS) requirements. In 5G networks [5], three main use cases are commonly identified, namely enhanced Mobile Broad Band (eMBB), ultra Reliable and Low Latency Communications (uRLLC), and Massive IoT (mIoT), using the same physical infrastructure. The resources are provisioned inside each slice in such a way that the SLA (Service Level Agreement) requirements specified for each tenant can be met.

According to the isolation level, we categorize slicing technologies into *soft* and *hard* slicing. In soft slicing [1, 4], despite that QoS performance guarantees are pledged to slices, the traffic is actually multiplexed in a queuing system. A high load on a physical link may introduce an additional latency for all the slices that are routed through that link. And the traffic in one slice may impact the other slices in case of congestion. However, within hard slicing [8], each slice has dedicated resources at both physical and MAC layers. Performance misbehaviors of one slice can not have any influence on the other slices.

The main technology used to provide hard isolation is Flex Ethernet (FlexE) [10]. As mentioned in [1], it is a key enabler of 5G networks. The way FlexE can provide isolation between slices is through the reservation of resources at physical and MAC layers in a Time-Division Multiplexing Access (TDMA) fashion. The capacity of physical ports inside FlexE-enabled devices is allocated to each slice in the form of slots, i.e., multiples of a fundamental unit, normally expressed in Gigabits. Once a slot is allocated to a slice, it cannot be shared with another one. When a slice is created, FlexE slots must be reserved on physical links and user traffic must be steered through these slots. A network controller is typically taking routing and slot allocation decisions with the goal of minimizing unused resources.

In this paper, we present the *Routing and Slot Allocation (RSA)* problem for hard slicing with FlexE in 5G networks where the goal is to minimize the cost of resource reservations for a slice, under the constraint that all services in the slice are accepted. We show that this problem is NP-hard and it cannot be approximated with constant factors unless P = NP. We also present an efficient heuristic to quickly approximate the optimal solution.

The RSA problem is similar to problems such as the *multi-commodity network optimization problems with general step cost functions* [6], or the *energy-aware routing with discrete link rates problem* [2]. However, a few key differences exist. Firstly, the problem studied by [6] considers splittable flows unlike our problems where each service must be routed on a unique path. Secondly, even though [2] consider unsplittable flows, we cannot apply their method due to statistical multiplexing available in IP-RAN networks (see Section 2.3). To the best of our knowledge, we are the first to propose a column generation algorithm to solve this problem.

The structure of the paper is the following. We explain *hard slicing* in Section 2 and formally present the RSA problem in Section 2.3. We then propose an extended formulation of the problem in Section 3 and detail the column generation procedure. We then show in Section 4 two heuristics and compare, in Section 5, our heuristics on realistic 5G scenarios using IP-RAN network. Finally, we conclude this paper and discuss future works in Section 6.

2 HARD SLICING

Hard physical isolation between different slices can be acheived with Flex Ethernet (FlexE). This section presents how the technology works and the Routing and Slot Allocation problem (RSA).

2.1 Flex Ethernet for hard slicing

As shown in Figure 1, the Optical Internetworking Forum (OIF) has designed the FlexE standard as an extension of the traditional IEEE 802.3 standard for wired Ethernet. In more details, FlexE is implemented at the layer 1.5 of the OSI stack, adding a *shim*

ISBN 978-3-89318-079-0 on OpenProceedings, ISSN 2510-7437

DOI: 10.5441/002/inoc.2019.14

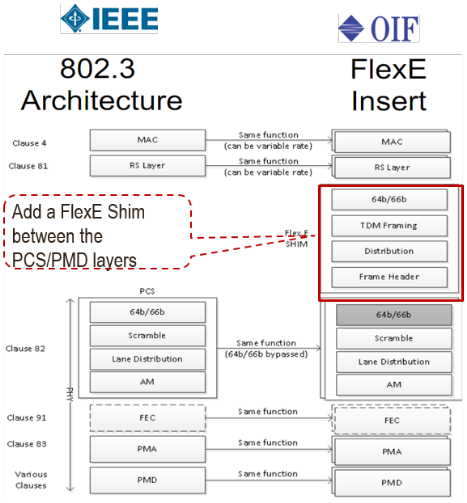


Figure 1: Extension of IEEE 802.3 to support Flex Ethernet.

layer which is in charge of allocating transmission slots to each slice with a fixed calendar, as presented in Figure 2. This rigid mapping forces the bandwidth reserved over a sub-interface to be expressed as a multiple of fundamental slot units. In the FlexE implementation we considered, the bandwidth is reserved in blocks of 5 Gb [10]. However, the first 5 slots allocated to each slice can be of 1 Gb, for a finer bandwidth reservation.

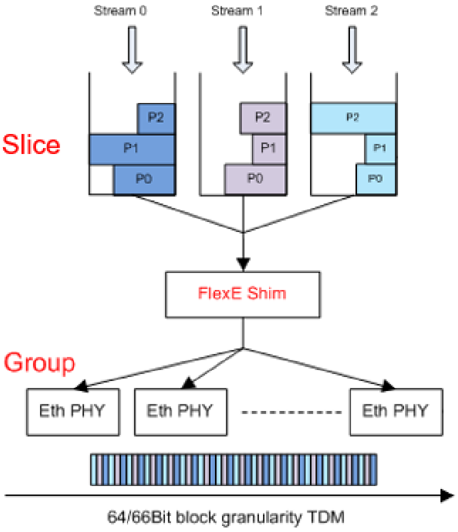


Figure 2: Role of the FlexE shim.

Data packets from the FlexE shim are then mixed on different PHY interfaces that carry all or part of the traffic coming from one or more sub-interfaces. The PHY interfaces are then multiplexing in a TDMA fashion, according to 64/66-bit block data line encoding. This multiplexing operation follows a rigid calendar, which is shared between the transmitter and the receiver to let the latter decodes the data when received.

2.2 Slot allocation policy

FlexE follows three bandwidth reservation rules: (i) it is necessary to activate enough slots on a link to cover all the services of the slice routed through that link; (ii) if there is enough activated slots over a link to accommodate a new service, it is not necessary to activate a new one; (iii) the slot activation sequence comes with a given order. For instance, referring to Figure 3, Service 1 of 7 Gb and Service 2 of 3 Gb need to use the same FlexE link. For Service 1, it is necessary to activate the first 5 1-Gb slots and 1 5-Gb slot, for a total of 10 Gb. This means that there are 3 Gb that are activated and not used by Service 1 and that can be used "for free" by Service 2. In particular, according to the FlexE standard, it is not possible to activate a 5-Gb slot before having activated all the 1-Gb slots. Thus, only the following link configurations are allowed: 1 Gb, 2 Gb, 3 Gb, 4 Gb, 5 Gb, 10 Gb, 15 Gb, 20 Gb and other multiples of 5 Gb slots.

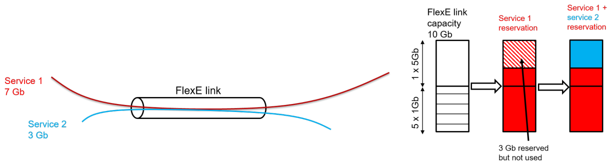

Service 1
7 Gb
Service 2
3 Gb
FlexE link
FlexE link capacity 10 Gb
1 x 5Gb
5 x 1Gb
Service 1 reservation
Service 1 + service 2 reservation
3 Gb reserved but not used

Figure 3: Scheme of FlexE link utilization

On top of the three rules mentioned above, there is another bandwidth reservation policy that must be followed in IP-RAN networks (IP networks for mobile radio networks in 4G or 5G). Figure 4 shows that in aggregation and core networks statistical multiplexing can be used to save resources. The main idea of statistical multiplexing is to assume all services crossing a link will not be active at the same time. Therefore, it is possible to reserve only a portion of the bandwidth required by the services. However, it is necessary to reserve enough bandwidth to ensure that (i) the scaled sum of the capacities of the services passes and (ii) each service alone can pass. The scaling factor applied in the aggregation and in the core network is different as it depends on the number of services using the network. This mechanism is referred to as *Convergence Ratio* (CR). The CR scaling factor can be applied only to services that explicitly support it. For example, if two services, requesting for 4 Gb each, are routed on a link with a convergence ratio of 2, 4 Gb must be allocated as to allow each service to be routed alone.

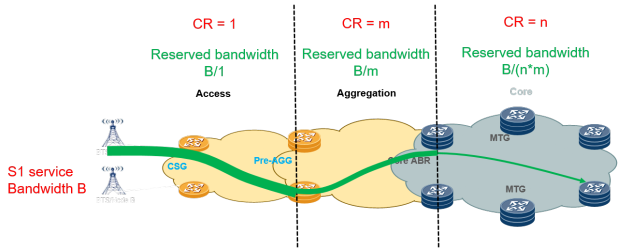


Figure 4: Convergence ratio (CR) in IP-RAN networks.

2.3 Routing and Slot Allocation problem

Let $G = (V, E)$ be the graph representing the network, where V is the set of nodes associated with the routers and E is the set of links between the routers. For each link $e \in E$ we consider a positive cost C_e per unit of bandwidth used, a capacity b_e and, a latency λ_e. In particular, b_e can be expressed as a multiple of a basic unit, referred to as slot, whose size is defined by the FlexE standard. For each link, it is possible to define a set of valid slot configurations S^e that enumerates the possible slot activations. To each link configuration $s \in S^e$ corresponds a bandwidth utilization ξ_{es}.

A slice consists of a set of demands K to be allocated in the network. Each demand $k \in K$ is characterized by a source node $s^k \in V$, a destination node $t^k \in V$, a bandwidth requirement D_k and a latency bound Λ_k. As we are considering an IP-RAN network, statistical multiplexing applies in some parts of the network for the subset $K_{\mathsf{C}} \subseteq K$ of the demands. For each link e, we define a CR factor μ^e and the amount of bandwidth used by a demand $k \in K_{\mathsf{C}}$ is given by $D_k^e = \mu^e D_k$. However, the bandwidth allocation of a link with statistical multiplexing must ensure that each demand in K_{C} can be routed alongside the demand without convergence ratio. Thus, the bandwidth usage of a link e to allocate the set of demands K_e, it is given by

$$u(e, K_e) = \sum_{k \in K_e \cap K_{\mathsf{NC}}} D_k + \max\left(\sum_{k \in K_e \cap K_{\mathsf{C}}} D_k^e, \max_{k \in K_e \cap K_{\mathsf{C}}} D_k \right) \tag{1}$$

where $K_{\mathsf{NC}} \subseteq K$ is the set of demands that are not requesting for statistical multiplexing.

The Routing and Slot Allocation problem (RSA) consists in *computing a feasible path for all demands within the slice, while respecting the link capacities and delay constraints and minimizing the cost of resource reservation in the network.*

3 COLUMN GENERATION MODEL

In this section, we first formulate the problem via an Integer Linear Program (ILP). As this model requires an exponential number of variables, we propose a pricing procedure, based on Column Generation (CG) techniques to dynamically add the necessary variables.

3.1 Problem formulation

For each demand $k \in K$, we denote by P^k the set of all possible paths between source s^k and destination t^k. The number of paths for each demand can be exponential. For each demand k and each path $p \in P^k$, a binary variable x_{kp} is equal to 1 if the path p is used by demand k, 0 otherwise. For this extended model we also consider the slot configuration variables y_{es} for each $e \in E$ and $s \in S^e$ defined in the previous section.

The following ILP *FlexE-CG* is a valid formulation for the FlexE problem.

$$\min \sum_{e \in E} C_e \sum_{s \in S^e} \xi_{es} y_{es} \tag{2a}$$

$$\text{s.t} \sum_{k \in K_{\mathsf{NC}}} \sum_{p \in P^k : e \in p} D_k x_{kp} + \sum_{k \in K_{\mathsf{C}}} \sum_{p \in P^k : e \in p} \mu^e D_k x_{pk} \leq \sum_{s \in S^e} \xi_{es} y_{es} \quad \forall e \in E \tag{2b}$$

$$\sum_{k' \in K_{\mathsf{NC}}} \sum_{p \in P^{k'} : e \in p} D_{k'} x_{pk'} + \sum_{p \in P^k : e \in p} D_k x_{kp} \leq \sum_{s \in S^e} \xi_{es} y_{es} \quad \forall e \in E, k \in K_{\mathsf{C}} \tag{2c}$$

$$\sum_{p \in P^k} x_{pk} \geq 1 \quad \forall k \in K \tag{2d}$$

$$\sum_{s \in S} y_{es} \leq 1 \quad \forall e \in E \tag{2e}$$

$$x_{pk} \in \{0, 1\} \quad \forall k \in K, p \in P^k \tag{2f}$$

$$y_{es} \in \{0, 1\} \quad \forall e \in E, s \in S^e \tag{2g}$$

The inequalities (2b) are the traditional capacity constraints on each link e where the amount of traffic for demands in K_{NC} and demands scaled with the convergence ratio in K_{C} must be smaller or equal than the size of the activated slot ξ_{es}. Inequalities (2c) ensure that each demand in K_{C} can be routed on its own. Inequalities (2d) ensure that at least one path is assigned to one demand. Inequalities (2e) guarantee that only one slot configuration is activate on each link. Remark that, as we aim at minimizing costs which are positive, it is useless to take two paths for each demand. In order to help the pricing procedure, we do not consider strict equality in (2e). The inequalities (2f) and (2g) are the integrality constraints.

Pricing procedure. Since the model *FlexE-CG* has an exponential number of variables, it is necessary to propose a pricing procedure to generate only the necessary columns (i.e., to activate variables) inside the CG algorithm. Indeed, the pricing procedure is a sub module of the CG algorithm allowing to generate only the necessary columns that improve the linear relaxation of the *FlexE-CG* model and allow to reach the optimal relaxed solution. The pricing procedure consists in solving a sub problem to define if there exists a column such that the associated constraint in the dual formulation is violated([3]).

At each step of the column generation algorithm, we obtain the optimal dual values $\delta^* \in \mathbb{R}_+^E$, $\pi^* \in \mathbb{R}_+^{E \cdot K_{\mathsf{C}}}$, $\gamma^* \in \mathbb{R}_+^K$, $\theta^* \in \mathbb{R}_+^E$ associated with the inequalities (2b), (2c), (2d), (2e), respectively. Thus, for a given demand $k \in K$, the separation of a violated dual constraint is equivalent to finding a path p such that

$$-\sum_{e \in p} D_k^e \delta_e - \sum_{e \in p} D_k \pi_e^k + \gamma_k > 0 \tag{3}$$

if $k \in K_{\mathsf{C}}$, and the following if $k \in K_{\mathsf{NC}}$:

$$-\sum_{e \in p} D_k \delta_e - \sum_{e \in p} \sum_{k' \in K_{\mathsf{C}}} D_k \pi_e^{k'} + \gamma_k > 0 \tag{4}$$

For each demand $k \in K_{\mathsf{C}}$ (resp. $k \in K_{\mathsf{NC}}$), the constrained shortest path where the cost on each link is $e \in E$ by $D_k^e \delta_e + D_k \pi_e^k$ (resp. $D_k \delta_e + \sum_{k' \in K_{\mathsf{C}}} D_k \pi_e^{k'}$) solves the pricing procedure. If solved optimally, it guarantees that a path is found if it exists. If the cost of the shortest path is strictly smaller than γ_k, then we add the column (variable) associated with this path and this demand to the problem. If for all demands, no columns are added, the column generation procedure terminates.

Note that additive end-to-end QoS constraints, such as delay, jitter or packet loss (taking the logarithm), can be integrated in the path computation procedure. In our heuristic algorithm, we use well-known algorithms such as LARAC [9] or GEN-LARAC [11] to solve the constrained shortest path problem.

3.2 Column generation algorithm

As mentioned in Section 3.1, the *FlexE-CG* formulation contains an exponential number of variables and is adapted to a column generation algorithm to solve its relaxation. Figure 5 depicts the whole procedure where the fractional solution is then fixed to integer using a rounding algorithm. Column generation relies on a pricing problem to generate variables on-the-fly instead of enumerating them in the master problem. We combine it with a constraint generation procedure for constraints (2c) to avoid any stability issue and improve convergence speed.

The algorithm works as follows: first, we warm-start the *FlexE-CG* model with a solution found using a greedy algorithm (see

Algorithm 2 for more details). Then, we proceed with the following steps:

1) The column generation alternates between solving the master problem and the pricing problems:
   a) We solve a reduced *FlexE-CG*, i.e., *FlexE-CG* with a subset of paths, using a linear solver.
   b) Using the dual values of *FlexE-CG*, for each demand, we look for constrained paths that violate (3) or (4), using the LARAC algorithm. If we find any, we add them to *FlexE-CG* and go back to step 1a).
2) We then search for any violated multiplexing constraint (2c). If none is violated, we have an optimal solution z^*_{LP} for the relaxation, otherwise we go back to step 1.

Algorithm 1 Randomized rounding

Input: A network $G = (V, E)$, link capacity b_e, $\forall e \in E$, set of demands K, set of paths $\bar{P} = \bigcup_{k \in K} \bar{P}_k$, vector $x \in \mathbb{R}^{|P|}$ of value for each path
Output: Set of paths P

```
$p^*_k \leftarrow \emptyset, \forall k \in K$
$K_e \leftarrow \emptyset \quad \forall e \in E$
for demand $k \in K$ do
    while $\bar{P}_k \neq \emptyset$ do
        $\tilde{p} \leftarrow$ path drawn at random from $\bar{P}_k$ with $Pr(p \text{ is selected}) = \frac{x_p}{\sum_{p \in \bar{P}_k} x_p} \quad \forall p \in \bar{P}_k$
        if $\forall$link $e \in \tilde{p} : u(e, K_e \cup \{k\}) \leq b_e$ then
            $p^*_k \leftarrow \tilde{p}$
            for $\forall$link $e \in \tilde{p}$ do
                $K_e \leftarrow K_e \cup \{k\}$
            break
        else
            $x_p \leftarrow 0$
            $P^k \leftarrow P^k \setminus \tilde{p}$
$K_{\mathrm{REJ}} \leftarrow \{k \in K : p^*_k = \emptyset\}$    ▷ Get set of demands not routed
return $\bigcup_{k \in K} \{p^*_k\} \cup Greedy(G, K_{\mathrm{REJ}}, K_e, b)$
```

Randomized Rounding. Since the column generation procedure only provides a relaxed solution for *FlexE-CG*, we need to derive an integral solution from it. We propose a randomized rounding algorithm, shown in Algorithm 1. For each demand, we randomly (with uniform distribution) choose a path amongst all the paths generated during the column generation procedure. The probability of choosing a path p is given by

$$P(k \text{ is routed on } p) = x^*_{pk}$$

where x^*_{pk} is the value of x_{pk} in the optimal solution of the relaxation of *FlexE-CG*. We check that the selected path can be routed on the current network configuration. If this is the case, we update the link-slot allocation and move to the next demand. Otherwise, we remove the path from the set of possible paths and pick a new one at random. If there is no more path in the pool, we add the demand to the list of *rejected demands*. Once all demands are considered and if the list of *rejected demands* is not empty, we try to find a solution for the rejected demands with the greedy algorithm.

Parallelization. As depicted below in Figure 5, the master problem and the pricing problems are solved iteratively but columns in the pricing can be generated in parallel. In the rounding step, we run in parallel several randomized rounding routines to ensure that the final solution will be integer and feasible. Finally, the best solution among those provided in the rounding step is selected.

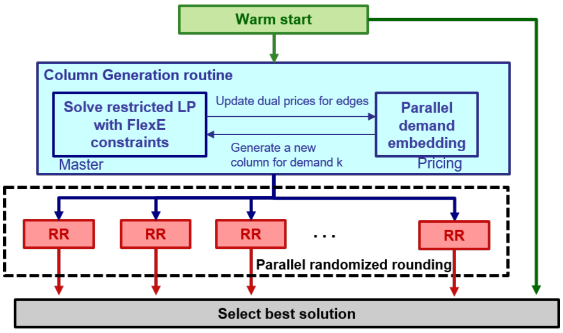

Warm start
Column Generation routine
Solve restricted LP with FlexE constraints
Update dual prices for edges
Parallel demand embedding
Generate a new column for demand k
Master
Pricing
RR
RR
RR
...
RR
Parallel randomized rounding
Select best solution

Figure 5: Algorithmic framework to solve *FlexE-CG*.

4 HEURISTICS

In this section, we present two heuristics we designed to solve the RSA problem. The first one is a simple greedy algorithm that we use as benchmark. The second one is an adaptation of a procedure from the literature [7] to solve a network planning problem with splittable flows and no considerations on statistical multiplexing.

4.1 Greedy algorithm

Algorithm 2 Greedy algorithm

Input: A network $G = (V, E)$, link capacity b_e, $\forall e \in E$, set of demands K to route, set of demands K_e on each link e
Output: Set of paths P

```
$P \leftarrow \emptyset$
$K_e \leftarrow \emptyset, \forall e \in E$
for demand $k \in K$ do
    $E^k \leftarrow \{e : u(e, K_e \cup \{k\}) \leq b_e\}$
    $w^k(e) = \begin{cases} 1 \text{ if } A(u(e, K_e)) \geq u(e, K_e \cup \{k\}) \\ 1 + C_e \text{ otherwise} \end{cases} \quad \forall e \in E^k$
    Build weighted graph $G^k = (V, E^k, w^k)$
    Find shortest path $p$ from $s^k$ to $t^k$ in $G^k$
    $P \leftarrow P \cup \{p\}$
    for link $e \in p$ do
        $K_e \leftarrow K_e \cup \{k\}$
return $P$
```

Algorithm 2 is a greedy algorithm that selects a path, for each demand, by solving a constrained shortest path problem and update the slot allocation accordingly.

For each $k \in K$, we build a weighted graph $G^k = (V, E^k, w^k)$ and search for a constrained shortest path from s^k to t^k on G^k using the LARAC algorithm [9].

The weights w^k_e are chosen in order to favor paths that do not need a bigger slot allocation to route k and is given by

$$w^k(e) = \begin{cases} 1 \text{ if } A(u(e, K_e)) \geq u(e, K_e \cup \{k\}) \\ 1 + C_e \text{ otherwise.} \end{cases}$$

where K_e is the set of demands on e and $A(x)$ returns the minimum bandwidth allocation needed to route x units of bandwidth.

We also filter out links that do not have enough capacity to route demand k. Once a path p is found, we update the sets K_e for each link on p and move on the next demand.

4.2 Gauss-Seidel algorithm

Algorithm 3 Gauss-Seidel algorithm

```
Input: A network $G = (V, E)$, link capacity $b_e \forall e \in E$, set of demands K, set of paths $P = \{p_k : \forall k \in K\}$
Output: Set of paths $P$
$E_{\mathsf{CAND}} \leftarrow E$
while $E_{\mathsf{CAND}} \neq \emptyset$ do
    $K_e \leftarrow \{k : e \in p_k\} \quad \forall e \in E$
    $\tilde{e} \leftarrow \arg\max_{e \in E_{\mathsf{CAND}}} C_e \times (S(u(e, K_e)) - u(e, K_e))$
    $E_{\mathsf{CAND}} \leftarrow E_{\mathsf{CAND}} \setminus \tilde{e}$
    $(P_{\mathsf{OLD}}, K_{\mathsf{OLD}}, b_{\mathsf{OLD}}) \leftarrow (P, K_{\tilde{e}}, b_{\tilde{e}})$
    $b_{\tilde{e}} \leftarrow \lfloor S(u(\tilde{e}, K_{\tilde{e}})) \rfloor$
    for demand $k \in K_{\mathsf{OLD}}$ do
        for link $e \in p_k$ do
            $K_e \leftarrow K_e \setminus k$
        $p_k \leftarrow \emptyset$
    $P_{\mathsf{NEW}} \leftarrow P \cup Greedy(G, K_{\mathsf{OLD}}, K_e, b)$
    if $Cost(P_{\mathsf{NEW}}) < Cost(P_{\mathsf{OLD}})$ then
        $P \leftarrow P_{\mathsf{NEW}}$
    else
        $P \leftarrow P_{\mathsf{OLD}}$
        Restore $(P_{\mathsf{OLD}}, K_{\mathsf{OLD}}, b_{\mathsf{OLD}})$ as current solution
return P
```

Finally, we present a Gauss-Seidel procedure in Algorithm 3 that aims at improving any existing solution, similar to the *link-rerouting* algorithm in [7]. It is a local search heuristic which tries to reduce the number of active slots of each link by rerouting demands on new paths.

More precisely, for a valid solution, the algorithm chooses the link $\tilde{e}$ with the most *free* bandwidth on it, weighted by its cost, i.e,

$$\arg \max_{e \in E_{\mathsf{CAND}}} C_e \times (A(u(e, K_e)) - u(e, K_e))$$

where E_{CAND} is the set of links not yet considered for removal. We remove all demands using e from the network and reduce the number of slot on e by one, e.g., if e was a FlexE link with a reservation of 15G, we reduce it to 10G. We then greedily route the removed demands on the new network configuration. If we obtain a lower cost with the new routing, we use it as our new best solution. Otherwise, we restore the link to its previous slot configuration, restore the removed demands on their previous paths. We continue until all links have been considered.

5 NUMERICAL RESULTS

In this section, we present numerical results to compare the algorithms on an IP-RAN scenario. The compact formulation (not presented in this paper) and the *FlexE-CG* model have been solved using CPLEX 12.7 and all algorithms have been executed on a server with 4 Intel(R) Xeon(R) CPU E5-4627 v2 @ 3.30GHz and 504GB of RAM.

5.1 IP-RAN scenario

We generate instances of an IP-RAN network with multiple domains connected to a mesh network. Each domain is composed of a set of nodes connected in single or dual-homing (access network) to a ring with probabilistic shortcuts (aggregation network). Services in slices can exists between nodes in the access networks or between a node in an access network and a node in the core network. The bandwidth requirement of services is randomly chosen between 50 Mb and 1 Gb. We consider two types of scenarios: hard slicing, denoted FlexE, and soft slicing, denoted VLAN (Virtual LAN), a candidate technology for this scenario. For VLAN, we assume that the granularity of each slot is 1 Mb, which is negligible compared to the size of the smallest demand. All algorithms are executed with a time limit of one hour. Results are averaged over 5 trials. A summary of the parameters used in the experiments is shown in Table 1.

Topology type	Instance name	# Nodes	# Edges	# Demands
Small	VLAN|FlexE$_{50}$	50	60	60
Middle	VLAN|FlexE$_{1250}$	1250	1600	300
Large	VLAN|FlexE$_{5000}$	5000	6000	600

Table 1: Number of nodes, edges and demands for each instance type

Lower bounds: Figure 6 shows the lower bounds obtained with the compact formulation and the optimal solution z^*_{LP} of the relaxation of *FlexE-CG*. The compact formulation can provide the optimal solution for small instances; we can thus evaluate the quality of the bounds computed by *FlexE-CG*. On small VLAN scenarios (i.e., with 50 demands), the bounds provided by *FlexE-CG* are close to the optimal (less than 3%); however they are larger for the hard slicing scenarios (around 22%) as the bigger granularity of FlexE worsens the relaxation of the objective function.

On middle size instances (i.e., with 1250 demands), the compact formulation cannot be solved to optimality within the one-hour limit. Moreover, the bounds computed is much smaller than the ones computed by *FlexE-CG*. Thus, we use the bounds of *FlexE-CG* to evaluate the solutions of our algorithms on middle and large instances.

Solution quality: In Figure 7, we compare the solutions of the greedy and *FlexE-CG* algorithms, improved by the Gauss-Seidel algorithm, in terms of gap to the best lower bound, i.e., the compact solution for small instances and the *FlexE-CG* bounds for middle and large instances. The gap is computed as $(z_{\mathsf{SOL}} - \mathsf{LB})/\mathsf{LB}$, where z_{SOL} is the value of the solution and LB is the best known lower bound of the instance. Solutions of the compact formulation provided by CPLEX are not shown as CPLEX cannot return a valid solution in one hour.

First, we can see that the greedy provides good solutions, whose gap is 10.5% in the worst case. *FlexE-CG* can further improve the solution provided by the greedy algorithm and, on average, the gap is reduced by 3.8%. Moreover, *FlexE-CG* solutions are close to the optimal on soft slicing scenarios, with a gap smaller than 1.4%. The gap of hard slicing scenarios is larger, up to 6.2%, on middle size instances. However, the gap to optimality might be smaller as the bounds for hard slicing are not as tight as the ones for soft slicing.

Computational time: Finally, in Figure 8, we compare the computational time of the algorithms. The compact formulation is quite slow to be solved compared to the other algorithms. While it takes up to 36s, on average to solve small instances, *FlexE-CG* finds a solution in less than 2 s and the greedy algorithms takes

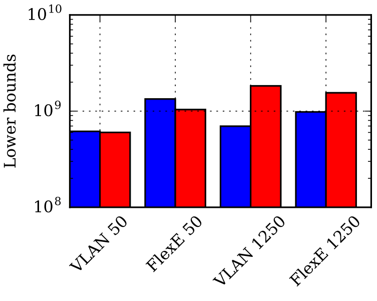


Figure 6: Average lower bounds obtained with the compact formulation (optimal for small networks) and *FlexE-CG* (1h timeout).

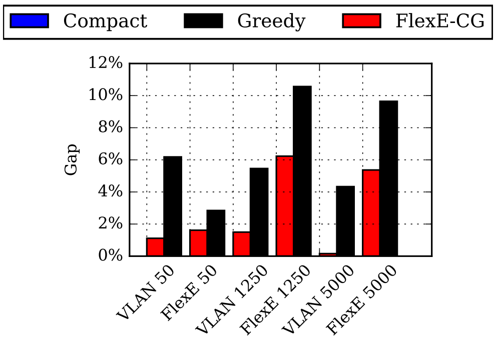


Figure 7: Average gap of each solutions for the greedy and *FlexE-CG*.

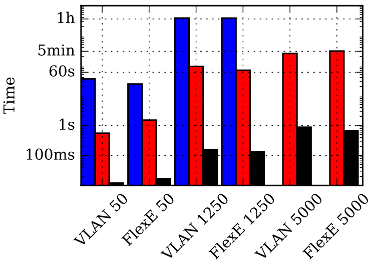


Figure 8: Average computation times of each algorithms.

less than 20 ms. As previously mentioned, the compact formulation exceeds the time budget on middle instances. The greedy remains efficient as it takes less than one second even for large instances. *FlexE-CG*, instead, is considerably slower than greedy for middle and large scale networks, but it provides for better results.

Given the different performance in terms of optimality gap and execution time of the two approaches, they could be used in parallel to efficiently solve the RSA problem. The greedy algorithm can be used to quickly accept demands in an online fashion, while *FlexE-CG* can be used to periodically reconfigure the network and minimize the total resource reservation cost.

6 CONCLUSION

In this paper, we presented the Routing and Slot Allocation problem for 5G hard slicing. We modeled the problem using mathematical programming and proposed an extended formulation, solved using column generation. We analyzed its strength against a basic integer linear formulation. Based on this extended formulation, we derived a matheuristic, referred to as *FlexE-CG*, that we benchmarked against a greedy algorithm. We also strengthened our matheuristic through an adaptation of the Gauss-Seidel procedure allowing to improve the performances of the two heuristics. We showed that the extended formulation can provide good dual bounds in a reasonable amount of time compared the the compact formulation. The derived heuristic manages to obtain an optimality gap smaller than 7%, while improving the cost value of the solutions provided by the greedy up to 4%. In future works, we will propose valid inequalities to reduce the computational time of our matheuristic and increase the dual bound. Furthermore, we will investigate on others matheuristics and exact method based on our extended formulation.

REFERENCES

[1] 5G Service-Guaranteed Network Slicing White Paper. Huawei whitepaper, February 2017.

[2] Mohamad Khattar Awad, Yousef Rafique, and Rym A. M'Hallah. Energy-aware routing for software-defined networks with discrete link rates: A benders decomposition-based heuristic approach. *Sustainable Computing: Informatics and Systems*, 13:31 – 41, 2017.

[3] V. Chvatal. *Linear Programming*. Freeman, USA, 1983.

[4] A. Destounis, G. Paschos, S. Paris, J. Leguay, L. Gkatzikis, S. Vassilaras, M. Leconte, and P. Medagliani. Slice-based column generation for network slicing. In *IEEE INFOCOM 2018 - Poster*, April 2018.

[5] X. Foukas, G. Patounas, A. Elmokashfi, and M. K. Marina. Network slicing in 5g: Survey and challenges. *IEEE Communications Magazine*, May 2017.

[6] V. Gabrel, A. Knippel, and M. Minoux. Exact solution of multicommodity network optimization problems with general step cost functions. *Operations Research Letters*, 25(1):15 – 23, 1999.

[7] Virginie Gabrel, Arnaud Knippel, and Michel Minoux. A comparison of heuristics for the discrete cost multicommodity network optimization problem. *Journal of Heuristics*, 9(5):429–445, Nov 2003.

[8] Liang Geng, Jie Dong, Stewart Bryant, Kiran Makhijani, Alex Galis, Xavier de Foy, and Slawomir Kuklinski. Network Slicing Architecture. Internet-Draft draft-geng-netslices-architecture-02, Internet Engineering Task Force, July 2017. Work in Progress.

[9] A. Juttner, B. Szviatovski, I. Mecs, and Z. Rajko. Lagrange relaxation based method for the qos routing problem. In *Proceedings IEEE INFOCOM 2001. Conference on Computer Communications. Twentieth Annual Joint Conference of the IEEE Computer and Communications Society (Cat. No.01CH37213)*, volume 2, pages 859–868, April 2001.

[10] OIF. Flex Ethernet 2.0 Implementation Agreement, June 2018.

[11] Ying Xiao, Krishnaiyan Thulasiraman, and Guoliang Xue. Gen-larac: A generalized approach to the constrained shortest path problem under multiple additive constraints. In *Algorithms and Computation*, 2005.

MILP approaches to practical real-time train scheduling: the Iron Ore Line case

Lukas Bach
SINTEF
Oslo, Norway
lukas.bach@sintef.no

Carlo Mannino
SINTEF
Oslo, Norway
carlo.mannino@sintef.no

Giorgio Sartor
SINTEF
Oslo, Norway
giorgio.sartor@sintef.no

ABSTRACT

Real-time train scheduling is a complex network optimization problem, which is receiving increased attention from scientists and practitioners. Despite a vast literature on optimization algorithms for train dispatching, there are very few examples of real-life implementations of such algorithms. Indeed, the transition from theory to practice poses several critical issues, and many simplifying assumptions must be dropped. MILP models become more involved and hard to solve in the short time available. Here we describe how we successfully tackled these issues for dispatching trains on a railway in the north of Norway and Sweden.

1 INTRODUCTION

Railway infrastructure is increasingly congested: passenger traffic is expected to grow by 3.2% yearly for the next 8 years, while freight traffic by 1.4% ([13]). Increasing pressure results in poorer punctuality. In principle, one could augment capacity by building more infrastructure, but this requires large investments and the benefits will only be available after some years. A quicker and cheaper way to increase capacity is to improve traffic management by network optimization. Several recent studies have shown improvements in the punctuality ranging from 10% to 100% [3, 6, 9, 10, 12]. In these and all other papers presented in a large literature (for recent reviews, see [4, 8]), the train scheduling problem is represented by means of *event graphs*. The seminal example is probably Balas' disjunctive graph introduced in [1] (where each node represents the starting of an operation), later extended to cope with blocking, no-wait job-shop scheduling problems by Mascis and Pacciarelli [11].

Despite this huge body of academic studies and successful stories, there have been only a few implementations of real-time train scheduling algorithms in real life [2, 9, 10]. Things are rapidly changing now, thanks to an increased interest by infrastructure managers worldwide in automatic train traffic control systems capable of maximizing punctuality or average velocity[1]. In this paper, we describe one such implementation, focusing on the modelling and algorithmic challenges we had to tackle when moving from theory to practice. First, standard simplifying assumptions must be discarded in order to produce solutions which are practically viable. Next, new solution approaches must be developed and implemented in order keep the computation time of the optimal solution in the range of few seconds. Indeed, Fischetti and Monaci [5] showed that state-of-the-art solvers are already unable to tackle rather small instances of the MILP models derived from the event graphs of these type of scheduling problems.

The implementation we discuss in this paper is applied to a critical part of the railway network that runs from Sweden to the coast of Norway, also known as the "Iron Ore Line". This single track line, well within the arctic circle, was originally built to transport iron ore from northern Sweden to the ice free waters of Narvik in northern Norway. The line is also used by a few passenger trains per day. In recent years, the increased number of iron ore trains as well as other types of freight trains has challenged the capacity of the line. An optimized dispatching could help ensuring that the physical capacity is used to its full extent.

One peculiar challenge of this piece of railway comes from its incline. This affects the speed of the heavy trains and may also affect their ability to stop in some of the stations. For example, fully loaded freight trains travelling from the iron ore mines in Sweden to Norway have constraints regarding where they are allowed to stop, while lighter freight trains travelling towards Sweden are allowed higher speed and flexibility. Moreover, some freight trains are also too long for some of the side tracks in certain small stations. Passenger trains may have limitations too. In fact, most of the stations do not have passenger platforms in all their internal tracks, constraining the number of passenger trains that are able to meet in the station. Instead, for all type of trains, another important aspect is the variability of their travel times. Indeed, moving from a stopped condition requires some time to accelerate; similarly, stopping a train requires a deceleration and thus extended running times.

All these practical constraints are usually ignored in theoretical works but they are crucial in real-world applications, requiring more refined models. In this work, we mainly focus on two aspects: a) being able to define a more diverse set of constraints within each station; b) model travel times of each train based on its stopping pattern.

Our starting point is the recent Benders' like decomposition approach to train rescheduling presented in [7, 9], extended to cope with all new physical and logical constraints. In this paper, we discuss the new features and the decomposition approach for this MILP problem. Furthermore we describe the actual implementation which has been tested by dispatchers on the iron-ore line.

2 A MILP FORMULATION

We start our description by considering a slightly simplified version of our problem, where the travel times of trains are fixed and do not depend on whether they stop.

The Iron Ore Line consists of a sequence of small stations and single-tracks. We follow here the micro-macro decomposition approach proposed in [7]. The macro problem is associated with the railway line, considered as a sequence of capacitated resources,

[1] In [2], a large North-American railway company claims that a 1% increase in average velocity of their freight trains brings to the company $200 million savings.

ISBN 978-3-89318-079-0 on OpenProceedings, ISSN 2510-7437

DOI: 10.5441/002/inoc.2019.15

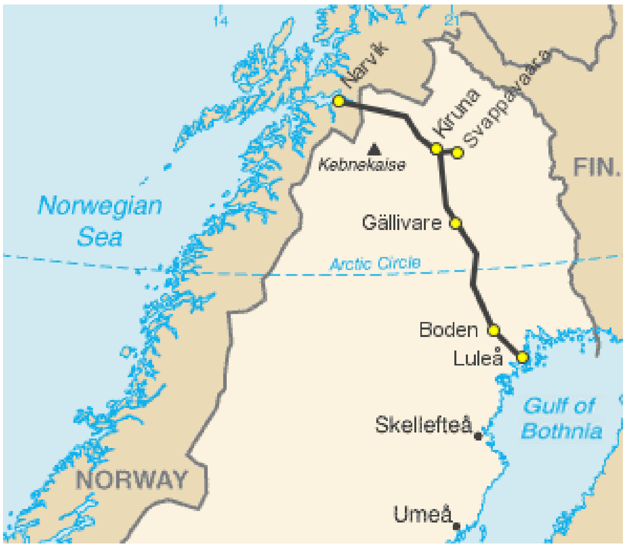


Figure 1: Iron Ore Line

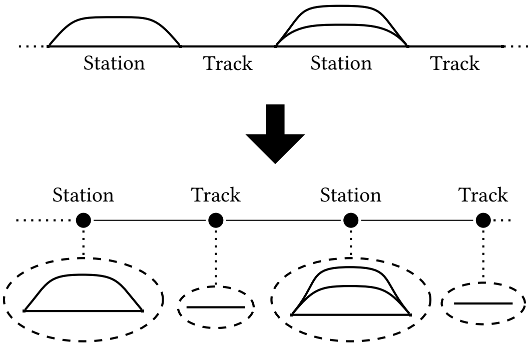


Figure 2: Line decomposition

alternating stations and tracks (see Figure 2). Observe that at this macro level, we avoid the detailed description of the movements (i.e. routing and scheduling) of each train in every station which is instead represented by the expected total time spent in the station. The micro problem is associated with the routing and scheduling of trains within each station and track, according to the arrival and departure times established by the macro problem. The decomposition allows us to treat the constraints generated in the micro level (e.g., track assignment, capacity) independently of each other. In Section 4 we will see how this decomposition can be exploited to solve the MILP model in a master-slave fashion. Instead, in this section we focus on describing how to formulate the constraints arising both at the macro and micro level, describing train movements.

For the line (or macro) problem, each train $a \in A$ is assigned a route, i.e., an ordered sequence $n_a^{r_1}, n_a^{r_2}, \ldots, n_a^{r_q}$ of *route nodes*, where $r_i \in R, i = 1, \ldots, q$ is a line resource, either a track or a station[2], and r_1, r_q are the origin and destination station, respectively. In this aggregation scheme, between each pair of nodes that represent two adjacent stations, there is always a track node.

Let N be the set of all route nodes for all trains in A, $N^O \subset N$ be the set of all nodes associated with origin stations, and $N^D \subset N$ the set of all nodes associated with destinations. We associate a scheduling variable $t_a^r \in \mathbb{R}$ with each route node $n_a^r \in N$, representing the time train a enters the resource r. There is also a fictitious variable $t^o \in \mathbb{R}$, which serves as a reference time for all trains (typically, but not necessarily, we have $t^o = 0$). Thus, we have

$$t_a^r - t^o \geq \Gamma_a, \quad n_a^r \in N^O, \tag{1}$$

where Γ_a is the earliest time train a can enter the network. Now let $n_a^r, n_a^{r+1} \in N$ be two consecutive route nodes in a particular train route. Note that the time a train exits a resource is precisely the time the train enters the subsequent resource in its route. Therefore, the following constraints hold:

$$t_a^{r+1} - t_a^r \geq \Lambda_a^r, \quad n_a^r \in N \setminus N^D, \tag{2}$$

where Λ_a^r is the minimum time it takes train a to traverse resource r. Moreover, for the destination nodes we have:

$$t_a^{\text{out}} - t_a^r \geq \Lambda_a^r, \quad n_a^r \in N^D, \tag{3}$$

where the fictitious t_a^{out}, $a \in A$, represents the time train a “leaves” the railway network, i.e., it concludes its journey at the arrival station. Constraints (1), (2), and (3) are usually called *precedence constraints*, and they model the *free running* of a train, i.e., the minimum time required by a train to travel along its route without obstacles from other trains. Incidentally, even if we will not make explicit use of the underlying event graph, it is worth mentioning here that this is built by associating a node with every time variable and a directed edge with every constraint (1), (2), and (3).

In general, one has to consider the interactions between trains travelling in the same network. Observe that, for a pair of distinct trains a, b traversing a resource r, exactly one of the following three conditions must occur:

(1) train a and b meet in resource r
(2) train a traverses resource r before train b
(3) train b traverses resource r before train a

Consider now a set of distinct trains $A(r) \subseteq A$ traversing a resource r. For each ordered pair of distinct trains $(a, b) \in A(r) \times A(r)$, we define y_{ab}^r to be equal to 1 if a exits r before b enters, and 0 otherwise. Furthermore, for each pair of trains $\{a, b\} \subseteq A(r)$, we introduce the binary variable x_{ab}^r, which is 1 if and only if a and b are simultaneously (i.e., they *meet*) in resource r. Then, we have that

$$y_{ba}^r + y_{ab}^r + x_{ab}^r = 1, \quad \{a, b\} \subseteq A(r), r \in R. \tag{4}$$

Accordingly, for every $\{a, b\} \subseteq A(r), r \in R$, the schedule t will satisfy a family of (indicator) *disjunctive constraints* as follows[3]:

$$\begin{array}{llll} (i) & y_{ab}^r = 1 & \Longrightarrow & t_b^r - t_a^{r+1} \geq 0, \\ (ii) & y_{ba}^r = 1 & \Longrightarrow & t_a^r - t_b^{r+1} \geq 0, \\ (iii) & x_{ab}^r = 1 & \Longrightarrow & \left\{ \begin{array}{l} t_b^{r+1} - t_a^r \geq 0 \\ t_a^{r+1} - t_b^r \geq 0 \end{array} \right. , \\ & y_{ab}^r, y_{ba}^r, x_{ab}^r \in \{0, 1\}. & & \end{array} \tag{5}$$

Indeed, $y_{ab}^r = 1$ implies that a exits r before b enters r and, similarly, $y_{ba}^r = 1$ implies that b exits r before a enters. On the other hand, when $x_{ab}^r = 1$, then both a and b exit the sector r after the other train enters it (i.e., they meet in r). Exploiting the

[2] Other decomposition schemes are possible, for instance by collapsing entire railway region in a single node of our master line problem.

[3] Constraints (5) are associated with special entities of the event graph called *disjunctive (alternative) edges*, see for instance [11].

big-M trick, the family of disjunctive constraints in (5) can be easily linearized as follows:

$$
\begin{array}{ll}
(i) & t_b^r - t_a^{r+1} \geq -M(1 - y_{ab}^r), \\
(ii) & t_a^r - t_b^{r+1} \geq -M(1 - y_{ba}^r), \\
(iii) & t_b^{r+1} - t_a^r \geq -M(1 - x_{ab}^r), \\
(iv) & t_a^{r+1} - t_b^r \geq -M(1 - x_{ab}^r), \\
 & y_{ab}^r, y_{ba}^r, x_{ab}^r \in \{0, 1\},
\end{array}
\qquad (6)
$$

A final set of constraints in the macro program will be used to represent the infeasibility of the micro problems, which in turn is associated to the resources in which the railway is decomposed.

Now, let t^* be a schedule that satisfies constraints (1), (2), (3), and (6), and suppose t^* minimizes a given objective function $c(t)$.

If the timetable t^* is feasible for every micro problem (i.e., for every station and every track section between successive stations), then it is feasible and optimal also for the overall problem. Otherwise, at least for one station or one track section, the time schedule decided by the macro problem cannot be attained. There may be several reasons for such infeasibility. Here we will describe the case where feasibility depends only on the set of trains simultaneously in the resource. For example, two passenger trains are not able to meet in a station where there are two internal tracks but only one passenger platform, but two freight trains may meet. On a single track no two trains can meet. Two short trains may pass each other in a siding, but not two long trains. Etc.

So, let $r \in R$ be a set of trains $Q \subseteq A$ *minimally infeasible for* r, if the trains in Q cannot meet simultaneously in r, but all proper subsets of trains in Q can meet. We define the set of $\mathcal{A}(r) \subset 2^A$ as the family of minimally infeasible set of trains for r. Clearly, for any $Q \in \mathcal{A}(r)$, at least two trains[4] in Q cannot meet in r. Note that if, according to a solution (t^*, x^*, y^*), all trains in Q meet in r, then we have $\sum_{\{a,b\} \subseteq Q} x_{ab}^{*r} = \binom{|Q|}{2}$ (namely the number of pairwise meetings in r of trains in Q is precisely $\binom{|Q|}{2}$). To prevent this to happen when the set Q is minimally infeasible, we can thus write the constraint:

$$\sum_{\{a,b\} \subseteq Q} x_{ab}^r \leq \binom{|Q|}{2} - 1, \quad Q \in \mathcal{A}(r), r \in R. \qquad (7)$$

In conclusion, a complete MILP formulation can be obtained by considering as objective function $c(t)$ the sum of the arrival times at destination of the trains, subject to constraints (1), (2), and (3) for all routes, and constraints (4), (6), and (7) for all resources $r \in R$ and all the minimally infeasible sets $Q \in \mathcal{A}(r)$ of trains.

The full model can be written as follow:

$$
\begin{array}{ll}
\min \; c(t) & \\
\text{subject to:} & \\
t_a^r - t^o \geq \Gamma_a, & n_a^r \in N^O \\
t_a^{r+1} - t_a^r \geq \Lambda_a^r, & n_a^r \in N \setminus N^D \\
t_a^{\text{out}} - t_a^r \geq \Lambda_a^r, & n_a^r \in N^D \\
y_{ba}^r + y_{ab}^r + x_{ab}^r = 1, & \{a, b\} \subseteq A(r), r \in R \\
t_b^r - t_a^{r+1} \geq -M(1 - y_{ab}^r), & \{a, b\} \subseteq A(r), r \in R \\
t_a^r - t_b^{r+1} \geq -M(1 - y_{ba}^r), & \{a, b\} \subseteq A(r), r \in R \\
t_b^{r+1} - t_a^r \geq -M(1 - x_{ab}^r), & \{a, b\} \subseteq A(r), r \in R \\
t_a^{r+1} - t_b^r \geq -M(1 - x_{ab}^r), & \{a, b\} \subseteq A(r), r \in R \\
\sum\limits_{\{a,b\} \subseteq Q} x_{ab}^r \leq \binom{|Q|}{2} - 1, & Q \in \mathcal{A}(r), r \in R \\
y_{ab}^r, y_{ba}^r, x_{ab}^r \in \{0, 1\}, & \{a, b\} \subseteq A(r), r \in R \\
t_a^r \in \mathbb{R}, & n_a^r \in N \\
t_a^{\text{out}} \in \mathbb{R}, & a \in A \\
t^o \in \mathbb{R}. &
\end{array}
\qquad (8)
$$

As mentioned above, the cost function $c(t)$ in (8) usually consists of the sum of the weighted arrival time at destination of all trains, that is $c(t) = \sum_{a \in A} w_a t_a^{\text{out}}$, where w_a is the weight of train a. However, this can be generalized to more complex functions. For example, an objective function commonly used in the railway industry is a piece-wise linear one, where the delay of a train is taken into account only if it is greater than few minutes[5].

3 AN EXTENDED MILP FORMULATION

As discussed in the previous section, the model in (8) assumes that travel times in tracks are constant and do not depend on the fact the train has stopped at the previous station or that it is going to stop in the next station. Similarly with travel times in stations. While this assumption is usually tolerated for passenger trains, it cannot be applied to heavy freight trains. In fact, they usually need a very long time to reach the nominal speed after they stopped, and to reach a full stop when travelling at nominal speed. Based on theese observations, we identified four different running times for each track and two different running times for each station.

We define by $R^T, R^S \subset R$ as the sets of resources that represent tracks and stations, respectively. Now, for each track $r \in R^T$ and for each train $a \in A$ we have:

- Λ_a^r: the minimum running time if a does not stop at the previous or next station;
- $\Lambda_a^r + \bar{\Delta}_a^r$: the minimum running time if a stops at the next station but not at the previous one;
- $\Lambda_a^r + \underline{\Delta}_a^r$: the minimum running time if a stops at the previous station but not at the next one;
- $\Lambda_a^r + \bar{\Delta}_a^r + \underline{\Delta}_a^r$: the minimum running time if a stops both at the previous and next station.

Instead, for each station $r \in R^S$ and for each train $a \in A$ we have:

- Λ_a^r: the minimum running time if a does not stop in this station;
- $\Lambda_a^r + \Delta_a^r$: the minimum running time if a stops in this station.

In order to include this flexible running times into model (8), we introduce binary variables $z_a^r, a \in A, r \in R^S$ that are equal to

[4] Observe that, by Helly's property, if every pair of trains in Q meet in r, then there exist a point in time where all trains in Q are simultaneously in r.

[5] Note that the delay of a train $a \in A$ can be computed by subtracting the originally scheduled arrival time at destination from t_a^{out}.

1 if train a stops in station r, 0 otherwise. In other words, we introduce this additional (indicator) constraint:

$$z_a^r = 0 \qquad \Longrightarrow \qquad t_a^{r+1} - t_a^r = \Lambda_a^r. \tag{9}$$

Similarly to what done in (6), we can linearize this constraint by using the big-M trick:

$$t_a^{r+1} - t_a^r \le \Lambda_a^r + M z_a^r. \tag{10}$$

Indeed, when z_a^r is equal to 0, then this constraint together with (2) imply that the travel time of train a in r is exactly Λ_a^r, which means that the train did not stop and traversed the station at its nominal speed.

With these "stopping" binary variables at hand, we can now define the constraints that model the travel times in stations and tracks more accurately.

For each station $r \in R^S$, and each train $a \in A$, we introduce the following set of constraints:

$$t_a^{r+1} - t_a^r \ge \Lambda_a^r + \Delta_a^r z_a^r. \tag{11}$$

Similarly, for each track $r \in R^T$, and each train $a \in A$, we have:

$$t_a^{r+1} - t_a^r \ge \Lambda_a^r + \bar{\Delta}_a^r z_a^{r-1} + \underline{\Delta}_a^r z_a^{r+1}, \tag{12}$$

where z_a^{r-1}, z_a^{r+1} represent the stopping variables at the previous and next stations, respectively.

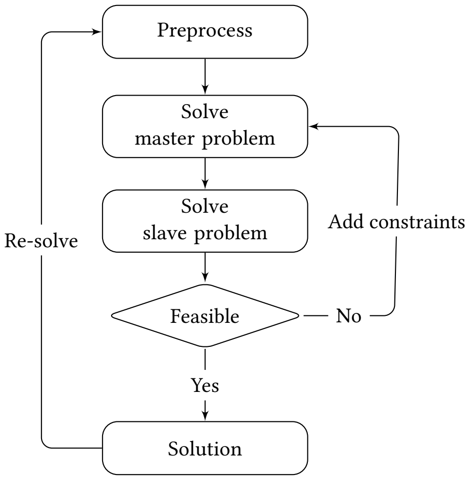
We can now substitute constraints (2) with constraints (10), (11), and (12) to obtain the final MILP model:

$$\begin{array}{ll}
\min\ c(t) & \\
\text{subject to:} & \\
t_a^r - t^o \ge \Gamma_a, & n_a^r \in N^O \\
t_a^{r+1} - t_a^r \ge \Lambda_a^r + \Delta_a^r z_a^r, & n_a^r \in N \setminus N^D, r \in R^S \\
t_a^{r+1} - t_a^r \ge \Lambda_a^r + \bar{\Delta}_a^r z_a^{r-1} + \underline{\Delta}_a^r z_a^{r+1}, & n_a^r \in N \setminus N^D, r \in R^T \\
t_a^{r+1} - t_a^r \le \Lambda_a^r + M z_a^r, & n_a^r \in N \setminus N^D, r \in R^S \\
t_a^{\text{out}} - t_a^r \ge \Lambda_a^r, & n_a^r \in N^D \\
y_{ba}^r + y_{ab}^r + x_{ab}^r = 1, & \{a,b\} \subseteq A(r), r \in R \\
t_b^r - t_a^{r+1} \ge -M(1 - y_{ab}^r), & \{a,b\} \subseteq A(r), r \in R \\
t_a^r - t_b^{r+1} \ge -M(1 - y_{ba}^r), & \{a,b\} \subseteq A(r), r \in R \\
t_b^{r+1} - t_a^r \ge -M(1 - x_{ab}^r), & \{a,b\} \subseteq A(r), r \in R \\
t_a^{r+1} - t_b^r \ge -M(1 - x_{ab}^r), & \{a,b\} \subseteq A(r), r \in R \\
\sum\limits_{\{a,b\} \subseteq Q} x_{ab}^r \le \binom{|Q|}{2} - 1, & Q \in \mathcal{A}(r), r \in R \\
y_{ab}^r, y_{ba}^r, x_{ab}^r \in \{0,1\}, & \{a,b\} \subseteq A(r), r \in R \\
z_a^r \in \{0,1\}, & n_a^r \in N, r \in R^S \\
t_a^r \in \mathbb{R}, & n_a^r \in N \\
t_a^{\text{out}} \in \mathbb{R}, & a \in A \\
t^o \in \mathbb{R}. &
\end{array} \tag{13}$$

4 SOLUTION APPROACH

In our real-life implementation of train rescheduling, Problem (13) is solved iteratively every 10 seconds. Each time, the current status of the trains (i.e. position, speed, etc.) is gathered from the field, along with the current status of the rail network. The associated initial event graph is built. This is the pre-processing phase. Then, the MILP associated to the current instance is solved. The solution method is based on the decomposition in the macro problem and the micro problems described in Section 2 and shown in Figure 2. Indeed, this can be seen a master-slave approach, as described in [7]. If the master (or macro) problem is infeasible, then there is no solution to the scheduling problem (and we are in a *deadlock* situation). Otherwise, it produces an optimal *tentative* schedule t^*. Recall that the schedule variables in the macro problem are associated with the times each train enters a macro railway resource, in our case a station or a track between two stations in the line. Thus, the schedule t^* may be interpreted as a tentative (*disposition*) timetable. Next, this timetable is used by all micro (or slave) problems in order to solve the routing/scheduling within each station or track (even though for tracks this is trivial). If they are all feasible, then we have found the optimal solution. Otherwise, we generate the constraints that invalidate the current schedule in at least on piece of the railway, forcing the master problem to find a new tentative schedule.

Figure 3: Solution Algorithm



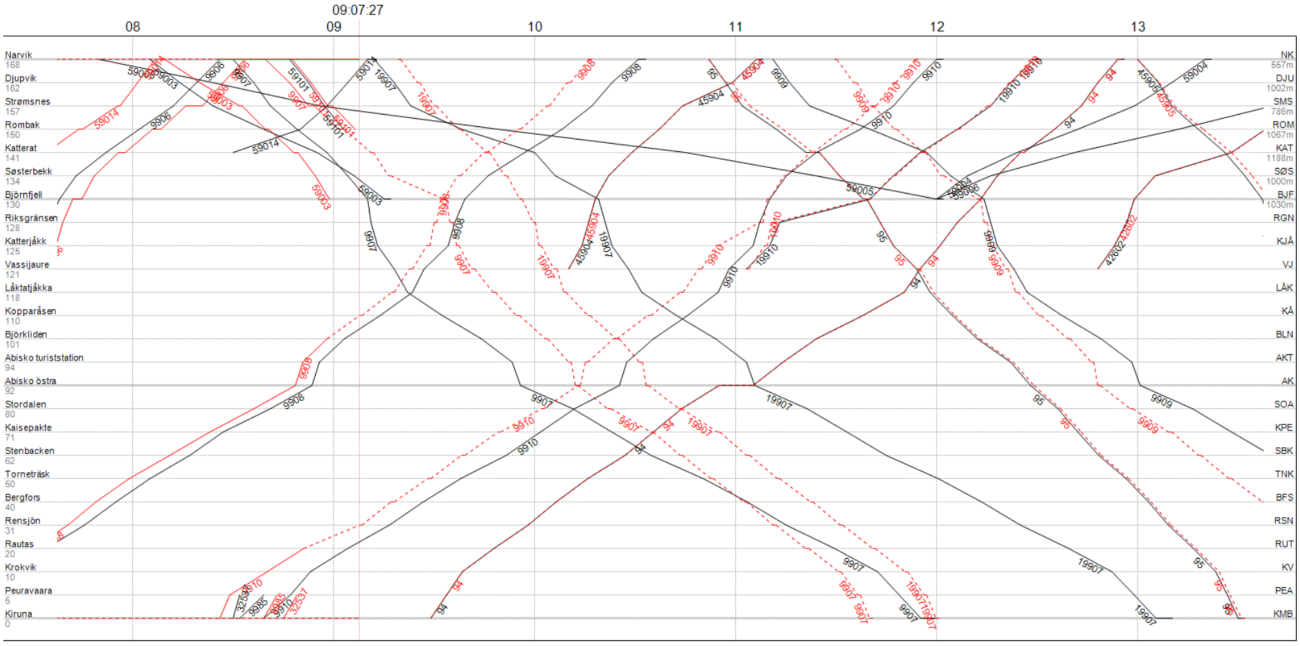
In Figure 3 we give a schematic representation of the overall algorithm. First, the real-time data are pre-processed and the event graph is updated (pre-processing). Second, the MILP Master Problem is solved considering only a subset (initially empty) of constraints (7): this MILP is called *restricted master.* The current master solution is then checked for feasibility by solving the slave problems. Namely, we check if any constraint (7) not included in the current restricted master is violated by the current solution - we call such violation a *conflict.* Observe that, while checking conflicts on tracks is in general a simple exercise, a similar check for stations can be indeed hard (some polynomial cases of practical interest are discussed in [7]). If this is the case, the violated constraints are added to the master problem and the process is iterated until no violated constraints exists.

Note that this delayed row-generation approach usually generates models that are much smaller than the ones generated by the full MILP formulation (see [7]). This helps to drastically reduce the computation time.

5 RESULTS & CONCLUSIONS

Implementing an algorithm of this sort in real-life poses, as described earlier, some additional challenges both with respect to removing theoretical assumptions and adapting formulations to problem specific peculiarities. Another major challenge is to interface with real-time systems in order to get the correct data in real-time. Collaboration with the dispatchers is very important as they have the final word on whether the decisions suggested by the algorithm is accepted or not. Here eliciting why they make their decisions should not be underestimated. The interaction

Figure 4: Train graph



with the dispatchers does also make it difficult to compare the algorithm to the current as we do simply not know what would have happened had the dispatchers accepted all dispatching suggestions.

The algorithm has been implemented into a user interface where we are able to show the dispatchers a classical train graph representing the line, see Figure 4. On the x-axis we have time, both past and present separated by a red line. The stations along the line are placed on the y-axis. Each train is shown as a line in the graph with its train number, the black line is the schedule, the full red line represent the past real-time data. Where the dashed red line is the future dispatching suggestions. This train graph has been tested over a period by the dispatchers in Narvik operational constrol center in real-time.

When running in a real-time setting the input data in the algorithm is updated every 10 seconds before it is executed again. Testing on real-time data over an extended period the approach presented in this paper has been able to provide solutions within (the wanted) 2 seconds. The planning horizon covers the following 2 hours, and the solution returned must be conflict free. The computing speed is extremely important as solutions which are constantly updated on the status of the trains and of the railway must be presented to dispatchers. Hence, with longer solution times the dispatching suggestions might already be out of sync with the real-time data.

The implementation was funded by the Norwegian National Research Council, and the system was indeed operative only on the Norwegian side of the line, supporting the Norwegian dispatchers sitting at Narvik. However, for the Swedish part of the line, there is a second control center located in Boden (not far from the Gulf of Bothnia where the line ends). It is worth noticing here that the limited coordination between the two brains of the line generates various problems. The dispatchers at Narvik may become aware of scheduling decisions taken at Boden only when the trains are approaching the border and in any case they cannot affect such decisions (if not occasionally through some laborious negotiations on the phones). It should be apparent that having a single optimization tool on both sides of the border would significantly increase the coordination, the quality of the overall solutions and the awareness of both teams of dispatchers.

REFERENCES

[1] Egon Balas. 1969. Machine sequencing via disjunctive graphs: an implicit enumeration algorithm. *Operations research* 17, 6 (1969), 941–957.

[2] Srinivas Bollapragada, Randall Markley, Heath Morgan, Erdem Telatar, Scott Wills, Mason Samuels, Jerod Bieringer, Marc Garbiras, Giampaolo Orrigo, Fred Ehlers, et al. 2018. A Novel Movement Planner System for Dispatching Trains. *Interfaces* 48, 1 (2018), 57–69.

[3] Quentin Cappart and Pierre Schaus. 2017. Rescheduling railway traffic on real time situations using time-interval variables. In *International Conference on AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, June 5–8, 2017, Padua, Italy.* Springer, Cham, 312–327.

[4] Francesco Corman and Lingyun Meng. 2015. A review of online dynamic models and algorithms for railway traffic management. *IEEE Transactions on Intelligent Transportation Systems* 16, 3 (2015), 1274–1284.

[5] Matteo Fischetti and Michele Monaci. 2017. Using a general-purpose mixed-integer linear programming solver for the practical solution of real-time train rescheduling. *European Journal of Operational Research* 263, 1 (2017), 258–264.

[6] Pavle Kecman, Francesco Corman, Andrea D'Ariano, and Rob MP Goverde. 2013. Rescheduling models for railway traffic management in large-scale networks. *Public Transport* 5, 1-2 (2013), 95–123.

[7] Leonardo Lamorgese and Carlo Mannino. 2015. An exact decomposition approach for the real-time train dispatching problem. *Operations Research* 63, 1 (2015), 48–64.

[8] Leonardo Lamorgese, Carlo Mannino, Dario Pacciarelli, and Johanna Törnquist Krasemann. 2018. Train Dispatching. In *Handbook of Optimization in the Railway Industry.* Springer, Cham, 265–283.

[9] Leonardo Lamorgese, Carlo Mannino, and Mauro Piacentini. 2016. Optimal train dispatching by Benders'-like reformulation. *Transportation Science* 50, 3 (2016), 910–925.

[10] Carlo Mannino and Alessandro Mascis. 2009. Optimal real-time traffic control in metro stations. *Operations Research* 57, 4 (2009), 1026–1039.

[11] Alessandro Mascis and Dario Pacciarelli. 2002. Job-shop scheduling with blocking and no-wait constraints. *European Journal of Operational Research* 143, 3 (2002), 498–517.

[12] Paola Pellegrini, Grégory Marlière, and Joaquin Rodriguez. 2016. A detailed analysis of the actual impact of real-time railway traffic management optimization. *Journal of Rail Transport Planning & Management* 6, 1 (2016), 13–31.

[13] SCI-Verkher. 2017. Rail transport markets - global market trends 2016-2025.

Minimum Concurrency for Assembling Computer Music

Carlos E. Marciano
Federal University of Rio de Janeiro
Rio de Janeiro, RJ
cemarciano@poli.ufrj.br

Abilio Lucena
Federal University of Rio de Janeiro
Rio de Janeiro, RJ
abiliolucena@cos.ufrj.br

Felipe M. G. França
Federal University of Rio de Janeiro
Rio de Janeiro, RJ
felipe@ieee.org

Luidi G. Simonetti
Federal University of Rio de Janeiro
Rio de Janeiro, RJ
luidi@cos.ufrj.br

ABSTRACT

An effective algorithmic solution for resource-sharing problems in heavily loaded systems is *Scheduling by Edge Reversal (SER)*, essentially providing some level of concurrency by describing an order of *operation* for nodes in a graph. The resulting concurrency is a hard metric to optimize, as the decision problems associated with obtaining its *extrema* have been proved to be NP-complete. In this paper, we propose a novel approach involving longest cycles for solving the *Minimum Concurrency Problem* to proven optimality. Moreover, we show how this model can be used in the field of algorithmic composition to assemble a maximum-length loop of original computer music, capturing fundamental concepts in music theory. To illustrate this strategy, we present a complementary simulation accessible through the *Web*.

1 INTRODUCTION

Resource-sharing problems arise naturally in many scenarios, where graph algorithms are often employed to provide a distributed, asynchronous scheduling solution. By representing each process as a node, we define that nodes are connected by an edge if and only if they share a resource. Specifically, in neighborhood-constrained systems, a process is only allowed to *operate* if and only if all of its neighbors are *idle*, meaning that all of its required resources must be available at the time of *operation*. As a consequence, multiple processes requiring the same resource form a *clique*, a complete sub-graph in which only one node is allowed to *operate* at a time. A connected undirected graph representing resource dependencies among processes, as illustrated in Figure 1(a), will be referred to as a *resource graph* throughout this paper.

Under a heavy load assumption, where nodes are constantly demanding access to their required resources, an effective scheduling algorithm to ensure fairness and prevent starvation is *Scheduling by Edge Reversal (SER)*. Introduced by Gafni and Bertsekas [7] in 1981 and later formalized by Barbosa and Gafni [3] in 1989, *SER* has inspired many distributed resource-sharing applications ranging from asynchronous digital circuits [5] to the control of traffic lights in road junctions [4].

The execution of SER may be summarized as follows: by taking a directed acyclic graph *(DAG)* such as the one in Figure 1(b) as input, *SER* simultaneously *operates* all sinks, meaning that all nodes with no outgoing edges are allowed to utilize the resources they demand to perform their corresponding tasks. Once every sink is done *operating*, the orientation of their incoming edges is reverted, effectively allowing other nodes to become *sinks* themselves. This process is repeated indefinitely, as each new iteration will generate a new *DAG*, allowing different nodes to utilize resources and *operate*. Eventually, orientations will start repeating themselves, leading to the existence of periods. In fact, as observed by Barbosa and Gafni [3], all nodes operate the same number of times within a given period. Figure 1(c) illustrates this procedure.

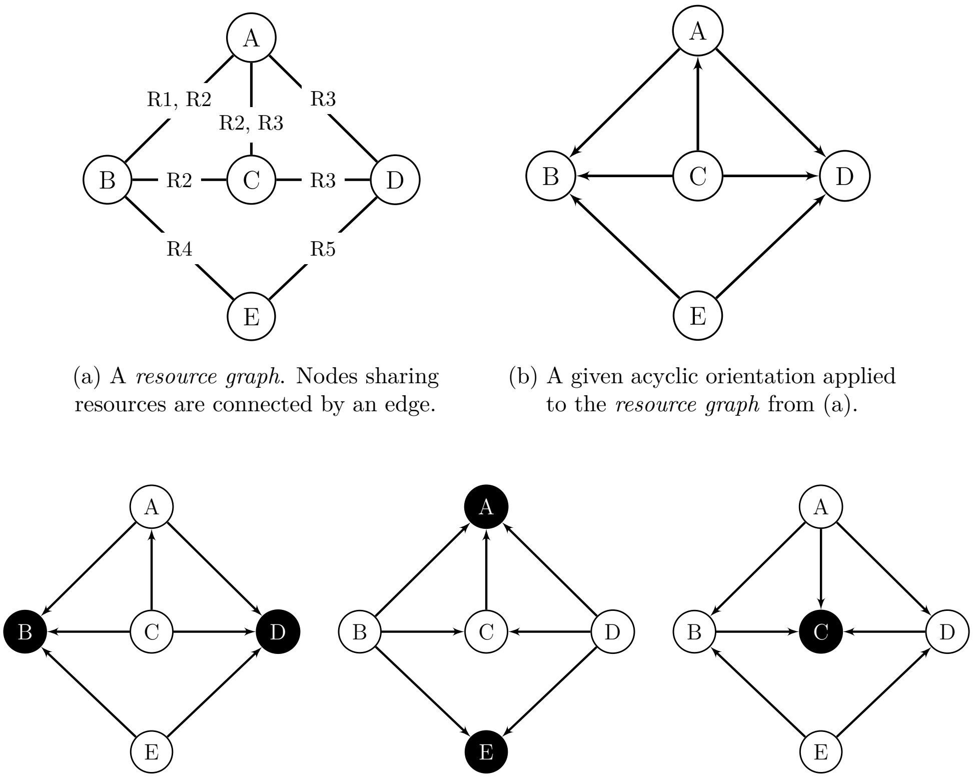
In order to apply *SER* to any *resource graph* and obtain a corresponding schedule, an initial acyclic orientation must be generated. This initial *DAG* will directly impact the overall concurrency of the edge reversal procedure, leading to periods of different lengths and of different orientations. Intuitively, a highly concurrent dynamic will result in more nodes *operating* simultaneously while minimizing the amount of steps where each node is *idle*. Although a formal definition of concurrency is kept for Section 2, it's already inevitable to inquire about the complexity of problems such as obtaining the orientations that lead to the *extrema* of this metric. In fact, the decision problems associated with identifying the maximum as well as the minimum concurrency yielded by a given *resource graph* have been proved to be NP-complete by Barbosa and Gafni [3] and by Arantes Jr [11], respectively.

Contrary to intuition, obtaining the orientations of a resource graph from which *SER* will provide minimum concurrency is advantageous to a number of applications. For instance, Gonçalves et al. have employed *SER* under minimum concurrency to diminish the amount of *Web marshalls* needed for the distributed decontamination of *Webgraphs* [9, 14, 16], while Alves et al. have shown, through simulations of real conflagration scenarios, that less concurrency implies in a reduced number of automated firefighters required to control the flames [2]. However, despite *SER's* intrinsic connection to rhythms, no application in the field of algorithmic composition exists in the literature. As such, this paper presents a novel mechanism which, under minimum concurrency, schedules musical *phrases* to create the lengthiest possible original tracks that capture fundamental concepts in music theory, such as *rhythm* and *polyphony*. This is only possible by developing an optimization strategy for solving the *Minimum Concurrency Problem (MCP)*, which is also presented in this work as an original contribution.

The following is how the remainder of this paper is organized. In Section 2, we recall some graph-theoretic definitions associated with *SER*, including a formal metric for concurrency. Section 3, in turn, describes the concepts involved in our proposed reformulation of *MCP*. Finally, in Section 4, we show how minimum concurrency under *SER* can be used to assemble a maximum-length loop of computer music, expressing our concluding remarks and future work suggestions in Section 5.

ISBN 978-3-89318-079-0 on OpenProceedings, ISSN 2510-7437

DOI: 10.5441/002/inoc.2019.16



(a) A *resource graph.* Nodes sharing resources are connected by an edge.

(b) A given acyclic orientation applied to the *resource graph* from (a).

(c) A schedule (from left to right) taking (b) as the initial orientation. Sinks (in black) are allowed to *operate*, after which they revert their edges so new sinks may be formed. Once node C is done *operating*, the leftmost orientation will be repeated, leading to a period.

Figure 1: Scheduling by Edge Reversal as a distributed solution for scheduling processes (nodes) in a resource-sharing system.

2 GRAPH-THEORETIC BACKGROUND

Initially, as defined in Barbosa and Gafni [3], we shall characterize the necessary terminology to define *concurrency* under *SER*. As such, let $G = (V, E)$ be a connected undirected graph where $|E| \geq |V|$ (i.e. G is not a tree). Let $\kappa \subseteq V$ denote an undirected simple cycle in G, that is, a set of vertices that form a sequence of length $|\kappa| + 1$ of the form $i_0, i_1, ..., i_{|\kappa|-1}, i_0$. If κ is traversed from i_0 to $i_{|\kappa|-1}$, we say that it is traversed in the clockwise direction. Otherwise, we say that it is traversed in the counterclockwise direction. Let K denote the set of all simple cycles of G.

Moreover, an *acyclic orientation* of G is a function expressed as $\omega : E \longrightarrow V$ such that no undirected cycle κ of the form $i_0, i_1, ..., i_{|\kappa|-1}, i_0$ exists for which $\omega(i_0, i_1) = i_1$, $\omega(i_1, i_2) = i_2$, ..., $\omega(i_{|\kappa|-1}, i_0) = i_0$. Let Ω denote the set of all acyclic orientations of G.

Lastly, given an undirected simple cycle κ and an *acyclic orientation* ω, let $n_{cw}(\kappa, \omega)$ be defined as the number of edges oriented clockwise by ω in κ. Similarly, let $n_{ccw}(\kappa, \omega)$ be defined as the number of edges oriented by ω in the counterclockwise direction. Therefore, the *concurrency* of a graph G is defined as a function $\gamma : \Omega \longrightarrow \mathbb{R}$ such that:

$$\gamma(\omega) = \min_{\kappa \in K} \left\{ \frac{\min \{n_{cw}(\kappa, \omega), n_{ccw}(\kappa, \omega)\}}{|\kappa|} \right\} \qquad (1)$$

In other words, given an orientation ω, we check every simple undirected cycle κ of G and calculate the number of edges oriented in the clockwise direction as well as the number of edges oriented in the counterclockwise direction. We take the minimum of these two values and divide the result by the size of the undirected cycle κ. Whichever $\kappa \in K$ returns the smallest value will dictate the system's concurrency.

Finally, we must note that an equivalent result can also be obtained from a dynamic analysis. Let a *period* of length p be a sequence of distinct acyclic orientations $\alpha_0, ..., \alpha_{p-1}$ induced by the execution of *SER*. Let m be the number of times a node *operates* within a *period*, which is equal to all nodes. The expression $\gamma(\omega) = m/p$ is equivalent to Equation 1, despite being less significant to this paper. As an example, the concurrency provided by the schedule in Figure 1(c) is equal to 1/3, and can be obtained through both expressions.

3 OBTAINING MINIMUM CONCURRENCY

Our main goal in this section is to propose a linear-time algorithm for obtaining the minimum concurrency yielded by a resource graph G given one of its longest simple cycles as input. This reduction will essentially provide a computational model for the *Minimum Concurrency Problem (MCP),* allowing previously developed techniques for the *Longest Cycle Problem (LCP* [8]) to also be effective for *MCP.*

Initially, we shall derive a different expression for minimizing $\gamma(\omega)$ over all $\omega \in \Omega$. Given that Ω is a finite set, let γ^* denote the minimum value that Equation 1 assumes over all $\omega \in \Omega$:

$$\gamma^* = \underset{\omega\in\Omega}{min}\left\{\underset{\kappa\in K}{min}\left\{\frac{min\,\{n_{cw}(\kappa,\omega), n_{ccw}(\kappa,\omega)\}}{|\kappa|}\right\}\right\} \tag{2}$$

The following lemma holds:

Lemma 3.1. $\gamma^* = \underset{\kappa\in K}{min}\left\{\frac{1}{|\kappa|}\right\}$.

Proof. Consider Equation 1. For a given ω', let κ' be the simple cycle that minimizes the internal fraction. Let x be defined as $x = min\,\{n_{cw}(\kappa',\omega'), n_{ccw}(\kappa',\omega')\}$, bringing Equation 1 to a value of $\gamma(\omega') = x/|\kappa'|$.

However, for every $\kappa \in K$, there will always exist an acyclic orientation ω such that $n_{cw}(\kappa,\omega) = 1$ and $n_{ccw}(\kappa,\omega) = |\kappa|-1$, or vice versa (this follows immediately from the fact that a directed cycle would only exist if and only if either $n_{cw}(\kappa,\omega) = 0$ or $n_{ccw}(\kappa,\omega) = 0$).

Therefore, there must also exist an orientation ω for κ' such that either $n_{cw}(\kappa',\omega) = 1$ or $n_{ccw}(\kappa',\omega) = 1$. Consequently, if ω', when applied to κ', didn't produce the result $x = 1$, there will necessarily exist another acyclic orientation ω that will lead to $\gamma(\omega) = 1/|\kappa'|$.

Now, consider Equation 2. If γ^* is less than $1/|\kappa'|$, then there must exist a simple cycle κ^* which, under an orientation ω^*, will produce $1/|\kappa^*| < 1/|\kappa'|$. As such, Equation 2 has become a minimization problem over all $\kappa \in K$. □

Lemma 3.1 is essentially the problem of finding a longest undirected cycle of G, whose minimum concurrency will be equal to the reciprocal of the size of its circumference.

We now show how to obtain ω^*, an orientation for which $\gamma(\omega^*) = \gamma^*$. Let κ^* be a longest simple cycle of G, meaning that $|\kappa^*| \geq |\kappa|$ for all $\kappa \in K$. The following theorem holds:

Theorem 3.2. *Given any longest cycle $\kappa^* \in K$ as input, there exists a linear-time algorithm for finding an orientation $\omega^* \in \Omega$ such that $\gamma(\omega^*)$ is minimum over all $\omega \in \Omega$.*

Proof. The proof of Lemma 3.1 states that minimum concurrency will be attained if an orientation ω^* is applied to G under the condition that $n_{cw}(\kappa^*,\omega^*) = 1$ and $n_{ccw}(\kappa^*,\omega^*) = |\kappa^*| - 1$ or vice versa, where κ^* is a longest cycle. Orienting κ^* under the aforementioned conditions can be performed in linear-time by traversing the cycle κ^* and assigning an increasing identification number $1, ..., |\kappa^*|$ to each visited vertex, resulting in a topological ordering of the cycle. By orienting the corresponding edges towards the vertices with lower identification numbers, only one edge (connecting the vertices with the highest and the lowest identification numbers) will be oriented in the opposite direction from the other $|\kappa^*| - 1$ edges, fulfilling the requirement.

It is now necessary to prove that it is possible to orient the remaining edges of G such that the resulting orientation ω^* is always acyclic. Let $S = V - \kappa^*$ be the set of the remaining vertices of G. Let us assign an increasing identification number $|\kappa^*| + 1, ..., |V|$ to each vertex in S, and then orient all edges of G towards the vertices with lower identification numbers. By contradiction, if the resulting orientation ω^* were cyclic, there would need to exist a path $i_0, i_1, ..., i_0$ (i.e. a directed cycle). However, since edges always lead to vertices of lower identification numbers, it is impossible to return to i_0 after leaving it, for any $i_0 \in V$. As such, no cycles are formed. □

Algorithm 1: A linear-time algorithm for finding an acyclic orientation that leads to minimum concurrency given a longest cycle as input.

```
Input  : Undirected graph $G = (V, E)$ and longest
         cycle $\kappa^* \subseteq V$
Output: Acyclic orientation $\omega^*$ for which $\gamma(\omega^*)$ is
         minimum
$id = 1$
$v = \kappa^*.getFirstVertex()$
for $i$=1 to $\kappa^*.size()$ do
    Assign $id$ to $v$
    Increment $id$
    v = $\kappa^*.getClockwiseNeighborOf(v)$
end
while a vertex $v \in V$ with no id exists do
    Assign $id$ to $v$
    Increment $id$
end
Create an empty orientation $\omega^*$
foreach undirected edge $uv \in E$ do
    if $id(v) > id(u)$ then
        Orient edge such that $\omega^*(u, v) = u$
    end
    else
        Orient edge such that $\omega^*(u, v) = v$
    end
end
return $\omega^*$
```

Finally, we structure the proof discussed in Theorem 3.2 as the algorithmic procedure presented in Algorithm 1. Its correctness relies on the aforementioned proof. Note that linear-time is attained only if the method *getClockwiseNeighborOf*(v) is $O(1)$. This will depend on the data structure used for storing κ^*, which is usually an array containing the vertices of the cycle in the order they should be visited. In this case, *getClockwiseNeighborOf*(v) will simply return the next element in the array and fulfill the $O(1)$ requirement. Since G is always a connected graph where $|E| \geq |V|$ as defined in Section 2, the overall time complexity of the algorithm is $O(m)$, where $m = |E|$.

4 ASSEMBLING COMPUTER MUSIC

As expressed by Shan and Chiu [19], effective computer music generation is the dream of computer music researchers. Previous explicit approaches (where composition rules are specified by humans) have resorted to *Hidden Markov Models* to capture the sequence requirements of melody [17], but are usually limited to composing *counterpoint* or *harmonization* for already existing tunes [6].

In this section, we show how a system under *SER's* minimum concurrency is capable of generating a maximum-length loop of pre-recorded musical *phrases*, while respecting fundamental concepts in music theory and creating original melodies for blues, jazz and rock music. In Subsection 4.1, we introduce the terminology that will be used throughout Subsection 4.2 to provide a strategy for representing musical *phrases* as graphs. Lastly, in Subsection 4.3, we discuss implementation-specific details for a complementary simulation included in Appendix A.

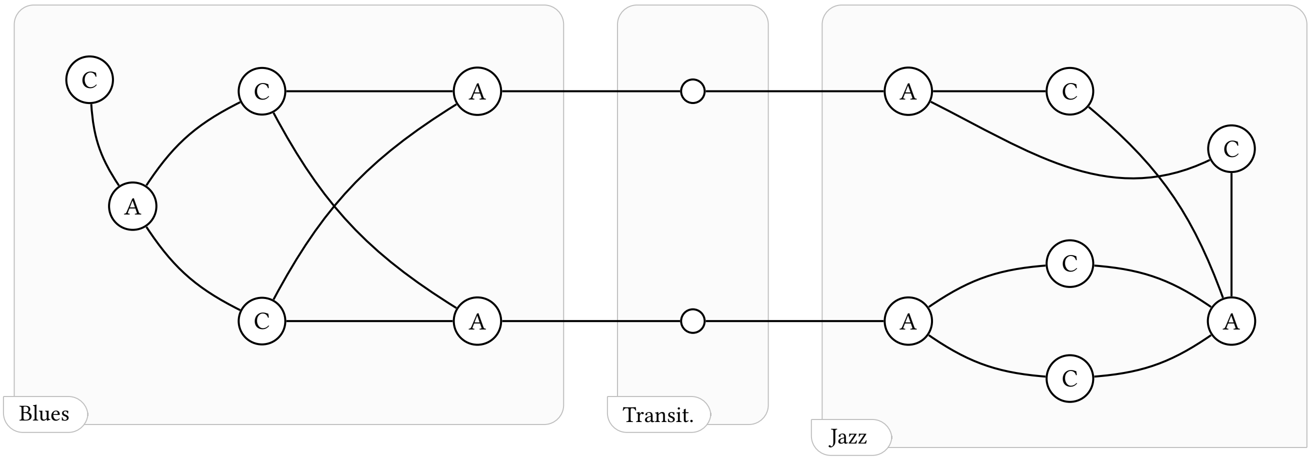


Figure 2: A *resource graph* where nodes marked as "A" and "C" represent *antecedent* and *consequent phrases*, respectively. Nodes connected by an edge are unable to be executed simultaneously, but are allowed to be played in sequence.

4.1 Music Theory Definitions

Initially, we shall define the necessary terminology from music theory employed throughout this section, for which we resort to Schmidt-Jones' book [18]. A musical **phrase** corresponds to a group of individual notes that, together, express a definite melodic idea. It is customary for *phrases* to appear in pairs: the first *phrase* often sounds unfinished until it is completed by the second, almost as if the latter were answering a question posed by the former. *Phrases* that respect this dynamic are called **antecedent** and **consequent**, respectively.

A **bar** (or *measure*) is a group of *beats* that occur during a segment of time. When more than one independent melody takes place during the same *bar*, we call a piece of music **polyphonic** (e.g. Pachelbel's "Canon"; last chorus of "One Day More", from the musical "Les Miserables"). Finally, a **lick**, or *short motif*, corresponds to a brief musical idea that appears in many pieces of the same genre. In this work, a pair of *antecedent* and *consequent phrases*, when played sequentially, will also be referred to as a *lick*.

4.2 Graph Representation

Although we believe that music generation through *SER* can be employed to assemble any musical unit (such as chords or individual notes) into a composition, the application we propose revolves around scheduling *phrases*. Specifically, we would like to capture the following requirements:

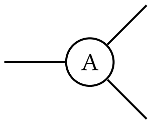
(i) A **consequent** *phrase* may only be played after an **antecedent** *phrase*, forming a *lick*;

(ii) If two or more *phrases* are playing at the same time, either they are all **antecedent** or all **consequent**;

(iii) *Phrases* of different intensities (e.g. number of notes) may not go well together;

(iv) The final composition must be a loop, contain all available *phrases* and be of **maximum length**.

When arranging previously recorded (or generated) *phrases* into a graph, our goal is to structure which *phrases* can be played sequentially and which can be played simultaneously, creating a *polyphony*. By representing each *phrase* as a node, we are able to capture the aforementioned restrictions through the insertion of edges. In a *resource graph*, an edge between two nodes represents the inability of those nodes to *operate* at the same time. As such, an edge between two *phrases* is able to prevent them from occurring during the same *bar*, while allowing each separate *phrase* to be played in sequence.



(a) An *antecedent* phrase.

Property	Value
Type	*Antecedent*
Genre	Blues
Note Count	8
File	antec02.mp3

(b) Node attributes.

Figure 3: An example of a node and its attributes.

Above, in Figure 3, we present the information contained within each node. A *note count*, corresponding to the number of notes within a phrase, is used to measure its intensity. For this specific example, two nodes will be connected to each other if and only if:

(1) they're of different types (*antecedent* and *consequent*);

(2) their *note count* is within a specified threshold;

(3) and they belong to the same genre.

Moreover, we'd like to make this example more interesting by allowing a transition between two different genres: *blues* and *jazz*. By introducing **transitional phrases** that incorporate elements from both genres, a more seamless changeover can be achieved. *Antecedent phrases* from *blues* and *jazz*, when connected to *transitional nodes*, can act as gateways that allow access to their respective genres.

Figure 2 illustrates all the previously discussed components. Nodes marked as "A" and "C" represent *antecedent* and *consequent phrases*, respectively. The further a node is from a *transitional* node, the more *intense* is the *phrase* it represents. Due to the *antecedent / consequent* dynamic, the resulting graph is *bipartite*, for which *MCP* remains NP-complete [12].

4.3 Implementation Details

In order to demonstrate the ideas discussed in Subsection 4.2, we have developed a simulation showing how the *phrase*-scheduling dynamic, when applied to a graph such as the one in Figure 2, is able to produce musical loops of maximum length. In this subsection, we document our steps and discuss implementation details that may be useful for future work. The final result, featuring the *resource graph* from Figure 2, is presented in Appendix A.

As discussed in Section 3, the first step in the process of obtaining minimum concurrency is identifying a longest simple cycle. Although this task is visually straightforward when considering the *resource graph* from Figure 2, larger instances require a computational approach. As such, we relied on the *Simple Cycle Problem* branch-and-cut strategy proposed in Lucena, Cunha and Simonetti [15], which is based on a formulation that decomposes simple cycles into one simple path and an additional edge. We implemented this procedure in the **C** programming language and used the *XPRESS Mixed Integer Programming* package to solve linear programs and manage the branch-and-cut tree.

Despite the example from Figure 2 only containing 15 nodes, our computational results have shown that the aforementioned strategy is able to solve, in under 1 hour, instances of random graphs with as many as 2 000 nodes and 40 034 edges (probability $p = 0.01$ for an edge to exist between two nodes), being an appealing approach for larger instances. In turn, a linear-time implementation of Algorithm 1 is employed to provide an acyclic orientation for the *resource graph*, yielding minimum concurrency. The pipeline presented in Figure 4 summarizes this process.

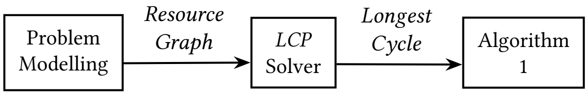


Figure 4: Implementation pipeline for solving *MCP*.

Note, however, that initial orientations may violate requirement (ii), which states that *antecedent* and *consequent* phrases are not allowed to be played together. This is because sinks may be formed anywhere in the graph when orienting nodes outside the original longest cycle. However, this is merely an initialization issue: once a *SER period* is reached, the system will enforce, through the edge-reversal dynamic, that *antecedent phrases* will only become sink nodes when a *consequent phrase* reverts its edges, and vice versa.

Having attained minimum concurrency for the *resource graph* in Figure 2, we switched our attention to developing a visualization strategy. From a compatibility perspective, a web simulation built in **JavaScript** is both lightweight and easy to access on most platforms. Moreover, two convenient libraries, available under the *MIT License*, made this choice even more appealing: **Vis.js** [1], which enabled us to visually represent any graph and handle the necessary edge-reversal dynamics; and **Howler.js** [20], providing a reliable audio interface when dealing with multiple files.

Finally, we curated audio recordings responsible for the *rhythm* sections (also known as *backing tracks*) and recorded all *antecedent* and *consequent phrases* on an electric guitar. Given that this small simulation is comprised of only 15 nodes, the process of syncing each *phrase* to their corresponding *backing track* was performed manually. For instance, a *12-Bar Blues* composition may alternate between *antecedent* and *consequent phrases* every 2 *bars*. Different *phrases* have different starting points within this window, requiring an offset to account for synchronization. However, once synced, *phrases* may be played whenever a new *2-bar* window starts. As such, by setting the edge-reversal frequency to *2 bars*, every *phrase* will sound natural when their corresponding node becomes a sink.

5 CONCLUSION

In this paper, two main contributions to *SER* were presented: first, we reformulated the *Minimum Concurrency Problem*, providing a viable approach for its optimization and allowing many empirically attractive *LCP* solvers to also be effective for *MCP*. Secondly, we proposed a novel strategy for assembling original computer music, which schedules all available *building blocks* (in our example, musical *phrases*) into a maximum-length loop, all the while incorporating essential music-theoretical restrictions.

Regarding *SER's* debut in algorithmic composition, we are eager to discover how other researchers and musicians may employ this technique and its variations to create unique songs. We note that the *Web* is a never-ending repository of musical *phrases*, many of which are encoded in *MIDI* format. *MIDI* is a technical standard that allows a musical pattern to be described and synthesized by a computer [10], replacing the need for physical recording and manual synchronization. This gain in development speed can allow for the modelling of truly large *resource graphs*, producing hour-long tracks of exclusively distinctive music.

Another aspect that can be investigated is controlling the level of *polyphony* within a song. For instance, higher concurrency values imply in a large number of independent melodies occurring during the same *bar*, which may lead to undesirable noise throughout the composition. As such, minimum concurrency not only provides a maximum-length loop of music, but also avoids an oversaturation of sounds that may lead to low-quality *polyphony*. Currently, we investigate how **octave** information (the frequency range in which the fundamental pitch of each note is found) can be used to control which sounds should be played simultaneously (*e.g.:* a phrase whose notes were recorded near *octave* C_3 could be played alongside a phrase with notes situated around *octave* C_5). This approach would avoid melody lines competing for the same frequency range, leading to more distinguishable and pleasant sounds.

Lastly, we invite other researchers to investigate a viable computational model for the *Maximum Concurrency Problem*, which consists of maximizing Equation 1 over the set of acyclic orientations Ω. This breakthrough would impact many distributed resource-sharing applications, such as routing *Automated Guided Vehicles (AGVs)* [13], scheduling job shop tasks [13] and controlling traffic lights in road junctions [4]. Naturally, new engaging applications that could benefit from *SER's* simplicity are also an interesting theme for future research, especially when combined with new theoretical advancements for this technique.

A MUSICAL SIMULATION

The musical simulation referred to throughout this paper is available online at the following website, and can be viewed in any browser: https://cemarciano.github.io/Song-Generator/.

This simulation is an open-source project distributed under the *GNU GPL v3.0 License*. Source code is available at the following website: https://github.com/cemarciano/Song-Generator.

REFERENCES

[1] B. V. Almende et al. 2015. vis.js - A dynamic, browser based visualization library. (2015). Retrieved February 22, 2019 from http://visjs.org/

[2] Daniel S. F. Alves et al. 2012. A Swarm Robotics Approach To Decontamination. In *Mobile Ad Hoc Robots and Wireless Robotic Systems: Design and Implementation*. IGI Publishing, Hershey, PA, USA, 107–122. https://doi.org/10.4018/978-1-4666-2658-4.ch006

[3] Valmir C. Barbosa and Eli M. Gafni. 1989. Concurrency in Heavily Loaded Neighborhood-Constrained Systems. *ACM Transactions on Programming Languages and Systems* 11, 4 (Oct. 1989), 562–584. https://doi.org/10.1145/69558.69560

[4] D. Carvalho, Fábio Protti, Massimo De Gregorio, and Felipe M. G. França. 2004. A Novel Distributed Scheduling Algorithm for Resource Sharing Under Near-Heavy Load. *Lecture Notes in Computer Science* 3544 (2004), 431–442. https://doi.org/10.1007/11516798_31

[5] Ricardo F. Cassia, Vladmir C. Alves, Frederico G. Besnard, and Felipe M. G. França. 2009. Synchronous-To-Asynchronous Conversion of Cryptographic Circuits. *Journal of Circuits, Systems and Computers* 18, 2 (2009), 271–282. https://doi.org/10.1142/S0218126609005058

[6] Jose D. Fernandez and Francisco Vico. 2013. AI Methods in Algorithmic Composition: A Comprehensive Survey. *Journal of Artificial Intelligence Research* 48, 1 (Nov. 2013), 513–582. https://doi.org/10.1613/jair.3908

[7] Eli M. Gafni and Dimitri P. Bertsekas. 1981. Distributed Algorithms for Generating Loop-Free Routes in Networks with Frequently Changing Topology. *IEEE Transactions on Communications* 29, 1 (Jan. 1981), 11–18. https://doi.org/10.1109/TCOM.1981.1094876

[8] Michael R. Garey and David S. Johnson. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, page 213.

[9] Vanessa C. F. Gonçalves, Priscila M. V. Lima, Nelson Maculan, and Felipe M. G. França. 2010. A Distributed Dynamics for WebGraph Decontamination. *Lecture Notes in Computer Science* 6415 (2010), 462–472. https://doi.org/10.1007/978-3-642-16558-0_39

[10] David Miles Huber. 2007. *The MIDI Manual* (3rd ed.). Routledge, New York, NY, USA. https://doi.org/10.4324/9780080479460

[11] Gladstone M. Arantes Jr. 2006. *Trilhas, Otimização de Concorrência e Inicialização Probabilística em Sistemas sob Reversão de Arestas*. Ph.D. Dissertation. Federal University of Rio de Janeiro, Rio de Janeiro, Brazil. https://www.cos.ufrj.br/index.php/pt-BR/publicacoes-pesquisa/details/15/2039

[12] M. S. Krishnamoorthy. 1975. An NP-hard problem in bipartite graphs. *SIGACT News* 7, 1 (Jan. 1975), 26–26. https://doi.org/10.1145/990518.990521

[13] Omar Lengerke, HernÃạn G. AcuÃśa, Max S. Dutra, F. M. G. FranÃğa, and Felix A. C. Mora-Camino. 2012. Distributed control of job-shop systems via edge reversal dynamics for automated guided vehicles. *International Conference on Intelligent Systems and Applications* 1 (April 2012), 25–30. https://doi.org/10.13140/RG.2.1.3054.5127

[14] Linda Luccio, Fabrizio annd Pagli. 2007. Web Marshals Fighting Curly Link Farms. *Lecture Notes in Computer Science* 4475 (2007), 240–248. https://doi.org/10.1007/978-3-540-72914-3_21

[15] Abilio Lucena, Alexandre Cunha, and Luidi G. Simonetti. 2013. A New Formulation and Computational Results for the Simple Cycle Problem. *Electronic Notes in Discrete Mathematics* 44 (Nov. 2013), 83–88. https://doi.org/10.1016/j.endm.2013.10.013

[16] Marina Moscarini, Rossella Petreschi, and Jayme L. Szwarcfiter. 1998. On node searching and starlike graphs. *Congressus Numerantium* 131 (1998), 75–84. DOI unavailable, must be requested directly from authors.

[17] Gerhard Nierhaus. 2009. *Algorithmic Composition: Paradigms of Automated Music Generation*. Springer-Verlag, Vienna, Austria. https://doi.org/10.1007/978-3-211-75540-2

[18] Catherine Schmidt-Jones. 2007. *Understanding Basic Music Theory*. OpenStax CNX, Houston, TX, USA. http://cnx.org/content/col10363/1.3/

[19] Man-Kwan Shan and Shih-Chuan Chiu. 2010. Algorithmic compositions based on discovered musical patterns. *Multimedia Tools and Applications* 46, 1 (Jan. 2010), 1–23. https://doi.org/10.1007/s11042-009-0303-y

[20] James Simpson et al. 2013. howler.js - JavaScript audio library for the modern Web. (2013). Retrieved February 22, 2019 from https://howlerjs.com/

Routing and Resource Assignment Problems in Future 5G Radio Access Networks

Amal Benhamiche
Orange Labs, Châtillon
France
amal.benhamiche@orange.com

Wesley da Silva Coelho
Orange Labs, Châtillon
France
wesley.dasilvacoelho@orange.com

Nancy Perrot
Orange Labs, Châtillon
France
nancy.perrot@orange.com

ABSTRACT

Given a *mobile network* composed of a set of devices, a set of antennas (Base Stations) and a discrete set of radio resources, we define a *domain* as a subset of devices/antennas that communicate via radio transmission links in order to exchange data for a specific service. In this context, we are interested in the Domain Creation (DC) problem. It consists in finding an allocation of radio resources to the transmission links of the network so that different domains, each one related to a specific service (gaming, video streaming, content sharing, etc.), can be implemented simultaneously. Every domain has specific requirements in terms of quality of the transmission links (SINR) and hardware resources dedicated to carrying out the corresponding service. We give an integer linear programming formulation for the problem and propose two classes of valid inequalities to strengthen its linear relaxation. The resulting formulation is used within a branch-and-cut algorithm for the problem. We further propose an efficient heuristic obtained from solving the routing and resource assignment sub-problems separately. We assess the efficiency of our approaches through some experiments on instances of varying size and realistic input data from Orange mobile network.

1 INTRODUCTION

In future 5G networks, mobile User Equipment (UEs) will be able to host functions that give them new abilities such as sharing connectivity, capacity and CPU resources with other UEs, regardless of the ongoing traditional communications. The 5G wireless technology, along with the evolution of mobile users behavior and needs, will make the current scheme of communication (UE to Base Station) no longer optimal in terms of radio resource utilization. The Device-to-Device (D2D) communication mode is one of the new approaches presented as a promising alternative to traditional communication in cellular networks. A D2D communication is defined as a direct communication between two mobile or fix user devices, without traversing the Base Station (BS) [3]. This technology allows to reuse radio resources and to decrease end-to-end latency of local communications. Then, D2D would allow a set of UEs geographically close to each other to establish direct D2D communications, or span multiple links (multi-hop D2D communications), to access a given service (e.g. video streaming or gaming) while ensuring the required service quality.

A *domain* is defined as the set of UEs and BSs that are used to establish mobile communications (D2D or cellular) related to a specific service. The communication is either direct or uses multiple links (D2D or via the BS). Two UEs can then communicate through cellular links, using the BS or D2D links, and both technologies can coexist within the same mobile network. In any case, radio resources should be allocated to every active link involved in a communication, and the SINR (Signal-to-Interference-plus-Noise Ratio) level required by the service should be ensured.

In this context, we consider a mobile network composed of a set of devices (UEs), a set of antennas (BS) and a set of services eligible to D2D communications, with their associated traffic matrices. These traffic matrices are in the form of data volume to be exchanged between pairs of devices. The involved devices can communicate through one or several links, either using D2D or cellular communication (via the BS). A non-negative weight, corresponding to the SINR, is associated with each link. It is a measure of the quality of the communication that could be established using this link. Every service requires a minimal quality threshold in terms of SINR and available resources (hardware capacity for the devices, radio resources for the links) to be successfully established. The *Domain Creation* (DC) problem consists in finding a minimum cost allocation of the radio resources to the network links so that (*i*) every pair (link, resource) belongs to a unique domain, that is, it is used for a single service; (*ii*) the SINR of each pair (link, resource) assigned to a domain is above the quality threshold required by the corresponding service; (*iii*) every demand is routed from its origin to its destination within a domain and; (*iv*) all the required types of hardware capacities (CPU, RAM, battery) in the devices are satisfied.

State-of-the-art

The problem of assigning radio resources to transmission links is studied in [1] and [2] under the denomination of Frequency Assignment Problem (FAP). In [1], the authors give an ILP formulation for the problem and propose a branch-and-cut algorithm to solve it. The work in [2] presents several variants of FAP and discusses existing and original optimization (including exact and heuristic) approaches to solve them. The routing and resource allocation aspects are combined in the so-called *Routing and Wavelength Assignment* (RWA) problem (see [6] and [11] for instance) which arises in Optical Networks. The DC problem differs from the above cited problems in that traffic demands are *unsplittable* (each demand has to be sent along a unique path using D2D or cellular links), several types of capacities on the devices are considered, and radio resources re-use is submitted to distance constraints so as to avoid radio interference. [13] has focused more specifically on the optimization of mobile network resources using D2D technology. As the related optimization problem is relatively difficult to be solved in competitive time, a greedy heuristic has been proposed as an alternative to solve such a problem. This heuristic is divided into two phases, each one responsible for the allocation of uplinks (from UEs to BSs) and downlinks (from BSs to UEs) in order to improve the total throughput and to significantly reduce the interference between classic and D2D communications.

ISBN 978-3-89318-079-0 on OpenProceedings, ISSN 2510-7437

DOI: 10.5441/002/inoc.2019.17

Our contribution

In this work, we formally define the Domain Creation Problem and we propose a node-arc (compact) ILP formulation to model it. We present some strategies to enhance the linear relaxation of the formulation along with two classes of valid inequalities. Our results are embedded within a branch-and-cut algorithm to solve the problem. We further propose a two-phase heuristic, obtained by decomposing the problem into a routing and a radio resource allocation subproblems. To solve the routing subproblem, we propose two methods : first, a LP-based heuristic from the linear relaxation of the compact formulation, second, a non-compact formulation obtained by generating a subset of relevant paths. Then, the allocation subproblem is transformed into a vertex coloring problem that is solved heuristically by an improved greedy algorithm. This greedy algorithm is compared to a dual bound given by the exact solution of the associated Max-Clique Problem. Numerical experiments are made on instances generated thanks to realistic parameters of Orange mobile networks.

Our paper is organized as follows. We introduce the DC problem in Section 2 and give an ILP formulation along with two classes of valid inequalities. An overview of our branch-and-cut algorithm is given in Section 3 with a brief description of the separation routines used to generate the cuts and some numerical results for small instances. The outline of our two-phase heuristic is detailed in Section 4. Finally, we give some concluding remarks in Section 5 and discuss future works on extensions of the problem.

2 THE DOMAIN CREATION PROBLEM

2.1 Problem definition

The network is represented by a directed graph $G = (V \cup U, A)$, where V is the set of nodes associated with devices, U the set of Base Station nodes, and A the set of arcs. We denote by $\delta^+(u)$ (resp. $\delta^-(u)$) the subset of arcs going from (resp. to) node u. Every node $u \in V$ has an associated weight vector $c^u = \{c^u_1, \ldots, c^u_b\}$, where $c^u_i \geqslant 0$ is the capacity available at node u for the physical resource $i \in C^d$. Every arc $e \in A$ has a weight denoted $SINR_e$ that expresses a measure on the quality of the transmission link represented by e. For every pair of arcs $e, f \in A$ we denote by $d(e, f)$ the *distance* between e and f. This value corresponds to the minimum distance between the opposite ends of the given pair of links, that is, the origin of one and the destination of the other. Namely, two arcs e and f are said to be *close* if $d(e, f) \leqslant D$, where D is a minimum acceptable distance. Let $R = \{1, \ldots, r\}$ be the set of available radio resources. An arc e of A is said to be *active* if a radio resource $r \in R$ is assigned to it. A resource $r \in R$ can be assigned to two different arcs $e, f \in A$ only if $d(e, f) > D$. Indeed, due to interference constraints, two communications cannot be established on e and f simultaneously using the same resource $r \in R$ if e is close to f. We denote by K the set of traffic demands to be routed and S the set of service types. Every demand $k \in K$ is defined by an origin node $o^k \in V$, a destination node $d^k \in V$ and a requested service s_k. Moreover, every $k \in K$ has an associated cost $\mu^e_{s_k}$ to use arc $e \in A$ and a traffic vector $a^{s_k} = (a^{s_k}_1, \ldots, a^{s_k}_b)$ where the element $a^{s_k}_m \geqslant 0$ denotes the quantity of physical resource type m from the set C^d of all resources types (e.g. CPU, RAM, storage) needed to process the service s_k requested by k. Finally, we denote by β_k the quality threshold needed by k to access the required service s_k.

The Domain Creation (DC) Problem consists in finding a minimum cost allocation of the radio resources in R to the active arcs of G so as to provide a feasible routing path for each demand. In particular, a routing $p^k = \{(o^k, u), \ldots, (v, d^k)\}$ for a demand k is said to be feasible if

- all the arcs of p^k have a SINR value above the quality threshold β_k required by the demand k and,
- all the nodes in p^k have enough capacity to satisfy the resource requirements of k.

An instance of the problem, with fives devices and one BS, is illustrated in Figure 1. For sake of clarity, each pair of arcs between two nodes is represented by an edge. The edges representing D2D links are shown in solid lines while edges representing device to BS links are in dashed lines. Two different services, namely *gaming* and *video streaming*, and three demands per service are to be delivered:{(u_1,u_3), (u_2,u_5), (u_4, u_2)} and {(u_4, u_3), (u_3, u_1), (u_5, u_2)}, assuming that twelve radio resources $\{r_1, \ldots, r_{12}\}$, are available. A feasible solution is represented in Figure 2. The figure on the left side represents the *Gaming* domain, where all three demands are satisfied through D2D links. The figure on the right side is the *Video streaming* domain, where just one demand uses the BS. Note that, using the legacy approach, all demands must pass through the BS, using one uplink (from UE origin to BS) and one downlink (from the BS to UE destination) for each demand. Since all active links share at least one arc extremity (BS), the solution would require all the available resources (a different one for each active link) to avoid interference. Thus, using D2D communications allow here to save 50% of radio resources compared to a "fully cellular" solution.

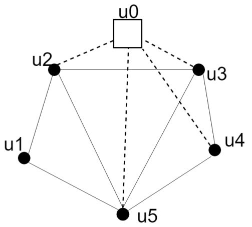


Figure 1: Example of a network with 6 nodes and 6 demands to be delivered for 2 different services

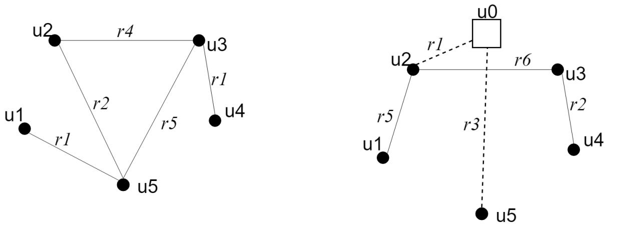


Figure 2: Feasible solution: active links and associated radio resource for *Gaming* domain (left) and *Video streaming* domain (right)

2.2 ILP formulation

In this section, we propose a compact ILP formulation for the DC problem followed by some valid inequalities to be used in a branch-and-cut algorithm.

2.2.1 Notations and formulation. The three types of binary variables are:

- x^k_{er}, $e \in A$, $k \in K$, $r \in R$ that takes the value 1 if the arc e is used

by the demand k and assigned with the resource r, 0 otherwise.
• y_i^k, $i \in V \cup U$, $k \in K$, that takes the value 1 if the node i is used by the demand k, 0 otherwise.
• z^r, $r \in R$ that takes the value 1 if the resource $r \in R$ is assigned with at least one arc, 0 otherwise.

Then, the DC problem can be formulated as:

$$\min \sum_{k \in K} \sum_{e \in A} \sum_{r \in R} \mu_e^{s_k} x_{er}^k + \sum_{r \in R} \psi z^r \tag{1}$$

s.t.

$$\sum_{e \in \delta^-(u)} \sum_{r \in R} x_{er}^k - \sum_{e \in \delta^+(u)} \sum_{r \in R} x_{er}^k = \begin{cases} 1, & \text{if } u = d_k, \\ -1, & \text{if } u = o_k, \\ 0, & \text{otherwise,} \end{cases}$$

$$\forall k \in K, \forall u \in U \cup V, \tag{2}$$

$$x_{er}^k \beta_k \leq SINR_e \qquad \forall\ k \in K, \forall\ e \in A, \forall\ r \in R, \tag{3}$$

$$\sum_{k \in K \setminus T(i)} y_i^k a_m^{s_k} \leq c_m^i \qquad \forall\ i \in V, \forall\ m \in C^d, \tag{4}$$

$$2x_{er}^k - y_i^k - y_j^k \leq 0 \qquad \forall k \in K, \forall r \in R \forall (i,j) = e \in A, \tag{5}$$

$$x_{er}^k + x_{fr}^{k'} \leq z^r \qquad \forall r \in R, \forall k, k' \in K, \forall e \in A, \forall f \in D(e), \tag{6}$$

$$x_{er}^k \in \{0,1\} \qquad \forall k \in K, \forall e \in A, \forall r \in R, \tag{7}$$

$$y_i^k \in \{0,1\} \qquad \forall k \in K, \forall i \in V, \tag{8}$$

$$z^r \in \{0,1\} \qquad \forall r \in R. \tag{9}$$

This formulation has a polynomial number of variables and constraints. The objective (1) is to minimize the total costs composed of non-negative routing and radio resource utilization costs. The first set of inequalities (2) are the flow conservation constraints. They ensure that each demand is routed along a unique path between its origin node and its destination node. Note that such a routing path can either span arcs corresponding to D2D links or use a node of U corresponding to some BS. Inequalities (3) guarantee that a demand for a given service is routed along edges whose SINR satisfies the quality threshold required by this service. (4) express the capacity constraints in every node for the different types of hardware resources, with $T(i)$ being the set of demands $k \in K$ that have $i \in V$ as origin or destination. These capacity constraints are needed only on intermediate nodes, that is, the nodes that are not the origin nor the destination of a given demand $k \in K$. Inequalities (5) are linking constraints and inequalities (6) guarantee that the same radio resource is not assigned to different edges unless they are distant enough, where $D(e)$ is a set of arcs that are close to a given arc $e \in A$. Note that, as the objective is to minimize the resources, in any optimal solution one radio resource at most will be allocated to the same arc for a given demand. Finally, (7)-(9) are the integrity constraints.

The ILP formulation (1)-(9) can be strengthened by replacing inequalities (3) by:

$$x_{er}^k \leqslant \lfloor \frac{SINR_e}{\beta_k} \rfloor, \forall k \in K, e \in A, r \in R. \tag{10}$$

2.2.2 Symmetry. Due to the inherent symmetry of this problem, there is possibly a large number of feasible solutions. The breaking symmetry constraints (11) are inspired by classical inequalities in combinatorial optimization (see [8]), and can help in reducing the number of symmetric solutions in the formulation.

$$z^r \geqslant z^{r+1}, \qquad \forall 1 \leqslant r \leqslant |R| - 1, \tag{11}$$

(11) allow to assign the radio resources in an ordered way, forbidding to use a resource $r + 1$ if r is available. Note that a vector $(\overline{x}, \overline{y}, \overline{z}) \in \{(x,y,z) \in \{0,1\}^{(m \times |R| + n) \times |K| + |R|} : (x,y,z)$ satisfies (2)–(11)$\}$ is clearly a feasible solution to the DC problem.

2.2.3 Valid inequalities. We introduce now two more classes of inequalities valid for the DC problem.

(i) Clique-based inequalities:
Given an instance of DC problem, we define the *conflict graph* associated with a node $u \in V$ as follows. For each capacity type $m \in C^d$, let $\mathcal{H}_u^m = (V_u^m, E_u^m)$ be the undirected graph obtained from the set of demands K as follows. A node v_k in V_u^m is associated with every demand $k \in K$ and there exists an edge $v_k v_l \in E_u^m$ between two nodes v_k, v_l of V_u^m if $a_m^{s_k} + a_m^{s_l} > c_m^u$. In other words, an edge in $\mathcal{H}_u^m$ exists if two demands cannot be packed together in the node u due to the lack of capacity for the resource type m. Consequently, a clique in the graph $\mathcal{H}_u^m$ corresponds to a set of demands that cannot use simultaneously the node u. Hence, we denote by $C_{\mathcal{H}}$ the set of cliques in the graph $\mathcal{H}_u^m$.

Proposition 2.1. *Let u be a node of V and $m \in C^d$ be a physical resource type. Then the following inequalities*

$$\sum_{k \in C} y_u^k \leqslant 1, \forall C \in C_{\mathcal{H}} \tag{12}$$

are valid for the DC problem.

Proof. Let $\widetilde{C}$ be a clique in $\mathcal{H}_u^m$. It is clear that if two demands k_1, k_2 from clique $\widetilde{C}$ use the resource m of node u, the capacity constraint (4) for the resource m will be violated. In other words, each edge $v_{k_i} v_{k_j}$ of the clique $\widetilde{C}$ represents an infeasible packing of the demands k_i, k_j in the node u. □

(ii) Strengthened neighborhood inequalities:
The second family of valid inequalities strengthen constraints (6). They are obtained by considering the *interference* graph $\mathcal{N} = (V_N, E_N)$ defined as follows. Every node $u \in V_N$ corresponds to an arc in A and two nodes u_e, u_f of V_N (associated respectively with the arcs e and f from A) are interconnected by an edge if e and f are close enough from each other (ie if $d(e,f) \leqslant D$). Consequently, a clique in the graph $\mathcal{N}$ corresponds to a subset of arcs in A that are pairwise close, and cannot receive the same radio resource due to interference constraints. Likewise in the conflict graph defined before, we denote by $C_{\mathcal{N}}$ the set of cliques in the graph $\mathcal{N}$.

Proposition 2.2. *The following inequalities*

$$\sum_{k \in K} \sum_{e \in C} x_{er}^k \leqslant z^r, \forall r \in R, C \subseteq C_{\mathcal{N}} \tag{13}$$

are valid for the DC problem.

Proof. Let $\widetilde{C}$ be a clique in $\mathcal{N}$ and u_e, u_f two nodes of V_N that belong to clique $\widetilde{C}$. Clearly, if e and f are allocated the same resource $r \in R$ in a solution, then it cannot be feasible for the DC problem. □

Note that similar inequalities are used in [1] and [2] for the Frequency Assignment Problem.

3 BRANCH-AND-CUT ALGORITHM

3.1 Overview of the algorithm

We have developed a branch-and-cut algorithm for the DC problem based on the results presented in Section 2. The algorithm has been implemented in C++ using CPLEX 12.6 as a LP solver, with presolve heuristics and internal cuts being disabled. We have tested our approach

- first by solving formulation (1)-(9) along with the strengthened SINR inequalities (10) and symmetry breaking (11) inequalities and
- by further using the clique-based (12) and strenghthened neighborhood (13) valid inequalities, in addition to the formulation (1)-(9).

We have used two heuristic procedures to generate dynamically inequalities (12) and (13). Both separation routines rely on a greedy algorithm introduced in [10] for the Independent Set problem, that finds a clique in the conflict graph (respectively the interference graph) with appropriate weights on the nodes. We then add the corresponding violated clique-based (respectively strenghthened neighborhood) inequalities, if any, to the current LP. Both classes of valid inequalities are separated throughout the branch-and-cut tree and several inequalities may be added at each iteration of the algorithm.

3.2 Computational results

We present below some early experiments obtained for a set of small instances containing 5 to 15 nodes and up to 7 demands. For these tests, each scenario contains only 1 antenna and 2 service domains. For each instance, realistic data was provided by Orange, including the network topologies and SINR values. Table 1 shows the impact of inequalities (10) and (11) on the initial model (formulation (1)-(9)). In the first column, the name of each instance refers to the number of users (U#) and demands (D#). The next four columns show the computational time (in seconds) for the initial formulation, then when adding symmetry breaking constraints (11), strenghthened SINR constraints (10) and both constraints, respectively. We can notice that using constraints (11) allows to obtain the best execution time for two out of five instances tested ($U5\_D8$ and $U10\_D5$) while the instances $U5\_D6$ and $U10\_D7$ have the lower runtime when applying constraints (10) and (11), combined. For instance $U15\_D7$, the optimal solution could not be found after 3 hours of execution. We have tested the impact of using our valid inequalities and compared the results to CPLEX branch-and-bound for the strengthened model (formulation (1)-(2) + (4)-(11)). Table 2 shows the results obtained on five instances (same as in Table 1) when (i) no additional cuts are used, (ii) using strenghthened neighborhood inequalities (13) in addition to formulation (1)-(9), (iii) using clique-based inequalities (12) in addition to strengthened model and (iv) both cuts are used in the branch-and-cut. We can observe that the gap at root node is substantially reduced when adding cuts (the gap value decreases from 59.65% to 23.58% for instance $U5\_D8$ when using strenghthened neighborhood cuts) and so for the size of the branch-and-cut tree (for instance $U10\_D5$, we range from 1327 nodes without cuts to 84 nodes when both cuts are used). Overall, the strenghthened neighborhood cuts are more efficient in reinforcing the strengthened model.

Table 1: Runtime comparison (in seconds) with strengthening constraints

Instances	Initial	Symmetry	SINR	Symmetry & SINR
U5_D2	0,05	0,06	0,05	0,06
U5_D6	187,18	11,55	77,18	**10,71**
U5_D8	600,56	**45,16**	90,70	72,43
U10_D5	1677,21	**290,33**	697,49	435,08
U10_D7	8746.32	3569.18	4453.58	**2967.45**
U15_D7	10800*	10800*	10800*	10800*

4 TWO-PHASE HEURISTIC

Since solving the initial formulation has impractical runtime even for small instances, we propose a solving method based on a decomposition of DC problem into two subproblems: the *routing subproblem* and the *resource allocation subproblem*, that are to be solved separately. The objective of the first subproblem is to find an elementary path for each demand while minimizing the total link utilization costs. Then, the second subproblem provides a resource allocation to each active link obtained from the routing subproblem solution.

4.1 Routing subproblem

We propose two formulations for the routing subproblem: a compact formulation obtained by relaxing the resource assignment constraints (6) from (1)-(9), and a path reformulation.

4.1.1 Compact formulation. This formulation is the compact formulation obtained by relaxing the resource allocation constraints (6) from the formulation (1)-(9). The returned solution is a set of elementary paths for each request, respecting the capacities of the nodes along the paths. Two kinds of binary variables remains in the formulation : x_e^k that takes value 1 if the link $e \in A$ is used by the request $k \in K$ and the variables y_k^i. The objective is to minimize the total cost of active links:

$$min \sum_{k \in K} \sum_{e \in E} \mu_e^{s_k} x_e^k \tag{14}$$

Solving approach: The linear relaxation of this formulation is strengthened replacing **(5)** by:

$$x_e^k - y_i^k \leq 0 \qquad \forall\, k \in K, \forall\, (i,j) = e \in E, \tag{15}$$

$$x_e^k - y_j^k \leq 0 \qquad \forall\, k \in K, \forall\, (i,j) = e \in E. \tag{16}$$

and adding the following inequalities:

$$y_i^k (a_m^{s_k} - c_m^i) \leq 0 \quad \forall\, i \in V, \forall\, m \in C^d, \forall\, k \in K \backslash T(i). \tag{17}$$

The linear relaxation of this strengthened subproblem is solved using CPLEX. Then, a heuristic procedure is used to get a feasible integer solution. This approach is summarized in Algorithm 1. First, the linear relaxation is solved to optimality. Then, variables having an integer optimal value are fixed by updating the right hand side of the constraints (step 5). Finally, this residual ILP formulation is solved to optimality using CPLEX, giving rise to an integer solution for the whole problem. Step 11 is the rounding procedure on solution found by step 1. This heuristic is particularly efficient for this problem since most of the optimal variable values of the linear relaxation are integer.

4.1.2 Path formulation. The second formulation is a path formulation. We assume that all feasible paths P_k have been previously generated for each demand k. Thus, we define new binary variables x_p^k that take value 1 if the path $p \in P_k$ is used by

Table 2: Solution quality comparison between models with and without additional cuts

	Strengthened Formulation			Strengthened + Neighborhood cuts			Strengthened + Clique-based cuts			Strengthened + both cuts		
Instances	root gap (%)	runtime (s)	tree size	root gap (%)	runtime (s)	tree size	root gap (%)	runtime (s)	tree size	root gap (%)	runtime (s)	tree size
U5_D2	**0.00**	0.05	**1**	**0.00**	0.06	**1**	**0.00**	0.06	**1**	**0.00**	0.05	**1**
U5_D6	10.00	11.49	7	**10.00**	11.67	8	**10.00**	12.33	**2**	**10.00**	13.31	**2**
U5_D8	59.65	154.65	470	**23.58**	130.69	**291**	33.71	389.01	1225	32.00	150	599
U10_D5	57.89	677.21	1327	**33.09**	1370.90	190	48.23	1262.83	**91**	33.09	2265.83	**84**
U10_D7	62.26	3236.61	1850	**50.04**	8654.21	2040	62.26	3251.99	1850	50.04	9648.12	2040
*U15_D7**	77.23	10800	58	**72.37**	10800	8	77.23	10800	70	72.37	10800	22

Algorithm 1 LP-based Heuristic for the Routing subproblem

```
Solve LP (G, K)
Let FractionalDemands be the set of demands with associated optimal fractional variables
for each demand k do
    if all associated variable have integer value then
        update the LP by decreasing used capacities on vertices traversed by the associated paths
    else
        FractionalDemands : FractionalDemands ∪ {k}
    end if
end for
Solve ILP (G,FractionalDemands)
Update solution
Return solution found
```

the demand $k \in K$, 0 otherwise. The objective now is to minimize the sum of the active path weights:

$$min \sum_{k \in K} \sum_{p \in P_k} \mu_p^{s_k} x_p^k \tag{18}$$

where $\mu_p^{s_k}$ is the cost of path $p \in P_k$, defined as the sum of the weights on the arcs of p. Let α_p^e, respectively γ_p^i, be an indicator parameter with value 1 if the arc e, respectively the node i, is in the path p. The constraints of the path formulation are:

$$\sum_{p \in P_k} x_p^k = 1 \qquad \forall\, k \in K, \tag{19}$$

$$\alpha_p^e x_p^k \beta_{s_k} \leq sinr_e \qquad \forall\, k \in K, e \in A, \forall\, p \in P_k, \tag{20}$$

$$\sum_{k \in K \setminus T(i)} \sum_{p \in P_k} \gamma_p^i x_p^k a_m^{s_k} \leq c_m^i \qquad \forall\, i \in V, \forall\, m \in C^d, \tag{21}$$

$$x_p^k \in \{0, 1\} \qquad \forall\, k \in K, \forall\, p \in P_k. \tag{22}$$

Constraints (19) assure that each demand uses one path. Constraints (20) are the SINR constraints, and (21) are the capacity constraints.

Solving approach: This formulation, restricted to a subset of paths, is solved to optimality using CPLEX. The subsets of paths are generated thanks to the algorithm proposed by Yen [12]. This algorithm returns the K-shortest loopless paths for a graph with non-negative edge costs. It uses a shortest path algorithm as an intermediary to construct the whole solution. In our context, we use Dijkstra's algorithm [5], which has a good performance and a polynomial complexity. Using Yen's algorithm gives a guarantee to generate all the feasible paths due to the characteristics of our use case: for each demand, once a path uses the base station we are sure that all the feasible D2D paths have already been generated, then the path generation is stopped. This feature is due to greater weights on BS arcs. The difference between the weights of the BS links and the D2D ones plays an important role in the construction of the paths subsets: the greater this difference is, the longer it could be to obtain all the paths that use only D2D communication. Finally, to reduce the size of the formulation, a pre-processing on the SINR constraints is operated before solving both routing formulations: only the valid SINR links constraints are kept in the formulation.

4.2 Resource Allocation subproblem

The resource allocation subproblem consists in allocating the radio resources to each active link provided by the solution of the first subproblem.

4.2.1 Solving approach. Let $G^a(V^a, E^a)$ be a graph where each vertex in the set V^a represents an active link, that is, a link used by at least one path of the routing problem solution (steps 2-5 in Algorithm 2). An edge $e \in E^a$ is associated with a pair of vertices if the corresponding active links cannot share the same radio resource due to interference (steps 6-12). The objective is to assign a minimum number of colors (resources) to the vertices of the graph G^a, in such a way there is no adjacent vertices with the same color (step 14). This falls into a classical Vertex Coloring Problem [7]. The proposed heuristic has as input the initial graph of the problem, the values of the routing solutions and a criteria value which represents the minimum distance for a pair of links to have the same resource. This value was fixed to $D = 100$ meters so as to be representative of realistic use cases. Hence, for $dist()$ method, we calculate the distance between the opposite ends of the pair of links, that is, the origin of one with the destination of the other. The method returns *zero* for adjacent links. Finally, to find the coloring of the graph G^a (that is, allocate the resources to each active link) we use the *Coloring*() method which is the implementation of a classical greedy algorithm [9] for Vertex Coloring Problem. It is well known that the performance of this greedy algorithm is sensitive to the order of choice of the next vertex to be colored. For this reason, we randomly generate a large number of orders and choose the one that returns the minimum amount of colors. A lower bound value for the optimal solution is given by solving the associated Max-clique problem. For this

Algorithm 2 Resource Allocation subproblem

```
 1: Set $V^a$ and $E^a$ to empty sets
 2: for each activated link $e \in A$ do
 3:     Set Vertex $auxVertex.id \longleftarrow e.id$
 4:     Set $V^a : V^a \cup auxVertex$
 5: end for
 6: for each pair $(u, v) \in V^a$ do
 7:     if $dist(u, v) \leq criteria$ then
 8:         Set Link $auxLink.extremity1 \longleftarrow u$
 9:         Set Link $auxLink.extremity2 \longleftarrow v$
10:         Set $E^a : E^a \cup auxLink$
11:     end if
12: end for
13: Set graph $G^a(V^a, E^a)$
14: Coloring $(G^a)$
15: Return Resource Assignment
```

purpose we use the method proposed by [4], which is an exact approach using parallel programming.

4.3 Experiments and comparative results

The experiment conditions are the same as in section 3. We generated new instances, with 6 service domains available and the total amount of UEs is evenly divided between 7 core cells with one antenna installed on each one.

4.3.1 Routing subproblem. The numerical tests of the routing subproblem are summarized in table 3. The first column of the Compact Formulation part is the percentage of *active* variables that *are not* integers in the optimal LP solution. We note that it is always less than 6%. The second column is the gap between the solution found by the relaxation and the final solution found by the LP-based heuristic. The average gap is 2.30% with standard deviation equals to 3.85%. Finally, the third column is the total run time in seconds (pre-processing, relaxed solution and LP-based heuristic). In the second part of table 3, the pre-processing and solver run times of the path formulation are presented. The average number of paths generated in pre-processing is equal to 5.26 for each demand (with standard deviation equals to 2.39). Most of the runtime of the compact formulation based approach was spent in solving the linear relaxation of the problem. Given the very low number of fractional variables in the relaxed solution, the last IP on the residual formulation is quickly solved. It is also worth noticing that the pre-processing in path formulation is the most important step, since it is responsible for most of the solving time for all instances. However, its performance is relatively better than for the compact formulation, being on average 1.30 times faster. It is important to mention that the gap between the two formulations was always less than 0.50%.

Table 3: Routing subproblem: numerical tests

Instances	Compact formulation			Path Formulation	
	Act Frac Var (%)	Gap (%)	Total Runtime (s)	Pre processing (s)	Solver (s)
U700_D175	0.05	0.03	9.16	4.81	0.02
U700_D350	1.92	1.1	13.92	9.5	0.02
U700_D525	1.84	13	21.28	14.26	0.08
U700_D700	0.87	0.04	26.72	19.22	0.13
U1400_D350	5.22	0.02	46.83	27.48	0.8
U1400_D700	3.37	3.6	77.7	55.68	0.26
U1400_D1050	3.13	3.05	152.31	82.82	0.42
U1400_D1400	1.79	0.49	143.78	110.81	0.60
U2100_D525	5.56	0.02	143.28	81.28	0.23
U2100_D1050	3.54	0.09	257.18	163.39	0.65
U2100_D1575	2.15	3.86	242.84	230	0.8

Table 4: Resource Allocation subproblem: numerical tests

Instances	Pre-processing (s)	Greedy (s)	Gap (%)
U700_D175	0.13	0.89	0
U700_D350	0.53	1.81	0
U700_D525	1.15	2.75	0
U700_D700	2.12	3.89	0
U1400_D350	0.54	1.74	0
U1400_D700	2.34	3.76	0
U1400_D1050	4.97	5.95	0
U1400_D1400	7.06	6.97	0
U2100_D525	1.32	2.07	0
U2100_D1050	4.45	5.25	0
U2100_D1575	11.21	9.51	0

4.3.2 Resource allocation subproblem. Table 4 shows the results for the resource allocation subproblem, where the *Pre-processing* column represents the time needed to transform the solution from the previous subproblem into a classic graph and find its max-clique. We can observe that even though they are extremely large graphs, the time needed to find the final solution is relatively short. This is due to the characteristics of the topology constructed from the assumptions and hypotheses previously presented. Generated graphs have an average density equals to 64.05% - standard deviation equals to 1.98%. Another important result emphases is that in all cases, we have found the optimal solution, proven by the lower bound value previously calculated by the exact max-clique algorithm (last column).

5 CONCLUDING REMARKS

In this paper, we have studied the Domain Creation problem, that is a routing and resource assignment problem arising in future 5G networks. We have proposed two algorithms: exact and heuristic, to solve it. The exact approach is based on a node-arc ILP formulation enhanced by two families of valid inequalities that are used within a branch-and-cut framework. The preliminary results show a significant impact of the cuts in strenghthening the LP relaxation and reducing the computation time. We expect that adding further classes of cuts and performing an analysis to find out the specificities of difficult instances (regardless of their size) will allow to solve even larger instances. A natural question would be to consider a non-compact formulation, based on path variables and propose a column generation based algorithm to solve it. On an other hand, our experiments show that the heuristic approach performs well, even on large instances with up to 2100 devices and 1500 service requests on 7-cell networks. It would be interesting and most probably very powerful to use it as a primal heuristic to boost efficiency of an exact algorithm. On a practical note, a tough but interesting extension is to include users mobility or temporal aspect in radio resource assignment.

ACKNOWLEDGMENTS

This work is supported by the french Agence Nationale de la Recherche (ANR), Project MAESTRO-5G ANR-18-CE25-0012.

REFERENCES

[1] K. I. Aardal, A. Hipolito, S. P. M. van Hoesel, and B. Jansen. 1996. A branch-and-cut algorithm for the frequency assignment problem. *Research Memorandum 96/011, Maastricht University* (1996).

[2] K. I. Aardal, S. P. M. van Hoesel, A. M. C. A. Koster, C. Mannino, and A. Sassano. 2007. Models and solution techniques for frequency assignment problems. *Annals of Operations Research* 153, 1 (01 Sep 2007), 79–129.

[3] Arash Asadi, Qing Wang, and Vincenzo Mancuso. 2014. A Survey on Device-to-Device Communication in Cellular Networks. *IEEE Communications Surveys and Tutorials* 16, 4 (2014), 1801–1819.

[4] Matjaz Depolli, Janez Konc, Kati Rozman, Roman Trobec, and Dusanka Janezic. 2013. Exact parallel maximum clique algorithm for general and protein graphs. *Journal of chemical information and modeling* 53, 9 (2013), 2217–2228.

[5] Edsger W Dijkstra. 1959. A note on two problems in connexion with graphs. *Numerische mathematik* 1, 1 (1959), 269–271.

[6] B. Jaumard, C. Meyer, and B. Thiongane. 2006. *ILP Formulations for the Routing and Wavelength Assignment Problem: Symmetric Systems.* Springer US, 637–677.

[7] Enrico Malaguti and Paolo Toth. 2010. A survey on vertex coloring problems. *International transactions in operational research* 17, 1 (2010), 1–34.

[8] François Margot. 2010. *Symmetry in Integer Linear Programming.* 647–686.

[9] David W Matula, George Marble, and Joel D Isaacson. 1972. Graph coloring algorithms. (1972), 109–122.

[10] G. L. Nemhauser and G. Sigismondi. 1992. A Strong Cutting Plane/Branch-and-Bound Algorithm for Node Packing. *Journal of the Operational Research Society* 43, 5 (1992), 443–457.

[11] A. E. Ozdaglar and D. P. Bertsekas. 2003. Routing and Wavelength Assignment in Optical Networks. *IEEE/ACM Trans. Netw.* 11, 2 (April 2003), 259–272.

[12] Jin Y Yen. 1970. An algorithm for finding shortest routes from all source nodes to a given destination in general networks. *Quart. Appl. Math.* 27, 4 (1970), 526–530.

[13] M. Zulhasnine, C. Huang, and A. Srinivasan. 2010. Efficient resource allocation for device-to-device communication underlaying LTE network. *2010 IEEE 6th International conference on wireless and mobile computing, networking and communications* (2010), 368–375.

Pooling Problems with Single-Flow Constraints

D. Haugland
Department of Informatics, University of Bergen
Bergen, Norway
dag@ii.uib.no

ABSTRACT

The pooling problem is a frequently studied extension of the traditional minimum cost flow problem, in which the composition of the flow is subject to restrictions. In a network consisting of three layers of nodes, the composition is given at the source layer. In the intermediate nodes, referred to as pools, the composition is a weighted average of the compositions in entering flow streams. The same is true at the sink layer, where upper bounds on the concentration of each component apply. Motivated by practical applications, and needs for heuristic methods for the standard pooling problem, the current work focuses on pooling problems where the flow graph is restricted to satisfy certain sparsity conditions. We consider in particular the requirements that each pool receives flow from at most one neighboring source, or sends flow to at most one neighboring sink. We prove that the pooling problem remains NP-hard after this and other similar extensions. It is also demonstrated how the single-flow constrained extensions can be modeled by means of mixed integer linear programming (MILP), without introducing bilinear terms. We also show that such MILP-models are useful for computing good feasible solutions to the original problem.

KEYWORDS

Network flow, pooling problem, mixed integer programming

1 INTRODUCTION

Most network flow models are built upon considerations of the flow as a homogeneous commodity. In many industrial settings, it is however essential to reflect variation in composition that may occur across the network. Contamination levels of crude oils supplied to a refinery depend on their sources of origin. Proportions in which major components of natural gas, such as methane, ethane, butane and propane, occur are not equal for all gas wells. For environmental or technical reasons, requirements to the final flow composition can be imposed at the reception points of the flow network. In applications of this kind, it is therefore crucial for network flow models to recognize not only the total flow, but also how the flow composition evolves from network sources to sinks. Updates of the composition at nodes where differently composed flow streams are pooled must be reflected by the models. A result of this is a computationally challenging problem referred to as the *pooling problem*.

The pooling problem resembles well-studied logistics models like the minimum cost flow problem and the transportation problem. While a bipartite network structure means that the problem can be modeled in terms of linear programming (LP), bilinear formulations appear to be inevitable when the network has three layers of nodes. While large instances of the minimum cost flow problem, with arbitrary network topology, can be solved fast, exact solution of pooling problem instances with much fewer nodes appears to be unrealistic.

Already decades ago, Haverly [19] recognized the pooling problem as a challenge where linear programming approaches may fail. Because of the anticipated intractability of the problem, early research was mainly directed towards heuristic methods [6, 10, 19], where iterative linearization is the core idea. Floudas and Visweswaran [13] reported the first exact solution algorithm for the pooling problem. Later, algorithms based on branch-and-bound [5, 14, 25–27], Lagrangian relaxation [1, 4, 8], particle swarm optimization [12], integer programming [11, 16], and semi-definite programming [22] have been studied. The industrial relevance of the problem, notably in petroleum refining [6, 10], the food industry [20], and in waste water processing [15, 24], has been acknowledged by many authors. The pooling problem survey by Misener and Floudas [23] has a comprehensive list of references to work in this area.

Standard pooling problems are defined as networks with three node layers, referred to as sources, pools and sinks, respectively. Arcs connect either a source to a pool or a pool to a sink. Quality constraints in terms of bounds on the composition are imposed at the sinks. Each bound relates to the relative content of a flow component, referred to as the quality. Even the class of instances with a unique pool is strongly NP-hard [2]. The computational complexity is however favorable if there are additional limitations on node cardinalities, as polynomial time algorithms have been developed for the case of a unique pool and an upper bound on either the number of sinks [17] or the number of quality constraints at each sink [2]. Polynomial running-time algorithms for the one-pool instance class also exist when the number of sources is bounded [9, 18].

The pooling problem remains NP-hard for networks with only two sources and two sinks, and only one flow component subject to constraints [17]. Such instances can be solved in polynomial time when the input data take values within given bounds [18]. While the algorithms with polynomial running time mentioned this far are based on LP, Baltean-Lugojan and Misener [7] prove that also strongly polynomial solution algorithms exist for a wide range of network topologies.

In certain applications [21], physical restrictions disallow flow along more than one arc entering or leaving a pool. While the pools may have multiple entering and leaving arcs in the flow network, decisions must be made as to which one of these to apply. This problem can be rephrased as a bilevel problem in the realm of network design: Find a subnetwork, such that each pool has only one entering arc and/or only

ISBN 978-3-89318-079-0 on OpenProceedings, ISSN 2510-7437

DOI: 10.5441/002/inoc.2019.18

one leaving arc, and solve the pooling problem on the selected subgraph. In this context, it becomes relevant that the pooling problem can be formulated as a compact LP if each pool has either in-degree or out-degree equal to one [17, Proposition 3]. By introducing design variables for each arc, the said extension is formulated as a mixed integer linear program (MILP), and, contrary to the standard pooling problem, bilinear constraints are easily avoided.

Extensions where restrictions on the number of flow-carrying arcs incident to each pool are also interesting from a computational point of view. The feasible region of the extended version is obviously a subset of its counterpart in the original version. In instances where the optimal solutions to the original and revised problems are not too far apart, the latter problem may serve as a close approximation to the former. If the more restricted version of the problem is solved with sufficiently smaller computational burden than what is the case of the original, it may thus be a key to effective inner approximation of the standard pooling problem.

In the current work, we consider four different variants of the pooling problem with constraints on the number of active arcs (Section 2). The contributions made to the pooling problem literature includes proofs of the NP-hardness of each of the new problems (Section 3). Further, we give MILP-formulations for the problems, along with valid inequalities and lifting procedures for strengthening the relaxations of the formulations (Sections 4–5). Preliminary experiments are reported (Section 6), demonstrating that the inner approximation idea enables improvements of the best known solution to four instances of the standard pooling problem.

2 NOTATION AND PROBLEM DEFINITION

Let $D = (N, A)$ be a directed acyclic graph with the node set N partitioned into the sets S of sources, P of pools, and T of sinks. The arc set $A = A_S \cup A_T$ is partitioned into $A_S \subseteq S \times P$ and $A_T \subseteq P \times T$, connecting sources with pools and pools with sinks, respectively. Thus, $H = \{(s,p,t) \in S \times P \times T : (s,p),(p,t) \in A\}$ is the set of directed paths in D. Let K be a finite set, the elements of which are referred to as *qualities*. For each node $i \in N$, define the upper flow bound b_i, let $b_{ij} = \min\{b_i, b_j\}$ for each arc $(i,j) \in A$, and let $b_{spt} = \min\{b_s, b_p, b_t\}$ for each path $(s,p,t) \in H$. Further, associate the unit cost c_{ij} with arc (i,j). For each quality $k \in K$, we introduce parameters for each source and each sink. Let q_s^k be the *concentration* of quality k at source s, and let q_t^k be the *upper bound on the concentration* of quality k at sink t.

For each $s \in S$, $p \in P$, and $t \in T$, we define the *neighbor sets* $P_s = \{p \in P : (s,p) \in A\}$, $S_p = \{s \in S : (s,p) \in A\}$, $T_p = \{t \in T : (p,t) \in A\}$, and $P_t = \{p \in P : (p,t) \in A\}$. Let $F(D,b) \subseteq \mathbb{R}_+^A$ be the *flow polytope* associated with D and b. That is, $x \in F(D,b)$ means that x is a vector with components x_{ij} corresponding to the arcs (i,j), satisfying the *capacity constraints* $\sum_{p \in P_s} x_{sp} \leq b_s$ $(s \in S)$, $\sum_{p \in P_t} x_{pt} \leq b_t$ $(t \in T)$, and $\sum_{s \in S_p} x_{sp} \leq b_p$ $(p \in P)$, and the *flow conservation constraints* $\sum_{s \in S_p} x_{sp} = \sum_{t \in T_p} x_{pt}$ $(p \in P)$. Any flow $x \in F(D,b)$ induces, for every $k \in K$, a concentration w_i^k at node $i \in N$. In the case of a source s, $w_s^k = q_s^k$. For nodes $j \in P \cup T$, the concentration is a solution to

$$w_j^k \sum_{i \in N:(i,j) \in A} x_{ij} = \sum_{i \in N:(i,j) \in A} w_i^k x_{ij}, \qquad (1)$$

reflecting the assumption that k represents a chemical compound, the concentration of which blends linearly when heterogeneous flow streams meet.

Definition 2.1. The Standard Pooling Problem amounts to finding a flow $x \in F(D,b)$ inducing a concentration $w \in \mathbb{R}^{N \times K}$ satisfying $w_t^k \leq q_t^k$ for each $t \in T$ and each $k \in K$, such that $\sum_{(i,j) \in A} c_{ij} x_{ij}$ is minimized.

In the remainder of the paper, we mainly focus on extensions of the Standard Pooling Problem, where additional constraints on the number of flow streams leaving/entering a pool are imposed.

Definition 2.2. Each of the following problems are identified by a set of constraints in addition to those applying to Definition 2.1:

- The Single-In Pooling Problem: For all $p \in P$, $x_{sp} > 0$ for at most one $s \in S_p$.
- The Single-Out Pooling Problem: For all $p \in P$, $x_{pt} > 0$ for at most one $t \in T_p$.
- The Single-In-And-Out Pooling Problem: For all $p \in P$, $x_{sp} > 0$ for at most one $s \in S_p$, and $x_{pt} > 0$ for at most one $t \in T_p$.
- The Single-In-Or-Out Pooling Problem: For all $p \in P$, $x_{sp} > 0$ for at most one $s \in S_p$, or $x_{pt} > 0$ for at most one $t \in T_p$.

3 COMPLEXITY

For all the single-flow constrained problems introduced in the previous section, the following observations are made: With knowledge to the sources (sinks) from (to) which the single flows enter (leave) a pool, the remaining problem can be solved in terms of a compact Linear Program (LP) [17, Proposition 3]. However, the problems are in general intractable.

Proposition 3.1. *The problems given in Definition 2.2 are NP-hard.*

Proof. There exists [17, Theorem 6] a polynomial reduction from the Maximum 2-Satisfiability Problem to an instance class of the Standard Pooling Problem, in which all feasible solutions satisfy the constraints of the Single-Out Pooling Problem, and thereby also the Single-In-Or-Out Pooling Problem. It follows that the latter two problems are NP-hard. Analogously, a polynomial reduction [17, Theorem 7] from the Minimum 2-Satisfiability Problem proves the NP-hardness of the Single-In Pooling Problem. That also the Single-In-And-Out Pooling Problem is NP-hard, is proved by the following reduction from the Partition Problem: Let $a_1, \ldots, a_n \in \mathbb{Z}_+$. Consider the instance of the Single-In-And-Out Pooling Problem where $S = \{s_1, \ldots, s_n\}$, $P = \{p_1, \ldots, p_n\}$, $T = \{t_0, t_1\}$, $A_T = P \times T$, $A_S = \{(s_i, p_i)\}_{i=1}^n$, $K = \emptyset$, $b_{s_i} = b_{p_i} = a_i$ $(i = 1, \ldots, n)$, $b_{t_0} = b_{t_1} = \frac{1}{2} \sum_{i=1}^n a_i$, $c_{s_i p_i} = -1$, and $c_{pt} = 0$ for $(p,t) \in A_T$. It follows that $\{a_1, \ldots, a_n\}$ is a yes-instance to the Partition Problem if and only if the minimum cost in the corresponding instance of the Single-In-And-Out Pooling Problem is $-\sum_{i=1}^n a_i$. □

4 MIXED INTEGER PROGRAMMING MODELS

All problems introduced in Definition 2.2 are formulated in terms of continuous variables representing path flow, and binary variables representing selection of arcs to carry the flow. In the models that follow, the flow x_{ij} along arc (i,j) is not represented by a dedicated variable, but it is available by summation of all flow variables corresponding to paths containing (i,j). In all models, x_{spt} denotes the flow along path $(s,p,t) \in H$. To model the Single-In-Or-Out Pooling Problem, let y be a binary vector over the arcs in D. For any arc $(s,p) \in A_S$, pool p receives flow uniquely along (s,p) if $y_{sp} = 1$. Analogously, if $y_{pt} = 1$, then pool p sends flow uniquely along arc $(p,t) \in A_T$. Letting H_i denote the set of paths intersecting node $i \in N$, this leads to the formulation:

$$\min_{x,y} \quad \sum_{(s,p,t)\in H} (c_{sp} + c_{pt})\, x_{spt} \tag{2}$$

$$\text{s.t.} \quad \sum_{(s,p,t)\in H_i} x_{spt} \le b_i \qquad i \in N \tag{3}$$

$$\sum_{p\in P_t} \sum_{s\in S_p} \left(q_s^k - q_t^k\right) x_{spt} \le 0 \qquad t \in T, k \in K \tag{4}$$

$$\sum_{s\in S_p} y_{sp} + \sum_{t\in T_p} y_{pt} = 1 \qquad p \in P \tag{5}$$

$$x_{spt} \le b_{spt}\,(y_{sp} + y_{pt}) \qquad (s,p,t) \in H \tag{6}$$

$$x \in \mathbb{R}_+^H, y \in \{0,1\}^A \tag{7}$$

As arc flow is replaced by path flow, there is no need for flow conservation constraints. Thus, the capacity constraints (3) ensure that only solutions in $F(D,b)$ are feasible. Because the concentration of quality $k \in K$ at sink t equals $\sum_{p\in P_t}\sum_{s\in S_p} q_s^k x_{spt} / \sum_{p\in P_t}\sum_{s\in S_p} x_{spt}$, constraints (4) impose the upper bound q_t^k on the concentration. Finally, flow on at most one arc entering pool $p \in P$, or at most one arc leaving p, is achieved by (5)–(6).

By addition of $y_{pt} = 0$ ($(p,t) \in A_T$) and $y_{sp} = 0$ ($(s,p) \in A_S$), respectively, (2)–(7) also becomes a formulation of the Single-In Pooling Problem and the Single-Out Pooling Problem.

The Single-In-And-Out Pooling Problem is formulated in terms of the binary path selection variables y_{spt} ($(s,p,t) \in H$). The objective is to minimize (2) subject to (3)–(4) and

$$\sum_{(s,p,t)\in H_p} y_{spt} = 1 \qquad p \in P \tag{8}$$

$$x_{spt} \le b_{spt} y_{spt} \qquad (s,p,t) \in H \tag{9}$$

$$x \in \mathbb{R}_+^H, y \in \{0,1\}^H \tag{10}$$

5 STRENGTHENING THE FORMULATIONS

This section gives some simple techniques for strengthening the continuous relaxations of the MILP-formulations. First, observe that for pools with only one entering or one leaving arc, the y-variables and corresponding constraints are not needed.

Observation 1. *Deletion of variables y_{sp} ($s \in S_p$) and y_{pt} ($t \in T_p$), as well as constraints (5)–(6), for all $p \in P$ such that $\min\{|S_p|, |S_p|\} = 1$, does not alter the optimal solution to (2)–(7).*

5.1 Lifted Inequalities

By a *maximum flow instance* of (2)–(7), we mean an instance in which $c_{p\bar{t}} = -1$ for a unique sink $\bar{t} \in T$ and all neighboring pools $p \in P_{\bar{t}}$, whereas $c_{ij} = 0$ for all other arcs $(i,j) \in A$. That is, the problem is to maximize the flow entering $\bar{t}$, subject to the imposed constraints. Analogously, if all arcs leaving a given source $\bar{s} \in S$ have cost -1, whereas other costs are zero, we face a maximum flow instance corresponding to source $\bar{s}$.

When the inducing node is a sink, the maximum flow instance is particularly easy to solve:

Proposition 5.1. *Any maximum flow instance of (2)–(7) corresponding to $\bar{t} \in T$ has an optimal solution (x,y) where $y_{p\bar{t}} = 1$ for all $p \in P_{\bar{t}}$, and $x_{spt} = 0$ for all $(s,p,t) \in H$ where $t \neq \bar{t}$.*

Proof. Let (x,y) be a feasible solution to (2)–(7). Assume that $\sum_{(s,p,\hat{t})\in H_{\hat{t}}} x_{sp\hat{t}} > 0$ for some sink $\hat{t} \neq \bar{t}$. Then, for a sufficiently small $\delta > 0$, also (x', y), where $x'_{sp\hat{t}} = (1-\delta)x_{sp\hat{t}}$ for all $(s,p,\hat{t}) \in H_{\hat{t}}$ and $x'_{spt} = x_{spt}$ for $(s,p,t) \in H \setminus H_{\hat{t}}$, is also feasible. Further, the objective function value at (x',y) is identical to the one at (x,y). For the largest such δ, $x'_{sp\hat{t}} = 0$ for some $(s,p,\hat{t}) \in H_{\hat{t}}$. It follows by induction that (2)–(7) has an optimal solution where $\bar{t}$ is the sole sink to receive non-zero flow. In such a solution, it is optimal to assign the value 1 to $y_{p\bar{t}}$ for all $p \in P_{\bar{t}}$, which completes the proof. □

The tractability of sink-induced maximum flow instances is contrasted by their source-induced counterparts:

Proposition 5.2. *The* Single-In-Or-Out Pooling Problem *is NP-hard for maximum flow instances corresponding to a source.*

Proof. The proof is by reduction from the Partition Problem: Let $a_1, \ldots, a_n \in \mathbb{Z}_+$. Consider the instance of the Single-In-Or-Out Pooling Problem where $S = \{s_1^+, \ldots, s_n^+, s_1^-, \ldots, s_n^-, \bar{s}\}$, $P = \{p_1, \ldots, p_n, \bar{p}\}$, $T = \{t_0, t_1\}$, $A_T = P \times T$, $A_S = \{(s_i^+, p_i), (s_i^-, p_i)\}_{i=1}^n \cup \{(\bar{s}, \bar{p})\}$, $K = \{k\}$, $b_{s_i^+} = b_{s_i^-} = a_i$, $b_{p_i} = 2a_i$ ($i = 1, \ldots, n$), $b_{\bar{s}} = b_{\bar{p}} = 2\sum_{i=1}^n a_i$, $b_{t_0} = b_{t_1} = 2\sum_{i=1}^n a_i$, $q_{s_i^+}^k = q_{s_i^-}^k = 0$ ($i = 1, \ldots, n$), $q_{\bar{s}}^k = 1$, $q_{t_0}^k = q_{t_1}^k = \frac{1}{2}$, $c_{\bar{s}\bar{p}} = -1$, and $c_{ij} = 0$ for all $(i,j) \in A \setminus \{(\bar{s}, \bar{p})\}$.

The quality constraints at sinks t_0 and t_1 ensure that the flow along arc $(\bar{s}, \bar{p})$ is at full capacity $2\sum_{i=1}^n a_i$ only if both sinks receive $\sum_{i=1}^n a_i$ flow units from $S \setminus \{\bar{s}\}$. Then the flow along the arcs entering $p_1, \ldots, p_n$ are at full capacity. From the single-flow constraints, it follows that each pool $p_1, \ldots, p_n$ delivers flow to exactly one sink. Hence, $(a_1, \ldots, a_n)$ is a yes-instance if and only if the maximum flow leaving $\bar{s}$ is $2\sum_{i=1}^n a_i$. □

Observation 2. *If (x,y) is an optimal solution to (2)–(7), then*

$$\sum_{p\in P_t} \sum_{s\in S_p} (c_{sp} + c_{pt})\, x_{spt} \le 0 \tag{11}$$

for all $t \in T$.

PROOF. Assume (11) is violated for some $t' \in T$. Define $x' \in \mathbb{R}_{+}^{H}$ such that $x'_{spt'} = 0$ ($p \in P_{t'}$, $s \in S_p$) and $x'_{spt} = x_{spt}$ ($t \in T \setminus \{t'\}$, $p \in P_t$, $s \in S_p$). Then, (x', y) is feasible, and $\sum_{(s,p,t)\in H} (c_{sp} + c_{pt})\, x'_{spt} < \sum_{(s,p,t)\in H} (c_{sp} + c_{pt})\, x_{spt}$, contradicting the optimality assumption. □

It follows from Observation 2 that, for any $(\bar{s}, \bar{p}, \bar{t}) \in H$, we can lift inequality (6) to

$$x_{\bar{s}\bar{p}\bar{t}} \leq \alpha_{\bar{s}\bar{p}\bar{t}} y_{\bar{s}\bar{p}} + \beta_{\bar{s}\bar{p}\bar{t}} y_{\bar{p}\bar{t}},$$

where $\alpha_{\bar{s}\bar{p}\bar{t}}$ and $\beta_{\bar{s}\bar{p}\bar{t}}$ are upper bounds on the optimal flow along $(\bar{s}, \bar{p}, \bar{t})$ under the mutually exclusive conditions $y_{\bar{s}\bar{p}} = 1$ and $y_{\bar{p}\bar{t}} = 1$, respectively. The latter bound is identified by the linear program

$$\begin{aligned}
\beta_{\bar{s}\bar{p}\bar{t}} \quad & = \max_{x} x_{\bar{s}\bar{p}\bar{t}} & & (12)\\
\text{s.t.} \quad & \sum_{p \in P_s \cap P_{\bar{t}}} x_{sp\bar{t}} \leq b_s & s \in S \quad & (13)\\
& \sum_{s \in S_p} x_{sp\bar{t}} \leq b_p & p \in P_{\bar{t}} \quad & (14)\\
& \sum_{p \in P_{\bar{t}}} \sum_{s \in S_p} \left(q_s^k - q_{\bar{t}}^k\right) x_{sp\bar{t}} \leq 0 & k \in K \quad & (15)\\
& \sum_{p \in P_{\bar{t}}} \sum_{s \in S_p} \left(c_{sp} + c_{pt}\right) x_{sp\bar{t}} \leq 0 & & (16)\\
& x \in \mathbb{R}_{+}^{H_{\bar{t}}}, & & (17)
\end{aligned}$$

while $\alpha_{\bar{s}\bar{p}\bar{t}}$ is the optimal objective function value to the same LP, with the additional constraints that $x_{s\bar{p}\bar{t}} = 0$ for all $s \in S_{\bar{p}} \setminus \{\bar{s}\}$.

Recently, a procedure for eliminating sinks at which the quality constraints (4) can be met only by the zero flow, has been suggested [11, Observation 1]. The above lifting techniques is built upon analogous principles, and has a corresponding elimination effect since $\beta_{sp\bar{t}} = 0$ for all $(s, p, \bar{t}) \in H_{\bar{t}}$ if (4) is too strict at $\bar{t}$. By virtue of the profitability condition (16), however, the lifting procedure is capable of eliminating more sink nodes, and it is consequently more selective than [11].

A stronger relaxation of the formulation for the SINGLE-IN-AND-OUT POOLING PROBLEM is obtained by lifting constraint (9) to

$$x_{spt} \leq \alpha_{spt} y_{spt} \quad (s, p, t) \in H.$$

5.2 Valid Inequalities

Because the flow along arc (s, p) cannot exceed b_{sp}, and because it is non-zero only if y_{sp} or $\sum_{t \in T_p} y_{pt}$ equals one, the following inequalities are valid in all problems (the SINGLE-IN-AND-OUT POOLING PROBLEM disregarded):

$$\sum_{t \in T_p} x_{spt} \leq b_{sp} \left(y_{sp} + \sum_{t \in T_p} y_{pt} \right) \quad (s, p) \in A_S \quad (18)$$

$$\sum_{s \in S_p} x_{spt} \leq b_{pt} \left(y_{pt} + \sum_{s \in S_p} y_{sp} \right) \quad (p, t) \in A_T \quad (19)$$

The arguments leading to (19) are analogous to those yielding (18).

When the capacities at the sinks T_p are large compared with capacities b_s and b_p, (18) becomes particularly effective. In the extreme case, when $\min\{b_t : t \in T_p\} \geq b_{sp}$, we have $b_{spt} = b_{sp}$ for all $t \in T_p$. Summating (6) over all $t \in T_p$ then yields

$$\sum_{t \in T_p} x_{spt} \leq |T_p|\, b_{sp} y_{sp} + b_{sp} \sum_{t \in T_p} y_{pt},$$

which obviously is weaker than (18). Analogously, (19) becomes effective when b_s ($s \in S_p$) is large compared with b_p and b_t.

A valid formulation of the SINGLE-IN-OR-OUT POOLING PROBLEM is obtained if (6) is replaced by (18)–(19). However, in the continuous relaxations of the formulations, inequalities (18)–(19) and constraints (6) complement rather than replace each other. This is seen by observing that for fractional values of y, (18)–(19) do not necessarily imply (6).

Inequality (19) is lifted to

$$\sum_{s \in S_p} x_{spt} \leq \sigma_{pt} y_{pt} + \sum_{s \in S_p} \alpha_{spt} y_{sp} \quad (p, t) \in A_T,$$

in a way analogously to what is outlined in Section 5.1. The upper bound $\sigma_{\bar{p}\bar{t}}$ on the optimal flow along a given arc $(\bar{p}, \bar{t}) \in A_T$, is given by the maximum value of $\sum_{s \in S_{\bar{p}}} x_{s\bar{p}\bar{t}}$, subject to constraints (13)–(17).

According to Proposition 5.1, maximizing the flow along an arc entering a sink does not involve consideration of other sinks. Consequently, the LP (12)–(17) has variables corresponding exclusively to paths in $H_{\bar{t}}$. Unfortunately, an analogous network reduction is not achieved when the maximum flow $\sigma_{\bar{s}\bar{p}}$ along $(\bar{s}, \bar{p}) \in A_S$ is to be maximized. Proposition 5.2 suggests that lifting of inequality (18) analogously to the lifting of (19) is considerably more expensive, and computing σ_{sp} is unlikely to be worth the computational cost.

With no efforts beyond those required in the lifting of (6) and (19), (18) is however lifted to

$$\sum_{t \in T_p} x_{spt} \leq b_{sp} y_{sp} + \sum_{t \in T_p} \beta_{spt} y_{pt} \quad (s, p) \in A_S.$$

In the SINGLE-IN-AND-OUT POOLING PROBLEM, the valid inequalities

$$\sum_{t \in T_p} x_{spt} \leq b_{sp} \sum_{t \in T_p} y_{spt} \quad (s, p) \in A_S$$

become effective when the sinks have relatively large capacities.

6 PRELIMINARY EXPERIMENTS

Feasible solutions to any of the problems of Definition 2.2 are also feasible in the STANDARD POOLING PROBLEM. Solution algorithms for the single-flow constrained problems, possibly with time interruption, can thus be considered as heuristic methods for the standard version of the problem. This section reports some preliminary experiments where this approach is benchmarked against other heuristics that recently have been analyzed in the literature.

6.1 Test Instances and Experimental Setup

The SINGLE IN-OR-OUT POOLING PROBLEM is the variant which preserves the largest part of the feasible region in the standard problem. Therefore, our experiments amount to submitting instantiations of model (2)–(7), with the addition of the valid inequalities (18)–(19), to a generic MILP-solver. Twenty publicly available benchmark instances are considered, each of which has previously [3, 11, 16] been analyzed in studies of heuristics for the Standard Pooling Problem. Dey and Gupte [11] report extensive experiments on 50 additional randomly generated instances, to which we do not have access. They compare variants of their MIP-techniques, based on discretization of the solution set, with various heuristics. Amongst these is the time-interrupted application of a generic global solver (BARON). As detailed reports on the solutions produced by all investigated methods are provided [11], the capabilities of the current approach to generate good feasible solutions is benchmarked against these.

The test instances are partitioned into three groups, where the node and quality cardinalities, $|S|$, $|P^*|$, $|T|$, and $|K|$ are constant within each group. Here, $P^* = \{p \in P : \max\{|S_p|, |T_p|\} > 1\}$ is the set of pools with more than one incident arc on at least one side. In instances $\mathrm{A0}, \ldots, \mathrm{A9}$, we have $|S| = 20$, $|P^*| = 10$, $|T| = 15$, and $|K| = 12$, in instances $\mathrm{B0}, \ldots, \mathrm{B5}$, $|S| = 35$, $|P^*| = 17$, $|T| = 21$, and $|K| = 17$, and in instances $\mathrm{C0}, \ldots, \mathrm{C3}$, $|S| = 60$, $|P^*| = 30$, $|T| = 40$, and $|K| = 20$. More details about the instances are given in [3]. Henceforth, this set of instances is denoted I.

To solve the MILP-instances, CPLEX (version 12.5.1.0) is used. Time bounds of 30 CPU-minutes (instances A0–A9 and B0–B5) and 60 CPU-minutes (instances C0–C3) are imposed. All runs are made on a Linux machine (64-bit) with two x86-processors (2.40 GHz) and 1.9GB of RAM.

6.2 Numerical Results

Following [11], a summary of the performance of the approach under study is given in terms of performance profiles. Let M be the set consisting of the 13 methods compared in [11], in addition to the current one. For each $m \in M$, let $z(m, i)$ be the cost of the solution that method m produced in instance $i \in I$, and define the corresponding score $\eta(m, i) = \frac{z(m,i) - \min\{z(n,i) : n \in M\}}{\max\{z(n,i):n \in M\} - \min\{z(n,i):n \in M\}}$. That is, $\eta(m, i) \in [0, 1]$, with lower values indicating better performance. A point (κ, γ) intersected by the performance profile of m tells that there exist $|I|\gamma$ instances i (but not more), in which $\eta(m, i)$ is no more than κ. Hence, higher profiles indicate stronger performance than lower ones.

Performance profiles obtained from previously reported experiments [11] are depicted in Fig. 1. Additionally, the red dashed profile represents the performance of the approach taken in the current work. We observe that for small values of κ, the red profile is dominated by the blue dotted profile, which represents the performance of BARON when assigned a time bound of 60 CPU minutes. This reflects the fact that BARON more often (in 9 instances) than the current method (in 6 instances) is the best-performing method. However, in the larger instances, the global solver struggles to find good feasible solutions, and finds only the zero solution in three of them. Modest growth in the corresponding profile is accordingly observed. Further experiments [11] focused on larger instances, demonstrate that BARON gets outperformed by the MILP techniques introduced in [11].

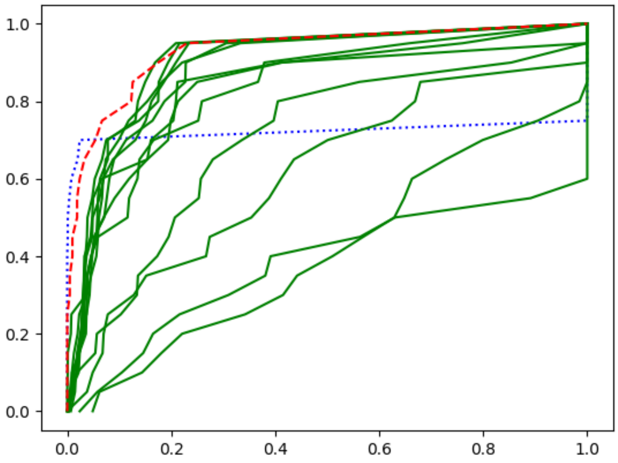


Figure 1: Performance profile of the current IP-approach (red dashed line) matched with BARON (blue dotted line), and other methods (green solid lines) from [11]

A feature of the solution approach analyzed in the current work is that only *sparse* solutions, in the sense of Definition 2.2, are considered. The high positions of the corresponding profile in Fig. 1 suggests that, in the instances under study, there exist near-optimal solutions to the Standard Pooling Problem featuring sparsity. At worst (i = A4), the score is $\eta(m, i) = 0.23$ (m denoting the current method). The largest optimality gap, relative to the lower bounds computed by BARON within one CPU hour [11], is in no instance above 17%. In one instance (A9), optimality in the Standard Pooling Problem is proved. The strength of the approach appears to be good worst-case performance, as only the profile of the method $\mathbb{A}(4)$ [11] has higher positioned points beyond $\kappa = 0.02$. In four of the instances (B3, B4, B5, and C2), the approach under study finds better solutions to the Standard Pooling Problem than the previously best known, reported in [11, 16].

Four of the test instances (A0–A3) are solved to integer optimality in less than 20 CPU seconds, three more (A4, A7, and A9) are solved in less than 7 CPU minutes, and another two instances (A5 and B1) are solved in less than 12 CPU minutes. In all the remaining 11 instances (A6, A8, B0, B2–B5, and C0–C3), the solver is interrupted because the time limit (30 and 60 CPU minutes, respectively) is reached. Upon interruption, the remaining relative optimality gap is below 1% in four instances (A6, A8, B4, and B5), and at most 17% (instance C1).

7 CONCLUSIONS

Although much progress on solution algorithms for the Standard Pooling Problem has been made over the last decade, it is still to be judged as a considerably difficult problem to solve. Restricted versions of the problems introduced in the current text are also shown to be NP-hard. Unlike their parent problem, however, the single-flow constrained pooling problems admit very natural MILP-formulations. By virtue of this, powerful MILP-solvers can provide non-trivial feasible solutions, at least in instances of modest size.

The Single-In-Or-Out Pooling Problem, which has received most of the attention in this work, has a potential to serve as an inner approximation of the standard problem. Some progress towards strong MILP-formulations for the problem has been made, and preliminary computational tests are encouraging. In the instances tested in the current work, the sparse solutions obtained are good approximations of the optimal solutions to the Standard Pooling Problem. To what extent the approximation capability is a general or an instance-specific property is a research question worthy of being investigated, both theoretically and experimentally.

An adequate experimental evaluation of the approach is left to be made. Numerical experiments reported so far are insufficient to conclude about strengths and weaknesses, and should not be considered as a complete assessment. In the full-length version of this paper, we will carry out a more thorough study of the theory and the solution methods for the pooling problem with single-flow constraints.

REFERENCES

[1] N. Adhya, M. Tawarmalani, and N. V. Sahinidis. 1999. A Lagrangian Approach to the Pooling Problem. *Industrial and Engineering Chemistry Research* 38, 5 (1999), 1956–1972.

[2] M. Alfaki and D. Haugland. 2013. Strong Formulations for the Pooling Problem. *Journal of Global Optimization* 56, 3 (2013), 897–916.

[3] M. Alfaki and D. Haugland. 2014. A Cost Minimization Heuristic for the Pooling Problem. *Annals of Operations Research* 222, 1 (2014), 73–87.

[4] H. Almutairi and S. Elhedhli. 2009. A New Lagrangian Approach to the Pooling Problem. *Journal of Global Optimization* 45, 2 (2009), 237–257.

[5] C. Audet, J. Brimberg, P. Hansen, S. Le Digabel, and N. Mladenović. 2004. Pooling Problem: Alternate Formulations and Solution Methods. *Management Science* 50, 6 (2004), 761–776.

[6] T. E. Baker and L. S. Lasdon. 1985. Successive Linear Programming at Exxon. *Management Science* 31, 3 (1985), 264–274.

[7] R. Baltean-Lugojan and R. Misener. 2018. Piecewise Parametric Structure in the Pooling Problem: From Sparse Strongly-Polynomial Solutions to NP-hardness. *Journal of Global Optimization* 71, 4 (2018), 655–690.

[8] A. Ben-Tal, G. Eiger, and V. Gershovitz. 1994. Global Minimization by Reducing the Duality Gap. *Mathematical programming* 63, 2 (1994), 193–212.

[9] N. Boland, T. Kalinowski, and F. Rigterink. 2017. A Polynomially Solvable Case of the Pooling Problem. *Journal of Global Optimization* 67, 3 (2017), 621–630.

[10] C. W. Dewitt, L. S. Lasdon D. A. Brenner, and S. A. Melhem. 1989. OMEGA: An Improved Gasoline Blending System for Texaco. *Interfaces* 19, 1 (1989), 85–101.

[11] S. Dey and A. Gupte. 2015. Analysis of MILP Techniques for the Pooling Problem. *Operations Research* 63, 2 (2015), 412–427.

[12] G. Erbeyoglu and U. Bilge. 2016. PSO-based and SA-Based Metaheuristics for Bilinear Programming Problems: an Application to the Pooling Problem. *Journal of Heuristics* 22, 2 (2016), 147–179.

[13] C. A. Floudas and V. Visweswaran. 1990. A Global Optimization Algorithm (GOP) for CERTAIN Classes of Nonconvex NLPs. 1. Theory. *Computers and Chemical Engineering* 14, 12 (1990), 1397–1417.

[14] L. R. Foulds, D. Haugland, and K. Jörnsten. 1992. A Bilinear Approach to the Pooling Problem. *Optimization* 24 (1992), 165–180.

[15] B. Galan and I. E. Grossmann. 1998. Optimal Design of Distributed Wastewater Treatment Networks. *Industrial and Engineering Chemistry Research* 37, 10 (1998), 4036–4048.

[16] A. Gupte, S. Ahmed, S. Dey, and M. S. Cheon. 2017. Relaxations and Discretizations for the Pooling Problem. *Journal of Global Optimization* 67, 3 (2017), 631–669.

[17] D. Haugland. 2016. The Computational Complexity of the Pooling Problem. *Journal of Global Optimization* 64, 2 (2016), 199–215.

[18] D. Haugland and E. M. T. Hendrix. 2016. Pooling Problems with Polynomial-Time Algorithms. *Journal of Optimization Theory and Applications* 170, 2 (2016), 591–615.

[19] C. A. Haverly. 1978. Studies of the Behaviour of Recursion for the Pooling Problem. *ACM SIGMAP Bulletin* 25 (1978), 19–28.

[20] J. Kallrath05. 2005. Solving Planning and Design Problems in the Process Industry Using Mixed Integer and Global Optimization. *Annals of Operations Research* 140, 1 (2005), 339–373.

[21] M. Kimizuka, S. Kim, and M. Yamashita. 2018. Solving Pooling Problems by LP and SOCP Relaxations and Rescheduling Methods. *AnrXiv:1804.02857 [math.OC]* (2018).

[22] A. Marandi, J. Dahl, and E. de Klerk. 2018. A Numerical Evaluation of the Bounded Degree Sum-of-Squares Hierarchy of Lasserre, Toh, and Yang on the Pooling Problem. *Annals of Operations Research* 265, 1 (2018), 67–92.

[23] R. Misener and C. A. Floudas. 2009. Advances for the Pooling Problem: Modeling, Global Optimization, and Computational Studies Survey. *Applied and Computational Mathematics* 8, 1 (2009), 3–22.

[24] R. Misener and C. A. Floudas. 2010. Global Optimization of Large-Scale Generalized Pooling Problems: Quadratically Constrained MINLP Models. *Industrial and Engineering Chemistry Research* 49, 11 (2010), 5424–5438.

[25] R. Misener, J. P. Thompson, and C. A. Floudas. 2011. APOGEE: Global Optimization of Standard, Generalized, and Extended Pooling Problems via Linear and Logarithmic Partitioning Schemes. *Computers and Chemical Engineering* 35, 5 (2011), 876–892.

[26] N. V. Sahinidis and M. Tawarmalani. 2005. Accelerating Branch–and–Bound through a Modeling Language Construct for Relaxation–Specific Constraints. *Journal of Global Optimization* 32, 2 (2005), 259–280.

[27] V. Visweswaran. 1996. Computational results for an efficient implementation of the GOP algorithm and its variants. *In I.E. Grossmann (Ed.): Global Optimization in Engineering Design. Kluwer Series in Nonconvex Optimization and Its Applications* 9 (1996), 111–153.

Challenges in System Reliability and its application in Network Optimization

Guillermo Rela
Facultad de Ingeniería. Universidad de la República
Montevideo, Uruguay
grela@fing.edu.uy

Franco Robledo
Facultad de Ingeniería. Universidad de la República
Montevideo, Uruguay
frobledo@fing.edu.uy

Pablo Romero
Facultad de Ingeniería. Universidad de la República
Montevideo, Uruguay
promero@fing.edu.uy

This paper has been retracted by the authors.

Pages 101-106 of the Proceedings volume are therefore left blank.

[note by the editor of OpenProceedings.org]

ISBN 978-3-89318-079-0 on OpenProceedings, ISSN 2510-7437

DOI: 10.5441/002/inoc.2019.19

This paper has been retracted by the authors.

Pages 101-106 of the Proceedings volume are therefore left blank.

[note by the editor of OpenProceedings.org]

This paper has been retracted by the authors.

Pages 101-106 of the Proceedings volume are therefore left blank.

[note by the editor of OpenProceedings.org]

This paper has been retracted by the authors.

Pages 101-106 of the Proceedings volume are therefore left blank.

[note by the editor of OpenProceedings.org]

This paper has been retracted by the authors.

Pages 101-106 of the Proceedings volume are therefore left blank.

[note by the editor of OpenProceedings.org]

This paper has been retracted by the authors.

Pages 101-106 of the Proceedings volume are therefore left blank.

[note by the editor of OpenProceedings.org]

A Nested Decomposition Model for Reliable NFV 5G Network Slicing

Huy Duong and Brigitte Jaumard
Computer Science and Software Engineering, Concordia University
Montreal, QC, Canada
bjaumard@cse.concordia.ca

ABSTRACT

With the 5th generation of mobile networking (5G) on our doorstep, optical network operators are reorganizing their network infrastructures so that they can deploy different topologies on the same physical infrastructure on demand. This new paradigm, called network slicing, together with network function virtualization (NFV), can be enabled by segmenting the physical resources based on the requirements of the application level.

In this paper, we investigate a nested decomposition scheme for the design of reliable 5G network slicing. It involves revisiting and improving the previously proposed column generation models, and adding in particular the computation of dual bounds with Lagrangian relaxation in order to assess the accuracy of the solutions.

Extensive computational results show that we can get ε-optimal reliable 5G slicing solutions with small ε (about 1% on average) in fairly reasonable computational times.

1 INTRODUCTION

The 5th generation of mobile networking (5G) is based on the key technologies of Software-Defined Networking (SDN) and Network Function Virtualization (NFV) in order to offer multiple services with various performance requirements, e.g., low latency, high throughput, high reliability, or high security. SDN allows network operators to remotely (re)configure the physical network in order to reserve on demand networking resources. Virtual compute nodes (i.e., node with computing resources such as servers or a data center) can enable Virtual Network Functions (VNFs) running on top of general-purpose hardware, such as a cloud infrastructure.

Within the context of 5G networks, network slicing is an end-to-end logical network provisioned with a set of isolated virtual resources on a shared physical infrastructure. Slices are provided as different customized services to fulfill dynamic demands, with flexible resource allocations. In other words, a network slice is a self-contained network with its own virtual resources, topology, traffic flow, and provisioning rules. Network slicing is therefore a key feature of 5G networks, which allows the efficient resource share of a common physical infrastructure and consequently, reduces operators' network construction costs.

An interesting feature of SDN is its ability to process traffic while forwarding it, using "network functions" or "network services". The latter ones can implement header processing and payload processing functions, such as network address translation (NAT), firewall, or domain name system (DNS). They are called Virtual Network Functions (VNFs) and can be implemented in software on conventional processing systems (e.g., servers or data centers) that are co-located with networking equipment. The sequence of functions that need to be set up for a specific flow is referred to as a "service chain."

In this paper, we propose a 5G network slicing design model and algorithm, based on nested column generation. It aims at maximizing the number of granted slices while addressing the reliability requirements of network slices. In order to avoid the costly exact solutions of the sub-problems, we discuss how to compute bounds using Lagrangian relaxation, so that we can assess the accuracy of the output solutions.

The paper is organized as follows. Section 2 contains the literature review. Section 3 provides the detailed problem statement of the design of reliable 5G network slicing. An original nested decomposition model is proposed in Section 4. Algorithmic aspects are covered in Section 5. Numerical results are described in Section 6 and conclusions are drawn in the last section.

2 LITERATURE REVIEW

2.1 5G Network Slicing

Several papers and surveys have already appeared on 5G network slicing and described their various challenges and opportunities [1, 14]. Similarly, many studies and several surveys have been devoted to Network Function Virtualization (NFV), e.g., [20].

Very few studies look at the combination of reliable 5G slicing and NFV. Tang *et al.* [18] propose an MILP for 5G network slicing that maximizes the number of granted slices while minimizing their failure rate, without providing protection mechanisms.

Some authors looked at network slicing and NFV, more often in the wireless networks than in the wired optical ones. Challenges are discussed in, e.g., [10, 14].

Lin *et al.* [11] propose an exact algorithm using column generation aiming to minimize the total embedding cost in terms of spectrum cost and computation cost for a single virtual network request. Moreover, validation of the exact algorithm is made on a six node network. Large data instances are solved using a heuristic. Destounis *et al.* [3] also propose an exact column generation algorithm for network slicing without the NSF features. They solved data instances up to 200 nodes. Carella *et al.* [2] exemplified Network slicing as an addition to the current Cloud architecture and evaluated on a testbed architecture based on the Fraunhofer FOKUS and TU Berlinopen source Open Baton toolkit.

2.2 Nested Column Generation Decomposition

The idea of nested column generation is not new: several authors have already investigated it for various problems, e.g., Song [17] in logistics, Dohn and Mason [4] for staff rostering, Karabuk [8] for scheduling paratransit vehicles and Vanderbeck [19] for two-dimension cutting-stock.

However, most studies did not worry about assessing accurately the quality of the output solutions, except, e.g., [6, 19].

ISBN 978-3-89318-079-0 on OpenProceedings, ISSN 2510-7437

DOI: 10.5441/002/inoc.2019.20

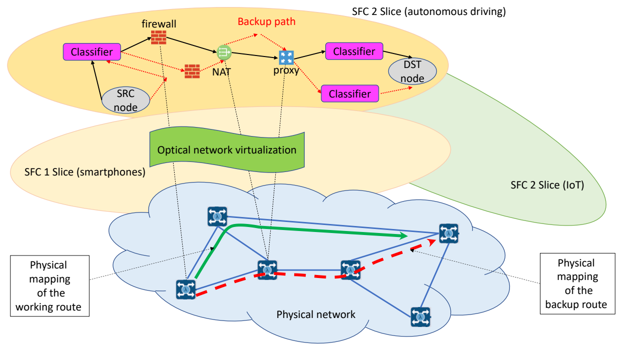

SFC 2 Slice (autonomous driving)
firewall
Backup path
Classifier
NAT
proxy
Classifier
DST node
SRC node
Classifier
Optical network virtualization
SFC 1 Slice (smartphones)
SFC 2 Slice (IoT)
Physical mapping of the working route
Physical network
Physical mapping of the backup route

Figure 1: 5G Reliable Slicing

3 PROBLEM STATEMENT AND NOTATIONS

3.1 Rel_5G_NFV Problem Statement

Consider a physical network G^{P} and a set K of connections, indexed by k. The Reliable 5G NFV Network Slicing (Rel_5G_NFV) problem consists of embedding/mapping the maximum number of slices onto the physical network while ensuring each slice is individually protected against any single link failure. We assume each slice is associated with a given application, that is characterized with the use of a single service function chain.

3.2 Notations

Physical Network. The physical network $G^{\text{P}} = (V^{\text{P}}, L^{\text{P}})$ is defined by its set of nodes V^{P}, indexed by v, set of links $\ell \in L^{\text{P}}$, with capacities $\text{CAP}_v \geq 0$ and $\text{CAP}_\ell \geq 0$ on both nodes and links, respectively.

5G Slicing. Each slice $S \in \mathcal{S}$ is associated with a virtual network $S = (V^S, L^S, \text{CAP}^S)$, which is defined by a set of virtual nodes V^S (indexed by v'), and virtual links L^S (indexed by ℓ'), with capacity requirements $\text{CAP}^S_{v'}$ and $\text{CAP}^S_{\ell'}$, respectively.

Virtual Networks. An embedding of S onto G^{P} consists of mapping:

- Each virtual node $v' \in V^S$ onto a physical node $v \in V^{\text{P}}$
- Each virtual link ℓ' onto a loop-free physical path, connecting two physical nodes u and v, to which the virtual nodes u' and v' have been mapped
- Each virtual "path" is protected by a virtual path, whose mapping is physical link-disjoint from the mapping of the first path,

in order to maximize the GoS.

A feasible embedding is an embedding in which all physical link and node capacity constraints are satisfied; that is, the sum of capacity demands of all virtual nodes embedded on a physical node is less than the capacity of this physical node, and the sum of the requests of all the virtual links going through a physical link does not exceed the capacity of this link.

In order to simplify the model and the algorithm, we work directly with the mapping of the virtual nodes/links, i.e., with physical nodes/links, without expressing explicitly the virtual links and nodes.

Service Function Chaining. Let F be the set of all services functions, indexed by f, and let C be the set of all service function chains, indexed by c. Any chain c is defined by an ordered sequence of n_c functions: $c = \{f_0, f_1, .., f_{n_c-1}\}$. The routing of any demand in a slice governed by SFC c must go through virtual compute nodes hosting the functions of c.

Application (Slice) Demand. Demands are provided for each slice S, with each slice being associated with one particular application, characterized by a given Service Function Chain (SFC) c_S. We denote by K^{sd,c_S} the demand for node pair $(v_s, v_d) \in \mathcal{SD}_{c_S}$, i.e., with traffic in slice S, subject to the requirement of SFC c_S, and by $\Delta^{sd,c}_{f_i}$ the required computational resource of function f_i for demand K^{sd,c_S}.

4 A NESTED DECOMPOSITION SCHEME

We now present a nested decomposition scheme, in which at the upper layer of the decomposition, we select the slice configurations for each slice demand. Each slice configuration is defined by a virtual network as defined in Section 3.2, which satisfies the demand K^{c_S} associated with its required application and corresponding SFC c_S.

Let Γ, indexed by γ, be set of all possible slice configurations. Each slice configuration γ is characterized by a slice S and its assigned resources. Each slice configuration γ is characterized by its slice index S, its node assigned resources R^γ_v, and its link assigned resources B^γ_ℓ. We have $\Gamma = \bigcup\limits_{c \in C} \Gamma_{c_S}$.

In order to simplify the notations, we will simply write c unless there is confusion.

4.1 Master Problem

Master problem maximizes the grade of service (GoS) subject to capacity constraints. It requires only one set of variables: $z_\gamma = 1$ if potential slice virtual network γ associated with c is selected, 0 otherwise, for $\gamma \in \Gamma_c$ and $c \in C$.

Objective:

$$\max \quad \sum_{c \in C} \sum_{\gamma \in \Gamma_c} \sum_{(s,d) \in \mathcal{SD}_c} K^{sd,c} \, z_\gamma \tag{1}$$

subject to:

$$\sum_{\gamma \in \Gamma_c} z_\gamma \leq 1 \qquad c \in C \tag{2}$$

$$\sum_{c \in C} \sum_{\gamma \in \Gamma_c} R^\gamma_v z_\gamma \leq \text{CAP}_v \qquad v \in V^{\text{P}} \tag{3}$$

$$\sum_{c \in C} \sum_{\gamma \in \Gamma_c} B^\gamma_\ell z_\gamma \leq \text{CAP}_\ell \qquad \ell \in L^{\text{P}} \tag{4}$$

$$z_\gamma \in \{0, 1\} \qquad \gamma \in \Gamma \tag{5}$$

Constraints (2) impose to select at most one virtual network (slice) for demand associated with $c \in C$. Constraints (3) enforce the compute node capabilities, while constraints (4) enforce the link transport capacities.

4.2 Slicing Pricing Problem (PP$_{\text{SLICE}}$)

In order to be able to compute the required node and link resource for a given slice, the pricing problem, or equivalently, the slice configuration generator, needs to provision the demand K^c. We define the following parameters.

Parameters:

- $\pi \in \Pi$: a logical path that defines a service path with chain c from s to d. Note that a logical path may go through a given physical link several times due to the sequence of functions in c.
- $\Pi^c_{sd} \subseteq \Pi$: set of all potential paths for service chain c from s to d.

- $a_v^{i,\pi} = 1$ if, on path π, function f_i is hosted on physical node v, 0 otherwise.
- δ_ℓ^π = number of times path π goes through link ℓ
- $x_\ell^\pi = 1$ if logical path π goes through physical link ℓ at least once, 0 otherwise.

Variables:

- $y_{\pi,p}^{sd,c} = 1$ if path π is the primary path to provision traffic from s to d, 0 otherwise.
- $y_{\pi,b}^{sd,c} = 1$ if path π is the backup path to provision traffic from s to d, 0 otherwise.

Objective:

$$\max \quad \text{RC}_{\text{PP}_{\textsc{slice}}} = \sum_{(s,d)\in\mathcal{SD}} K^{sd,c} - u_c^{(2)}$$
$$- \sum_{v\in V^{\text{P}}} u_v^{(3)} \sum_{(s,d)\in\mathcal{SD}} \sum_{i=0}^{n_c-1} \sum_{\pi\in\Pi_{sd}^c} \Delta_{f_i}^{sd} a_v^{i,\pi} (y_{\pi,p}^{sd,c} + y_{\pi,b}^{sd,c})$$
$$- \sum_{\ell\in L^{\text{P}}} u_\ell^{(4)} \sum_{(s,d)\in\mathcal{SD}} \sum_{\pi\in\Pi_{sd}^c} K^{sd,c} \delta_\ell^\pi (y_{\pi,p}^{sd,c} + y_{\pi,b}^{sd,c}) \quad (6)$$

Constraints:

One primary path per demand:

$$\sum_{\pi\in\Pi_{sd}^c} y_{\pi,p}^{sd,c} = 1 \qquad (v_s, v_d) \in \mathcal{SD} \qquad (7)$$

One backup path per demand:

$$\sum_{\pi\in\Pi_{sd}^c} y_{\pi,b}^{sd,c} = 1 \qquad (v_s, v_d) \in \mathcal{SD}. \qquad (8)$$

Link disjoint primary and backup paths:

$$\sum_{\pi\in\Pi_{sd}^c} x_\ell^\pi (y_{\pi,p}^{sd,c} + y_{\pi,b}^{sd,c}) \le 1 \qquad (v_s, v_d) \in \mathcal{SD}, \ell \in L^{\text{P}}. \quad (9)$$

Link and node capacities:

$$(R_v =) \sum_{(v_s,v_d)\in\mathcal{SD}} \sum_{i=0}^{n_c-1} \sum_{\pi\in\Pi_{sd}^c} \Delta_{f_i}^{sd} a_v^{i,\pi} (y_{\pi,p}^{sd,c} + y_{\pi,b}^{sd,c})$$
$$\le \text{CAP}_v \qquad v \in V^{\text{P}} \quad (10)$$
$$(B_\ell =) \sum_{(v_s,v_d)\in\mathcal{SD}} \sum_{\pi\in\Pi_{sd}^c} K^{sd,c} \delta_\ell^\pi \, (y_{\pi,p}^{sd,c} + y_{\pi,b}^{sd,c})$$
$$\le \text{CAP}_\ell \qquad \ell \in L^{\text{P}}. \quad (11)$$

4.3 Path Pricing Problem (PP$_{sd}$): Service path for Demand from v_s to v_d

For a given $(v_s, v_d) \in \mathcal{SD}$, we look for the generation of a path π from v_s to v_d, which can improve the linear programming relaxation of PP$_{\textsc{slice}}$.

Variables:

- $x_\ell^\pi = 1$ if path π uses ℓ, 0 otherwise.
- δ_ℓ^π = number of times path π goes through ℓ.
- $\varphi_\ell^{sd,c,i} = 1$ if, for service chain c, the path from v_s to v_d uses link ℓ to go from the location of function f_{i-1} to the location of function f_i, 0 otherwise. Note that, when $i = 0$, $\varphi_\ell^{sd,c,i}$ represents the path from the source to the first function, when $i = n_c$, it is the path from the last function to the destination.
- $a_v^i = 1$ if the ith function (f_i) of chain c is installed on node v, 0 otherwise.

Objective:

$$\max \quad \Big(- \sum_{v\in V^{\text{P}}} u_v^{(3)} \sum_{i=0}^{n_c-1} \Delta_{f_i}^{sd} a_v^{i,\pi} - \sum_{\ell\in L^{\text{P}}} u_\ell^{(4)} K^{sd,c} \delta_\ell^\pi) \Big)$$
$$- u_{sd,p}^{(7)} - \sum_{\ell\in L^{\text{P}}} x_l^\pi u_{sd}^{(9)}$$
$$- \sum_{v\in V} \sum_{i=0}^{n_c-1} \Delta_{f_i}^{sd} a_v^i u_v^{(10)} - \sum_{\ell\in L} u_\ell^{(11)} K^{sd,c} \delta_\ell^\pi \quad (12)$$

Constraints:

Aggregation of link usage:

$$\delta_\ell^\pi = \sum_{i=0}^{n_c} \varphi_\ell^{sd,c,i} \qquad \ell \in L^{\text{P}}. \qquad (13)$$

Multiple usage of a link:

$$\varphi_\ell^i \le x_\ell^\pi \qquad \ell \in L^{\text{P}}, i \in 0, .., n_c - 1. \qquad (14)$$

This set of constraints ensures that x_ℓ keeps track of physical link ℓ if it is used by any logical link. Indeed, a link can be used multiple times by a given path, this set of constraints result x_ℓ as used links, no matter how many times they are used. These variables play the role in the upper pricing where backup path and primary path must be disjoint.

Flow Conservation constraints

$$\sum_{\ell\in\omega^+(v)} \varphi_\ell^{sd,c,0} - \sum_{\ell\in\omega^- v} \varphi_\ell^{sd,c,0} + a_v^{sd,c,0}$$
$$= \begin{cases} 1 \text{ if } v = v_s \\ 0 \text{ else} \end{cases} \qquad v \in V^{\text{P}} \qquad (15)$$
$$\sum_{\ell\in\omega^+(v)} \varphi_\ell^{sd,c,n_c} - \sum_{l\in w^- v} \varphi_\ell^{sd,c,n_c} - a_v^{sd,c,n_c-1}$$
$$= \begin{cases} -1 & \text{if } v = v_d \\ 0 & \text{else} \end{cases} \qquad v \in V^{\text{P}} \qquad (16)$$
$$\sum_{\ell\in\omega^+(v)} \varphi_\ell^{sd,c,i} - \sum_{\ell\in\omega^-(v)} \varphi_\ell^{sd,c,i} + a_v^{sd,c,i} - a_v^{sd,c,i-1} = 0$$
$$v \in V^{\text{P}}, 0 < i < n_c. \qquad (17)$$

Constraints (15) ensure that demand starts at the source node, then is transferred through a path to the location of first function (unless first function is located at the source node). Similarly, constraints (16) make sure that the demand is delivered to the destination after it is processed by the last function (unless the last function is installed at the destination node). From the location of function $i - 1$ to the location of function i, constraints (17) define a path to connect them.

We next use constraints to eliminate the ineffective solutions and, as a consequence, those constraints help to improve the quality of the columns, i.e., slice configurations.

A unique node location for each function occurrence in the service chain:

$$\sum_{v\in V^{\text{P}}} a_v^i = 1 \qquad i = 0, 1, \ldots, n_c. \qquad (18)$$

If a link is not used, its corresponding x_ℓ can be set to zero:

$$x_\ell \le \delta_\ell^\pi \qquad \ell \in L^{\text{P}} \qquad (19)$$

Domain constraints:

$$x_\ell^\pi, \varphi_\ell^\pi, a_v^i \in \{0, 1\}; \qquad \delta_\ell^\pi \in \mathbb{Z}^+ \qquad (20)$$

Node capacity constraints:

$$\sum_{i=0}^{n_c-1} \Delta_{f_i} a_v^i \leq \textsc{cap}_v \qquad v \in V^{\text{P}} \tag{21}$$

Link capacity constraints:

$$\delta_\ell^\pi K^{sd,c} \leq \textsc{cap}_\ell \qquad \ell \in L^{\text{P}}. \tag{22}$$

We will discuss in the next section how to solve efficiently the path pricing problems, without requiring the solution of ILP programs at each iteration of the column generation algorithm.

5 NESTED COLUMN GENERATION ALGORITHM

Column generation [9] is based on the fact that, in the simplex method, the solver does not need to simultaneously access all variables of the problem. In fact, a solver can start working only with the basis (a particular subset of the constrained variables), then use a reduced cost to choose the other variables to access, as needed. It is today a very well known and powerful technique [5, 12], while column generation modeling remains an art when the decomposition is not deduced from the application of the Dantzig-Wolfe decomposition.

We next provide the details of our nested column generation algorithm and how we estimated the accuracies of the resulting solutions.

5.1 Nested CG and ILP Solution

The conceptual column generation scheme alternates between solving a restriction of the original problem, usually called restricted master problem, and a column generation phase which is used to augment the set of variables/columns of the restricted master problem using a so-called pricing problem. Here, the pricing problem can be decomposed into $|\mathcal{S}|$ slice pricing subproblems.

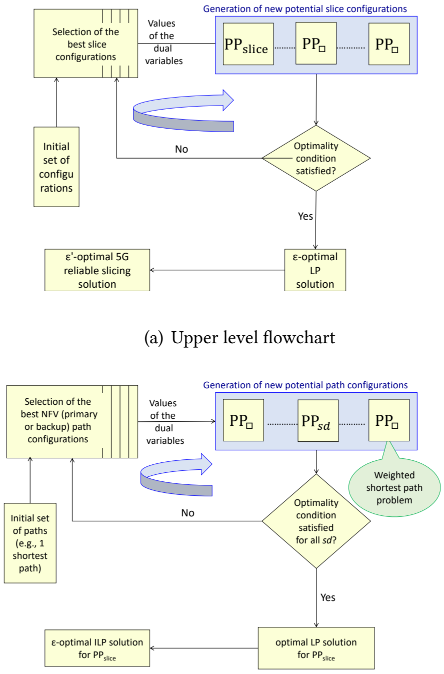
In order to guarantee reaching an optimal LP solution, it is required to solve at least once the pricing problem. In this study, we propose to solve the slice pricing problem, indeed, the slicing pricing subproblems using again a column generation algorithm. As these last subproblems are Integer Linear Programs (ILPs), and as we did not develop any branch-and-price algorithms to solve them, they are never solved optimally, and therefore we need to derive a linear relaxation bound in order to get upper bounds, see next section for the details.

In any case, at both decomposition levels, we use the column generation algorithm as long as we can derive new improving columns. For integer solutions, when we cannot improve anymore the LP solution, we use an ILP solver on the current constraint matrix, i.e., the constraint matrix made of all the columns generated so far, and deduce an ILP solution.

Flowcharts in Figure 2 summarize the algorithm. Accuracy of the output solutions is assessed with ε, which is defined as follows:

$$\varepsilon = \frac{\overline{z}_{\text{LP}} - \tilde{z}_{\text{ILP}}}{\tilde{z}_{\text{LP}}},$$

where $\overline{z}_{\text{LP}}$ is an upper bound the LP solution of problem (1)-(5), whose calculation is developed in Section 5.2. $\tilde{z}_{\text{ILP}}$ is the best found ILP solution (hence a lower bound on the ILP solution), as derived by the solution of the ILP solver on the constraint matrix of (1)-(5) when no more improved column can be generated by the solution of the slice pricing problem (6)-(11).


Generation of new potential slice configurations
Selection of the best slice configurations
Values of the dual variables
PP_{slice}
$\text{PP}_{\square}$
$\text{PP}_{\square}$
Initial set of configu rations
No
Optimality condition satisfied?
Yes
ε'-optimal 5G reliable slicing solution
ε-optimal LP solution

(a) Upper level flowchart

Generation of new potential path configurations
Selection of the best NFV (primary or backup) path configurations
Values of the dual variables
$\text{PP}_{\square}$
PP_{sd}
$\text{PP}_{\square}$
Weighted shortest path problem
Initial set of paths (e.g., 1 shortest path)
No
Optimality condition satisfied for all *sd*?
Yes
ε-optimal ILP solution for PP_{slice}
optimal LP solution for PP_{slice}

(b) Lower level flowchart

Figure 2: Flowcharts

In order to speed-up the solution of the path pricing subproblems, we first use a shortest path algorithm after noting that all the link costs are positive, taking into account the values of the dual variables. It is worth noting that the usage of a shortest path algorithm does not necessarily guarantee the generation of feasible lightpaths with respect to link and node capacities. However, those capacities are enforced in the slice pricing subproblems, and therefore taken care. When the path pricing subproblems are not able to generate improving paths (i.e., with a negative reduced cost), then we use an ILP solver to solve them, with the guarantee to satisfy all node and link capacities.

5.2 Solution Accuracy

The nested column generation framework allows the efficient exploitation of the substructures of a problem at the expense of a more difficult exact solution of the linear programming relaxation as it a priori requires the exact solution of the upper level pricing problem (here the slice $\text{PP}_{\textsc{slice}}$ pricing problem), i.e., a branch-and-price algorithm. In order to overcome that difficulty, we propose to compute an upper bound on the objective (i.e., reduced cost) of the $\text{PP}_{\textsc{slice}}$ problem, and then deduce an upper bound on the optimal LP solution of the Rel_5G_NFV master problem (1)-(5). It then allows the evaluation of the accuracy (gap) of output ILP solutions using the algorithm described in the previous section.

Consider the compact formulation associated with (1)-(5), i.e., the COMPACT model such that when applying a Dantzig-Wolfe decomposition to it, we derive model (1)-(5). Let

$$[\textsc{compact}] \qquad \max\{cx : Ax \leq b, x \in X\}.$$

Using the Dantzig-Wolfe decomposition of Model COMPACT, the slicing pricing problem, $\mathrm{PP}_{\textsc{slice}}$, can be written as follows:

$$\mathrm{RC}^{\star}_{\mathrm{PP}_{\textsc{slice}}} = \max\left\{\bar{c}\,x : x \in X^{\textsc{pricing}}\right\}. \tag{23}$$

We simply write RC to shorten $\mathrm{RC}_{\mathrm{PP}_{\textsc{slice}}}$ when there is no ambiguity so that $\mathrm{RC}^{\star}_{\mathrm{PP}_{\textsc{slice}}} = \mathrm{RC}^{\star}$.

In Figure 3, we rank the relative positions of the various values that we discuss below. Question marks indicate values that are not computed accurately, and that are upper/lower bounded.

The Lagrangian relaxation of the COMPACT Model can be written:

$$\mathrm{LR}(u) = \max_{x \in X} \left\{ L(u,x) = ub + \underbrace{(c - uA)x}_{\mathrm{RC}(u,x)} \right\}. \tag{24}$$

Following Vanderbeck [19] and Pessoa *et al.* [15], a valid upper bound for the COMPACT problem can be computed using Lagrangian Relaxation (LR). At any iteration τ of the column generation algorithm, i.e., when we re-optimize the linear relaxation of the master problem (1)-(5), the optimal $x_{\mathrm{RC}^{\star}}$ that maximizes $L(u_\tau, x_{\mathrm{RC}^{\star}})$ can be written:

$$\begin{aligned} x_{\mathrm{RC}^{\star}} &= \arg\max_{x \in X} L(u_\tau, x) \quad = \arg\max_{x \in X} \mathrm{RC}(u_\tau, x) \\ &= \arg\max_{i \in I} \mathrm{RC}(u_\tau, x^i) = \arg \max_{x \in X^{\textsc{pricing}}} \mathrm{RC}(u_\tau, x), \end{aligned}$$

where $x^i, i \in I$ denote the extreme points of X, see [13], Section II.3.6.

As $x_{\mathrm{RC}^{\star}}$ is known only if we solve $\mathrm{PP}_{\textsc{slice}}$ exactly, we can bound it in order to get an upper bound, $\bar{z}_{\mathrm{LP}}$, on the optimal value of the linear programming relaxation. Indeed, $\mathrm{RC}^{\star,\tau}_{\mathrm{ILP}} \leq \mathrm{RC}^{\star,\tau}_{\mathrm{LP}}$, where $\mathrm{RC}^{\star,\tau}_{\mathrm{LP}}$ is the optimal value of the LP relaxation of $\mathrm{PP}_{\textsc{slice}}$ at iteration τ of the column generation algorithm.

Consequently, $L(u_\tau, x_{\mathrm{RC}^{\star}}) = u_\tau b + \mathrm{RC}^{\star,\tau}_{\mathrm{ILP}} \leq u_\tau b + \mathrm{RC}^{\tau}_{\mathrm{LP}} = \bar{z}^{\tau}_{\mathrm{LP}}$.

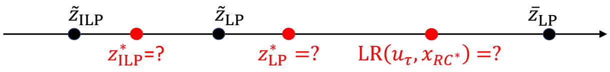

$\tilde{z}_{\mathrm{ILP}}$
$\tilde{z}_{\mathrm{LP}}$
$\bar{z}_{\mathrm{LP}}$
z^{*}_{ILP}=?
z^{*}_{LP} =?
$\mathrm{LR}(u_\tau, x_{RC^*})$ =?

Figure 3: Ranking of the various LP, LR and ILP values.

At each iteration τ of the column generation algorithm, each pricing problem is decomposed into $|\mathcal{S}|$ elementary slice pricing problems of the type $\mathrm{PP}_{\textsc{slice}}$. It implies:

$$\bar{z}^{\tau}_{\mathrm{LP}} = u_\tau b + \sum_{S \in \mathcal{S}} \mathrm{RC}^{\star}_{\mathrm{LP}} \left(\mathrm{PP}_{\textsc{slice}}(S)\right).$$

Note that the Lagrangian relaxation upper bound does not improve monotonically [15], thus, in order to derive the best possible upper bound, the algorithm must compute

$$\bar{z}_{\mathrm{LP}} = \min_{\tau} \bar{z}^{\tau}_{\mathrm{LP}} = \min_{\tau} \max_{S \in \mathcal{S}} \mathrm{RC}^{\star,\tau}_{\mathrm{LP}} \left(\mathrm{PP}_{\textsc{slice}}(S)\right).$$

It remains possible to add several columns (i.e., slices) at a time (whose $\widetilde{\mathrm{RC}}^{\tau}_{\mathrm{ILP}} \left(\mathrm{PP}_{\textsc{slice}}(S)\right) > 0$) to the master problem (1)-(5) in one iteration, as long as they are generated with the same set of dual values. Note that output ILP solutions of $\mathrm{PP}_{\textsc{slice}}(S)$ are not guaranteed to be optimal, hence the notation $\widetilde{\mathrm{RC}}$ to denote a heuristic solution of the slice pricing problem. Indeed, the algorithm has to go through all slice subproblems in each iteration to ensure the correctness of the Lagrangian bound.

6 NUMERICAL RESULTS

We implemented the model and algorithm described in the previous sections with a C++ program on a Linux computer with 773727 MB RAM and Intel Xeon E5-2687W v3 @ 3.10 GHz 2 processors, 20 cores. We first describe the data sets, and then we report on the performance of the algorithm.

6.1 Data Sets

We considered two topologies from SNDLib [16] and their characteristics are described in Table 1. We re-use the traffic matrix of [7] with four SFCs. In order to derive slice demand, for each original SFC in [7], we divided the overall traffic in 4 subsets, resulting into traffic demands for 16 slices. Transport capacities were set with the optimal solution when allowing only one NFV node.

Table 1: Data sets

Topologies	# nodes	# links	# connections per slice	# slices	Offered load
INTERNET2	10	34	90	16	1Tb
ATLANTA	15	44	210	16	1Tb

6.2 Model and Algorithm Efficiency and Accuracy

We conducted experiments with the same link transport capacities, and increased node capacities as we increase the number of NFV (compute) nodes. Corresponding accuracies and computational times (seconds) are reported in Table 2. We observe that resulting accuracies are less than 3% except for 4 cases where the gap can reach up to 5.6%. Data Instances are easier to solve as the number of NFVs is increasing, and computational times are fairly reasonable taking inot account the accuracies and the complexity of the design problem of reliable 5G network slicing.

Table 2: Nested CG performance

# NFV	INTERNET2		ATLANTA	
nodes	gap (%)	CPU	gap (%)	CPU
1	3.8	454.9	5.6	991.8
2	3.4	574.2	4.3	4,215.9
3	0.4	89.4	2.9	1,040.2
4	0.4	65.7	2.9	1,010.1
5	0.4	89.9	2.9	578.3
6	0.4	36.3	2.9	582.9
7	0.4	36.6	0.0	651.5
8	0.1	158.8	2.9	758.5
9	0.1	34.8	0.1	601.3
10	0.1	35.1	0.0	561.5
11	-	-	0.0	442.5
12	-	-	0.0	572.6
13	-	-	0.0	524.8
14	-	-	0.1	554.3
15	-	-	0.0	557.8

6.3 Network Spectrum Usage

We investigated how the network spectrum is used when the number of nodes with compute capacities is increasing, i.e., when there are more network functions distributed all over the network. We provide the results for the ATLANTA topology in Figure 4.

Plots of Figure 4 show that it is more or less the same subset of links which are the most loaded, but their load vary with the number and location of the NFVs, and the increase of the overall network load when the number of NFVs is increasing. Sometimes we see a drop in the load of a link, which is explained by the increase and position of more NFVs. In conclusion, dimensioning of the link is very dependent on the number and location of the NFVs.

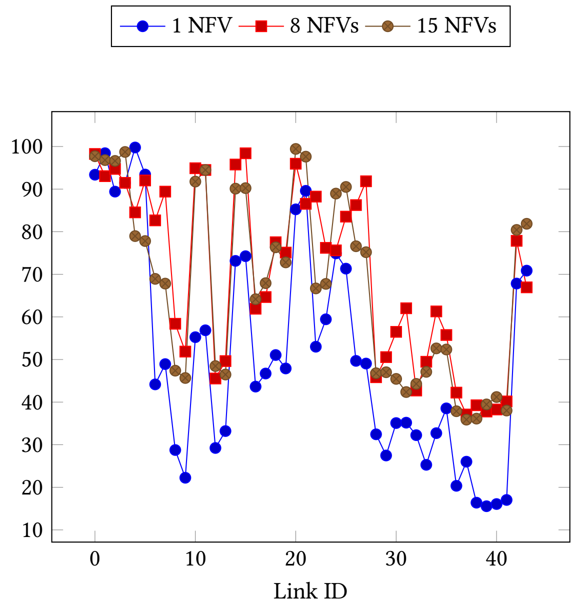


Figure 4: Physical Link Load - ATLANTA Topology

We also investigated the throughput evolution when the number of NFV nodes increases and results are depicted in Figure 5 for the ATLANTA network. We observe that as soon as we reach four or five NFV nodes, then the throughput does not increase significantly anymore.

7 CONCLUSIONS

We designed a first efficient nested decomposition scheme for reliable 5G slicing. Future work will include several algorithmic enhancements such as parallel solutions of pricing problems and greedy heuristics to generate an initial solution (i.e., initial columns at both decomposition levels).

ACKNOWLEDGMENTS

B. Jaumard has been supported by a Concordia University Research Chair (Tier I) on the Optimization of Communication Networks and by an NSERC (Natural Sciences and Engineering Research Council of Canada) grant.

REFERENCES

[1] M.S. Bonfim, K.L. Dias, and S.F.L. Fernandes. 2019. Integrated NFV/SDN Architectures: A Systematic Literature Review. *Journal ACM Computing Surveys (CSUR)* 51, 6 (2019), xxx – xxx.

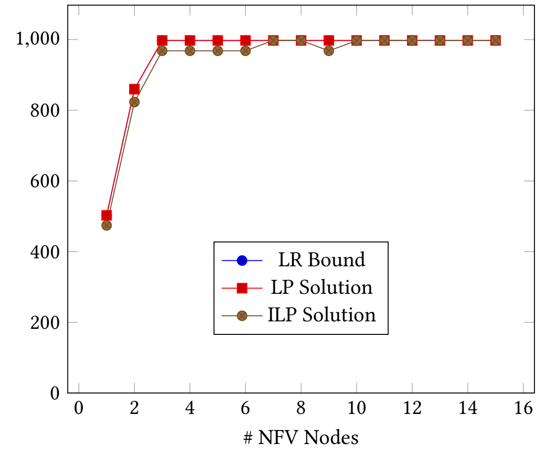


Figure 5: Throughput evolution with an increasing number of NFV nodes

[2] G. Carella, M. Pauls, A. Medhat, L. Grebe, and T. Magedanz. 2017. A Network Function Virtualization framework for Network Slicing of 5G Networks. In *Mobilkommunikation–Technologien und Anwendungen*. ITG-Fachtagung, Osnabrück, Deutschland, 1–7.

[3] A. Destounis, G. Paschos, S. Paris, J. Leguay, L. Gkatzikis, S. Vassilaras, M. Leconte, and P. Medagliani. 2018. Slice-based column generation for network slicing. In *Annual Joint Conference of the IEEE Computer and Communications Societies - INFOCOM*. IEEE, Honolulu, HI, USA, 1–2.

[4] A. Dohn and A. Mason. 2013. Branch-and-price for staff rostering: An efficient implementation using generic programming and nested column generation. *European Journal of Operational Research* 230 (2013), 157–169.

[5] J.B. Gauthier, J. Desrosiers, and M.E. Lübbecke. 2018. Vector Space Decomposition for Solving Large-Scale Linear Programs. *Operations Research* 66, 5 (2018), 1376–1389.

[6] F. Hennig, B. Nygreen, and M.E. Lübbecke. 2012. Nested Column Generation Applied to the Crude Oil Tanker Routing and Scheduling Problem with Split Pickup and Split Delivery. *Naval Research Logistics* 59 (April âĂŘ June 2012), 298–310. Issue 3âĂŘ4.

[7] N. Huin, B. Jaumard, and F. Giroire. 2018. Optimal Network Service Chain Provisioning. *IEEE/ACM Transactions on Networking* 26, 3 (June 2018), 1320–1333.

[8] S. Karabuk. 2009. A nested decomposition approach for solving the paratransit vehicle scheduling problem. *Transportation Research Part B* 43 (2009), 448–465.

[9] L.S. Lasdon. 1970. *Optimization Theory for Large Systems*. MacMillan, New York.

[10] X. Li, M. Samaka, H.A. Chan, D. Bhamare, L. Gupta, C. Guo, and R. Jain. 2017. Network Slicing for 5G: Challenges and Opportunities. *IEEE Internet Computing* 21, 5 (2017), 20–27.

[11] R. Lin, S. Luo, J. Zhou, S. Wang, B. Chen, X. Zhang, A. Cai, W.-D. Zhong, and M. Zukerman. 2018. Column generation algorithms for virtual network embedding in flexi-grid optical networks. *Optics Express* 26, 8 (Apr 2018), 10898–10913.

[12] M.E. Lübbecke and J. Desrosiers. 2005. Selected Topics in Column Generation. *Operations Research* 53 (2005), 1007–1023. Issue 6.

[13] George L. Nemhauser and Laurence A. Wolsey. 1988. *Integer and Combinatorial Optimization*. Wiley, New York.

[14] J. Ordonez-Lucena, P. Ameigeiras, D. Lopez, J.J. Ramos-Muñoz, J. Lorca, and J. Folgueira. 2017. Network Slicing for 5G with SDN/NFV: Concepts, Architectures and Challenges. *IEEE Communications Magazine* 55 (2017), 80–87. Issue 5.

[15] A. Pessoa, R. Sadykov, E. Uchoa, and F. Vanderbeck. 2018. Automation and Combination of Linear-Programming Based Stabilization Techniques in Column Generation. *INFORMS Journal on Computing* 30, 2 (2018), 339–360.

[16] SNDlib. 2005. Germany50 Problem. http://sndlib.zib.de/home.action/. (October 2005).

[17] S.H. Song. 2009. A nested column generation algorithm to the meta slab allocation problem in the steel making industry. *Journal International Journal of Production Research* 47, 13 (2009), 3625–3638.

[18] L. Tang, G. Zhao, C. Wang, P. Zhao, and Q. Chen. 2018. Queue-aware reliable embedding algorithm for 5G network slicing. *Computer Networks* 146 (December 2018), 138 – 150.

[19] F. Vanderbeck. 2001. A Nested Decomposition Approach to a Three-Stage, Two-Dimensional Cutting-Stock Problem. *Management Science* 47, 6 (2001), 864–879.

[20] B. Yi, X. Wang, K. Li, S.K. Das, and M. Huan. 2018. A comprehensive survey of Network Function Virtualization. *Computer Networks* 133 (2018), 212 – 262.

A heuristic algorithm for a vehicle routing problem with pickup & delivery and synchronization constraints

Seddik Hadjadj
Laboratoire d'informatique en image et systèmes d'information
Villeurbanne, France
mohamed-seddik.hadjadj@liris.cnrs.fr

Hamamache Kheddouci
Laboratoire d'informatique en image et systèmes d'information
Villeurbanne, France
hamamache.kheddouci@univ-lyon1.fr

ABSTRACT

In this paper, we consider a vehicle routing problem with pickup & delivery and synchronization constraint. One vehicle with a known and finite capacity has to visit *n* customers to pickup or deliver empty containers. At the same time, another vehicle has to deliver ready-mixed concrete by pouring it into the previously delivered containers. This implies dealing with capacity and temporal precedence constraints.

We propose a heuristic to tackle this problem. A two-step approach including a local search and a constructive algorithm. We provide some experiments that show positive results.

1 INTRODUCTION

This work is carried out in collaboration with a company which specializes in the sale of ready-mixed concrete.

Today, each ready-mixed concrete order requires the mobilization of one or more *mixer trucks*, even for very small quantities of concrete. However, such trucks are cumbersome, expensive, and disproportionate in some cases, especially when delivering small quantities of concrete.

Therefore, the company proposes a new delivery method to deal more effectively with such orders. The idea is to share a single mixer truck by several customers with small quantities (≤ 500 liters), which implies organizing optimized mixer truck tours.

On the other hand, to ensure the profitability of such truck tours, waiting times at each customer's location have to be reduced. Today, a mixer truck has to wait until the concrete is poured on site to leave a customer's location, and this causes a huge waste of time. To tackle this problem, the company proposes to pour the concrete from the truck into special containers instead of pouring it directly on site which is more difficult and takes more time. The truck can then leave faster, and the customer can use the concrete in the containers all day long. Waiting times are then drastically reduced.

However, since the containers which are supposed to contain the concrete are special, they also must be delivered to the customer. This implies organizing another tour to deliver the containers and pick them up after they have been used.

In brief, this new method is a three-step process :

1. A vehicle delivers a number of empty containers to the customer ;
2. Thereafter, a mixer truck delivers a certain quantity of ready-mixed concrete by pouring it into the containers delivered ;
3. The next day, the vehicle returns to the customer to pick up the containers after they have been used.

To ensure the profitability of this method, the company needs a decision support system that can generate two efficient and synchronized vehicle tours : a pickup & delivery tour for the containers, and a mixer truck tour to deliver the concrete, knowing that each customer has to receive the containers before the concrete (temporal precedence constraint), and that the vehicle carrying the containers has a maximal capacity (capacity constraint).

This paper aims to provide an efficient heuristic to build such synchronized vehicle tours minimizing the total travel times.

The paper is structured as follows. Section 2 provides a literature review of vehicle routing problems with pickup & delivery. Section 3 gives a formulation for the problem tackled in this work. In section 4, we present our heuristic. Section 5 is devoted to some experimental results, and section 6 concludes the paper.

2 LITERATURE REVIEW

We present a brief review of pickup & delivery problems.

2.1 Pickup & Delivery Problems

There are three main classes of pickup & delivery problem in the literature :

2.1.1 One-to-One Problems. One or more vehicles have to carry *n* commodities, where each commodity has an origin and a destination. One of the best known examples of this class is the *Dial-a-ride problem* which consists of transporting people from an origin to a destination. The problem has been studied for both single [13] and multiple [4] vehicle cases, with various types of constraints related to ride times, time windows [5, 14]...

2.1.2 One-to-Many-to-One Problems. Commodities are divided into "delivery commodities" and "pickup commodities". One or more vehicles have to carry the delivery commodities from the depot to the customers and the pickup commodities from the customers to the depot. Assuming that n_p is a set of pickup customers, and n_d a set of delivery customers, two cases have been distinguished for these problems : single demands, where $n_p \cap n_d = \emptyset$, and combined demands, where $n_p \cap n_d \neq \emptyset$. For the latter case, [7] consider various possible path types such as the *Hamiltonian* path, where each customer is visited once such that pickup and delivery are performed simultaneously, as well as the *double path*, where each customer that has a combined demand (pickup and delivery) is visited twice, the first time for pickup, the second for delivery. Several heuristics have been proposed for both path types for the single and the multi-vehicle cases [3, 12]...

2.1.3 Many-to-Many Problems. One or more vehicles have to transport goods between customers knowing that each customer can be a source or a destination of any type of good. Among the problems of this class, the *One-Commodity pickup and delivery*

ISBN 978-3-89318-079-0 on OpenProceedings, ISSN 2510-7437

DOI: 10.5441/002/inoc.2019.21

travelling salesman problem is the variant that we consider in this work. It was introduced in [10]. A single vehicle with a known and finite capacity has to carry a single commodity between pickup customers and delivery customers. Picked up commodities can be supplied to delivery customers. This problem is known to be *NP-Hard.* Moreover, checking the existence of a feasible solution is an *NP-Complete* problem [8]. Studies on such problems are relatively scarce. A branch and cut algorithm has been proposed in [10] for small instances, and two heuristics have been developed in [11] to tackle larger instances, in particular by defining "the infeasibility of a path", and adapting the nearest neighbourhood heuristic to increase the chance of obtaining a feasible solution. Furthermore, [9] have proposed a hybrid method combining GRASP (greedy randomized adaptive search procedure) and VND (variable neighbourhood descent) metaheuristics. This method gave better results than the previously proposed ones.
For a detailed survey on pickup and delivery problems, we refer the reader to [1].

3 PROBLEM FORMULATION

Given two vehicles V_1 and V_2 such that :

- V_1 is in charge of delivering (or picking up) empty containers, and has a known and finite maximum capacity Q (Q is the maximum number of containers that can be carried by the vehicle) ;
- V_2 is a mixer truck carrying a sufficient quantity of concrete.

And considering :

- D_1 the depot of V_1 ;
- D_2 the depot of V_2 ;
- $N = \{1, ..., n\}$ a set of n customers who require a visit of V_1 and/or V_2 ;
- $N = N_p \cup N_d$, where :
  - N_d is the set of customers who require delivering containers + concrete (who require a visit of both V_1 and V_2) ;
  - N_p is the set of customers who require picking up containers (who require a visit of V_1 only) ;
  - $N_p \cap N_d = \emptyset$.

The problem can be defined on a complete graph $G = (V, E)$ as follows (see Fig 1) :

- $V = \{D_1\} \cup \{D_2\} \cup N$ is a set of $n + 2$ nodes ;
- $E = \{(i, j), i, j \in V, i \neq j\}$ is a set of edges representing connections between nodes ;
- $C = \{c_{i,j}, (i, j) \in E\}$ represents the travel time between i and j ($c_{i,j} = c_{j,i}, \forall (i, j) \in E$) ;
- $D = \{d_i, i \in N\}$ is a set of customers' demands ($|d_i|$ is the number of containers to deliver to / pick up from customer i, $d_i < 0 \quad \forall i \in N_d$ and $d_i > 0 \quad \forall i \in N_p$) ;

Assuming that :

- $x_{i,j}$ is a boolean variable such that :
  - $x_{i,j} = 1$ if j is visited immediately after i by V_1,
  - 0 otherwise.
- $y_{i,j}$ is a boolean variable such that:
  - $y_{i,j} = 1$ if j is visited immediately after i by V_2,
  - $y_{i,j} = 0$ otherwise.

  (Note that $y_{i,j} = 0 \quad \forall i \in N_p, \forall j \in N_p$).
- q_i is the number of containers in V_1 after his visit to customer i (the initial number of containers in V_1 when leaving the depot D_1 is $q_{D_1} = Q_{init}$) ;
- $t_{1,i}$ represents the departure time of V_1 from customer i location ($t_{1,0}$ represents de departure time of V_1 from the depot D_1) ;
- $t_{2,i}$ represents the departure time of V_2 from customer i location ($t_{2,0}$ represents de departure time of V_2 from the depot D_2).

The objective is to find two optimized vehicle tours T_{V_1} and T_{V_2} minimizing the total travel times, such that T_{V_1} is a pickup & delivery tour through n customers, and T_{V_2} is a concrete delivery tour through the n_d customers who have received containers. Thus, we consider the following objective function :

$$min \quad \sum_{i=0}^{n} \sum_{j=0}^{n} x_{i,j} c_{i,j} + \sum_{i=0}^{n} \sum_{j=0}^{n} y_{i,j} c_{i,j} \tag{1}$$

Subject to :

$$\sum_{j \in N} x_{i,j} = 1 \qquad \forall i \in \{D_1\} \cup N \tag{2}$$

$$\sum_{i \in N} x_{i,j} = 1 \qquad \forall j \in \{D_1\} \cup N \tag{3}$$

$$\sum_{j \in N_d} y_{i,j} = 1 \qquad \forall i \in \{D_2\} \cup N_d \tag{4}$$

$$\sum_{i \in N_d} y_{i,j} = 1 \qquad \forall j \in \{D_2\} \cup N_d \tag{5}$$

$$x_{i,D_2} = 0 \qquad \forall i \in \{D_1\} \cup N \tag{6}$$

$$x_{D_2,i} = 0 \qquad \forall i \in \{D_1\} \cup N \tag{7}$$

$$y_{i,j} = 0 \qquad \forall i, j \in \{D_1\} \cup N_p \tag{8}$$

$$x_{i,j}(q_i + d_j - q_j) = 0 \qquad \forall i, j \in \{D_1\} \cup N \tag{9}$$

$$q_i \leq Q \qquad \forall i \in \{D_1\} \cup N \tag{10}$$

$$q_i \geq 0 \qquad \forall i \in \{D_1\} \cup N \tag{11}$$

$$q_{D_1} = Q_{init} \tag{12}$$

$$x_{i,j}(t_{1,i} + c_{i,j} - t_{1,j}) = 0 \qquad \forall i, j \in \{D_1\} \cup N \tag{13}$$

$$y_{i,j}(t_{2,i} + c_{i,j} - t_{2,j}) = 0 \qquad \forall i, j \in \{D_2\} \cup N_d \tag{14}$$

$$t_{2,i} \geq t_{1,i} \qquad \forall i \in \{D_2\} \cup N_d \tag{15}$$

$$t_{1,0} = 0 \tag{16}$$

Where :

- Constraints (2) and (3) ensure that each customer is visited exactly once by vehicle V_1, while constraints (4) and (5) ensure that each "delivery customer" is visited exactly once by vehicle V_2 ;
- Constraints (6) and (7) relate to the fact that V_1 cannot visit the depot of V_2, while (8) ensures that V_2 cannot visit neither the depot of V_1 nor the "pickup customers" ;
- Constraints (9) to (12) are related to vehicle capacity. If customer j is visited immediately after customer i ($x_{i,j} = 1$), then, the condition $q_j = q_i + d_j$ must be satisfied. Furthermore, q_i must be smaller then Q and greater than 0 ;
- Constraint (13) and (14) concern the computing of departure times of V_1 and V_2 from each customer's location. Thus, if customer j is visited by vehicle m immediately after customer i, then $t_{m,j} = t_{m,i} + c_{i,j}$;
- Constraint (15) concern the temporal precedence between V_1 and V_2. The vehicle V_2 cannot arrives at a customer's location before V_1. In other words, $t_{2,i} \geq t_{1,i}$.

Note that Picked up containers can be supplied to a delivery customer if necessary.

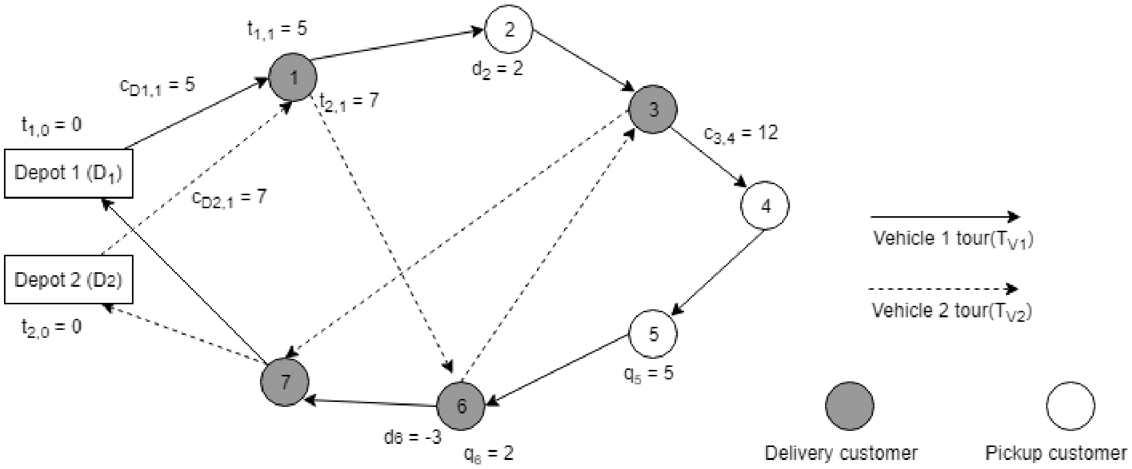


Figure 1: Synchronized vehicle tours.

4 PROPOSED HEURISTIC

To tackle to problem described above, we propose a two-step heuristic :

(1) We generate a feasible pickup & delivery tour for the vehicle V_1 (T_{V_1}) using the local search approach described below ;
(2) Then, we build a tour for V_2 (T_{V_2}) taking T_{V_1} as a strong constraint.

4.1 Generating the pickup & delivery tour

The pickup & delivery problem tackled here is the one-commodity pickup & delivery traveling salesman problem. We have a single vehicle (V_1) picking up or delivering a single type of commodity (empty containers). A picked up container can be supplied to another customer during a tour, and the vehicle has a maximum capacity that cannot be exceeded during a tour.
We propose a local search method which starts from an initial feasible solution S, and tries to improve it by moving to S', a feasible neighbouring solution of S, such that $f(S') < f(S)$, where $f(S)$ is the total travel time of V_1. The process is repeated until no improvement can be found.

4.1.1 Neighbourhood Structure. We use the 1-shift algorithm introduced in [2] to generate the neighbourhood of a given solution S. This method consists in changing the position of a customer in a tour from i to j. Customers who are in positions $i+1, i+2, ..., j$ of the tour are then shifted backwards (see Fig. 2).

4.1.2 Feasibility Checking. For each generated solution, we ensure that capacity constraints are respected. A feasible solution is a tour in which the number of containers loaded on the vehicle V_1 never exceeds its maximum capacity Q, and is never negative. Fig.2 presents an example of a feasible and an infeasible solution.
Given a feasible solution S and a 1-shift neighbouring solution S' of S obtained by shifting a customer from position i to j. It can easily be shown that S' is feasible if and only if the partial tour from customer i to customer j is feasible. Indeed, to check the feasibility of a neighbouring solution, we only check the feasibility of the tour between position i and position j.

4.2 Generating the mixer truck tour

Once the pickup & delivery tour for the vehicle V_1 is generated, we build a second tour for V_2 considering the first one as a strong constraint. Thus, starting from D_2, the idea is to choose, at each iteration of the procedure, the next customer to be visited. So, as it appears in Fig 3, among all customers who require a visit of V_2 :

- We identify those who can be visited by V_2 after the departure of V_1. In other words, when V_2 is at customer i location, we calculate $t_{2,i} + c_{i,j}$ for each customer j who requires a visit. We choose the next customer from those for whom $t_{2,i} + c_{i,j} \geq t_{1,j}$ (temporal precedence constraint) ;
- Among all the customers for whom the temporal precedence constraint is respected, we choose the nearest one (in terms of travel time) from the current position of the vehicle ;
- This procedure is repeated until all the customers are visited.

5 COMPUTATIONAL RESULTS

The approach described above was implemented in Java, and executed on AMD A10-7700K Radeon R7, 3.40 GHz With 8 GB RAM.
To the best of our knowledge, there is no benchmark instances for simultaneous vehicle routing problems with pickup & delivery. Therefore, we tested our algorithm on the Euclidian PDTSP instances generated by [6], which consider a single depot for each instance. The number of customers varies between 25 and 200. We adapted the instances to fit our constraints by considering the depot and the first customer of each instance as the depots of the two vehicles considered in our problem.
Table 1 shows the average results obtained by the pickup & delivery tour and the mixer truck tour. Pickup & delivery tours are more costly because they involve more customers. In the other hand, they are more flexible since they are not subject to temporal precedence constraint, contrary to mixer truck tours. Therefore, we can hope to obtain better results when focusing on the optimization of the pickup & delivery tours.

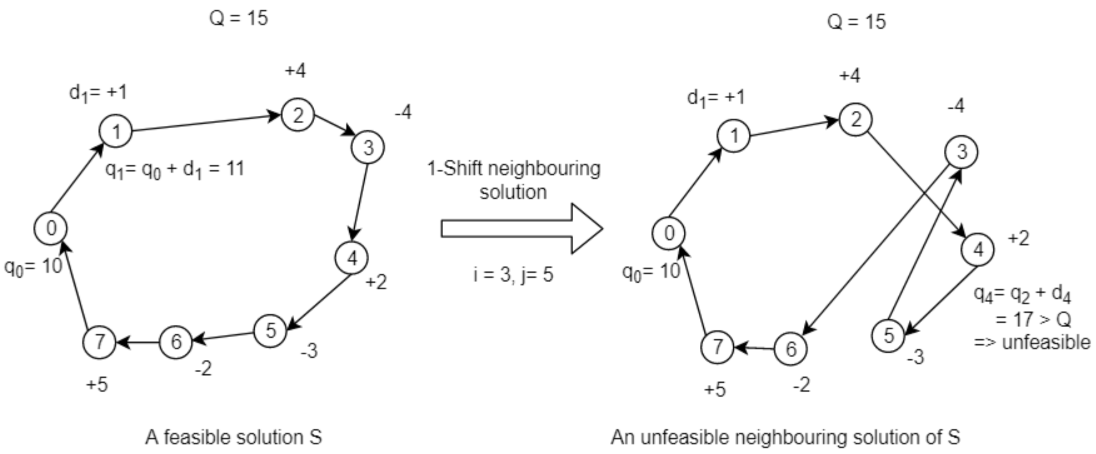

Q = 15
d_1= +1
+4
-4
q_1= q_0 + d_1 = 11
0
1
2
3
4
5
6
7
q_0= 10
+2
-3
-2
+5
A feasible solution S
1-Shift neighbouring solution
i = 3, j= 5
Q = 15
d_1= +1
+4
-4
+2
q_0= 10
q_4= q_2 + d_4 = 17 > Q => unfeasible
-3
-2
+5
An unfeasible neighbouring solution of S

Figure 2: 1-Shift algorithm

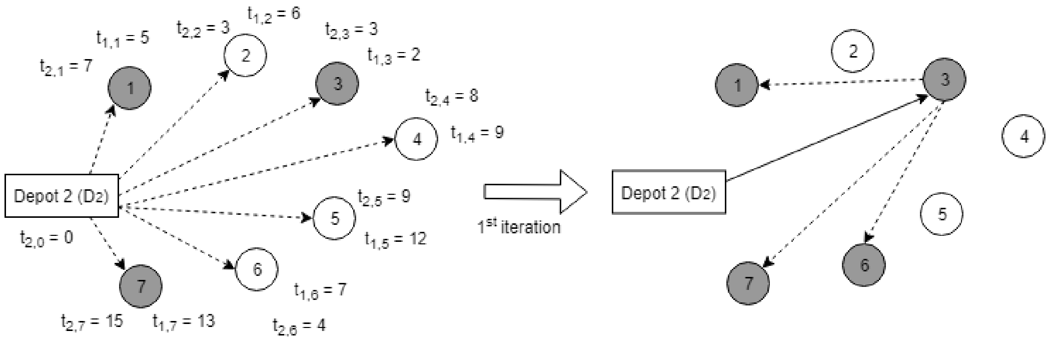


Figure 3: Building a mixer truck tour

Table 1: Average results on the Euclidian PDTSP instances

Number of customers	Pickup & delivery tour	Mixer truck tour
25	564.36	363,19
50	871.33	573
75	1143.1	779.24
100	1429.71	1001.41
150	2019.18	1613.86
200	2704.91	1813.22

6 CONCLUSIONS

We presented an approach to tackle a vehicle routing problem with pickup & delivery and synchronization constraint. This approach is a two-step heuristic. We start by generating a pickup & delivery tour for a first vehicle respecting vehicle capacity constraint. Then, we construct a second tour according to the first one for another vehicle, respecting temporal precedence constraint. The objective function considered is the minimization of the total travel times.

We tested our algorithms on the Euclidian PDTSP instances proposed in [6]. We adapted the instances to fit our constraints and collected the results, which were positive.

Future works will be devoted to the implementation of the ILP model proposed in this paper and the development of other approaches exploiting other types of heuristics, and including other constraints such as time windows, multiple vehicles...

REFERENCES

[1] Gerardo Berbeglia, Jean-François Cordeau, Irina Gribkovskaia, and Gilbert Laporte. 2007. Static pickup and delivery problems: a classification scheme and survey. *Top* 15, 1 (2007), 1–31.

[2] Dimitris Bertsimas. 1988. *Probabilistic combinatorial optimization problems.* Ph.D. Dissertation. Massachusetts Institute of Technology.

[3] Jeng-Fung Chen and Tai-Hsi Wu. 2006. Vehicle routing problem with simultaneous deliveries and pickups. *Journal of the Operational Research Society* 57, 5 (2006), 579–587.

[4] Jean-François Cordeau. 2006. A branch-and-cut algorithm for the dial-a-ride problem. *Operations Research* 54, 3 (2006), 573–586.

[5] Jean-François Cordeau and Gilbert Laporte. 2003. A tabu search heuristic for the static multi-vehicle dial-a-ride problem. *Transportation Research Part B: Methodological* 37, 6 (2003), 579–594.

[6] Michel Gendreau, Gilbert Laporte, and Daniele Vigo. 1999. Heuristics for the traveling salesman problem with pickup and delivery. *Computers & Operations Research* 26, 7 (1999), 699–714.

[7] Irina Gribkovskaia, Øyvind Halskau sr, Gilbert Laporte, and Martin Vlček. 2007. General solutions to the single vehicle routing problem with pickups and deliveries. *European Journal of Operational Research* 180, 2 (2007), 568–584.

[8] H Hernández-Pérez. 2004. Traveling salesman problems with pickups and deliveries. *Disertation, University of La Laguna, Spain* (2004).

[9] Hipólito Hernández-Pérez, Inmaculada Rodríguez-Martín, and Juan José Salazar-González. 2009. A hybrid GRASP/VND heuristic for the one-commodity pickup-and-delivery traveling salesman problem. *Computers & Operations Research* 36, 5 (2009), 1639–1645.

[10] Hipólito Hernández-Pérez and Juan-José Salazar-González. 2004. A branch-and-cut algorithm for a traveling salesman problem with pickup and delivery. *Discrete Applied Mathematics* 145, 1 (2004), 126–139.

[11] Hipólito Hernández-Pérez and Juan-José Salazar-González. 2004. Heuristics for the one-commodity pickup-and-delivery traveling salesman problem. *Transportation Science* 38, 2 (2004), 245–255.

[12] Arild Hoff and Arne Løkketangen. 2006. Creating lasso-solutions for the traveling salesman problem with pickup and delivery by tabu search. *Central European Journal of Operations Research* 14, 2 (2006), 125–140.

[13] Harilaos N Psaraftis. 1980. A dynamic programming solution to the single vehicle many-to-many immediate request dial-a-ride problem. *Transportation Science* 14, 2 (1980), 130–154.

[14] Paolo Toth and Daniele Vigo. 1996. Fast local search algorithms for the handicapped persons transportation problem. In *Meta-Heuristics*. Springer, 677–690.