# Lagrangian Decomposition Methods for Large-Scale Fixed-Charge Capacitated Multicommodity Network Design Problem

Rui Sa Shibasaki

Universidade Federal de Minas Gerais
Université Clermont-Auvergne

# Lagrangian Decomposition Methods for Large-Scale Fixed-Charge Capacitated Multicommodity Network Design Problem.

By
Rui Sá Shibasaki

A thesis presented for the dual degree of Doctor of Philosophy
Industrial Engineering and Computer Science
September 22nd 2020

| | Jury: |
|---|---|
| Philippe Mahey | Director (Directeur de thèse) |
| Maurício Cardoso de Souza | Director (Directeur de thèse) |
| Mourad Baïou | Co-Director (Co-directeur de thèse) |
| Nelson Maculan | Referee (Rapporteur) |
| Denis Cornaz | Referee (Rapporteur) |
| Jean-Philippe Gayon | President (Président) |
| Hande Yaman | Examiner (Examinatrice) |
| Francisco Barahona | Guest (Invité) |

Ecole Doctorale Sciences pour l'Ingénieur
Université Clermont-Auvergne.
Programa de Pós-Graduação em Engenharia de Produção
Universidade Federal de Minas Gerais.

*To God, and my parents*
*Maria Eudina and Mário.*

# Acknowledgements

The preparation of a Ph.D. thesis was one of the greatest challenges of my life, and such a long and difficult path would not have been possible to be completed without the help of others. First of all, I would like to thank God. Nothing would have been possible in my life without him.

Next, I would like to thank all the people that directly or indirectly contributed to this achievement. I am thankful to my mother, for the positivity, spiritual guidance, and enthusiasm. My father, who inspired me so much as a kid, and continually pushed me forward. My brother, for the support and funny moments. Still, I would like to thank all my relatives, for the support and concern.

I also would like to thank my Ph.D. advisors Maurício Cardoso de Souza and Philippe Mahey, for the encouragement, patience, and trust. Moreover, I am sincerely honored to have worked with professors Mourad Baïou and Francisco Barahona. In addition, I could not forget professor Martín Ravetti, to who I am very grateful for the support.

Fortunately, life is not just working, and relaxing times are always helpful to get through Ph.D. studies. In that sense, I am very thankful to all my friends, starting with the ones in Belo Horizonte, especially Matheus Meinicke and Camila Piovesana, who have intensively participated during this time. I also have to mention Matheus Alves, Julio Souza, Luiza Camisassa, and the guys from the blues band. Thank you for the times of great joy. Next, I would like to thank all my friends from Clermont-Ferrand, I wish I could name them all, one by one, but I point out especially Ugo Plaisantin, Charly Valet, Pascal Chaix, Pauline Huau, and La Coloc de la rue Fontgième. Great time together, thank you all for having made my life happier during my years in Clermont.

Last but not least, I would like to thank my friends from the BRAFITEC program Guilherme Martino, Fabio Silva, Marco Tulio, Ana Alice Barros, and others, for the incredible times in Belo Horizonte and France, since Masters studies. Furthermore, I cannot forget my dear friends from UFMG, especially Fernanda Alves, thank you for the happy times and shared frustrations. It is always great to feel you are not alone. I also would like to thank Rafael Colares, who has been

a friend since undergraduate studies, sharing life and work experiences, thank you a lot. Finally, I would like to thank Mathieu Gondran, who I am very glad having met at LIMOS. To conclude, I would like to thank Fapemig for the financial support, and Universidade Federal de Minas Gerais, and Université Clermont-Auvergne, places where I have learned a lot.

*'Anyone who has never made a mistake*
*has never tried anything new.'*
**- Albert Einstein**

# Abstract

Typically present in logistics and telecommunications domains, the Fixed-Charge Multicommodity Capacitated Network Design Problem remains challenging, especially when large-scale contexts are involved. In this particular case, the ability to produce good quality solutions in a reasonable amount of time leans on the availability of efficient algorithms. In that sense, the present thesis proposed Lagrangian approaches that are able to provide relatively sharp bounds for large-scale instances of the problem. The efficiency of the methods depends on the algorithm applied to solve Lagrangian duals, so we choose between two of the most efficient solvers in the literature: the Volume Algorithm and the Bundle Method, providing a comparison between them. The results showed that the Volume Algorithm is more efficient in the present context, being the one kept for further research.

A first Lagrangian heuristic was devised to produce good quality feasible solutions for the problem, obtaining far better results than Cplex, for the largest instances. Concerning lower bounds, a Relax-and-Cut algorithm was implemented embedding sensitivity analysis and constraint scaling, which improved results. The increases in lower bounds attained 11%, but on average they remained under 1%.

The Relax-and-Cut algorithm was then included in a Branch-and-Cut scheme, to solve linear programs in each node of the search tree. Moreover, a Feasibility Pump heuristic using the Volume Algorithm as solver for linear programs was implemented to accelerate the search for good feasible solutions in large-scale cases. The obtained results showed that the proposed scheme is competitive with the best algorithms in the literature, and provides the best results in large-scale contexts. Moreover, a heuristic version of the Branch-and-Cut algorithm based on the Lagrangian Feasibility Pump was tested, providing the best results in general, when compared to efficient heuristics in the literature.

**Keywords: Multicommodity Network Design, Lagrangian Relaxation, Volume Algorithm, Bundle Method, Relax-and-Cut, Branch-and-Cut.**

# Résumé

Typiquement présent dans les domaines de la logistique et des télécommunications, le problème de synthèse de réseau multi-flot à charge fixe reste difficile, en particulier dans des contextes à grande échelle. Dans ce cas, la capacité à produire des solutions de bonne qualité dans un temps de calcul raisonnable repose sur la disponibilité d'algorithmes efficaces. En ce sens, cette thèse propose des approches lagrangiennes capables de fournir des bornes relativement proches de l'optimal pour des instances de grande taille. L'efficacité des méthodes dépend de l'algorithme appliqué pour résoudre les duals lagrangiens, nous choisissons donc entre deux des solveurs les plus efficaces de la littérature: l'algorithme de Volume et la méthode Bundle, fournissant une comparaison entre eux. Les résultats ont montré que l'algorithme de Volume est plus efficace dans le contexte considéré, étant celui choisi pour le développement du projet de recherche.

Une première heuristique lagrangienne a été conçue pour produire des solutions réalisables de bonne qualité pour le problème, obtenant de bien meilleurs résultats que Cplex pour les plus grandes instances. Concernant les limites inférieures, un algorithme Relax-and-Cut a été implémenté intégrant une analyse de sensibilité et une mise à l'échelle des contraintes, ce qui a amélioré les résultats. Les améliorations des bornes inférieures ont atteint 11%, mais en moyenne, elles sont restées inférieures à 1%.

L'algorithme Relax-and-Cut a ensuite été inclus dans un schéma Branch-and-Cut, pour résoudre des programmes linéaires dans chaque nœud de l'arbre de recherche. De plus, une heuristique Feasibility Pump lagrangienne a été implémentée pour accélérer la recherche de bonnes solutions réalisables. Les résultats obtenus ont montré que le schéma proposé est compétitif avec les meilleurs algorithmes de la littérature et fournit les meilleurs résultats dans des contextes à grande échelle. De plus, une version heuristique de l'algorithme Branch-and-Cut basé sur le Feasibility Pump lagrangien a été testée, fournissant les meilleurs résultats en général, par rapport aux heuristiques de la littérature.

**Mots-clés: Synthèse de réseau multi-flot, Relaxation lagrangienne, Algorithme de Volume, Bundle, Relax-and-Cut, Branch-and-Cut.**

# Resumo

Tipicamente presente nas áreas de logística e telecomunicações, o problema de síntese de redes multi-fluxo de custo fixo e capacitada permanece desafiador, especialmente quando contextos de grande escala estão envolvidos. Nesse caso, a capacidade de produzir soluções de boa qualidade em um tempo computacional praticável depende da disponibilidade de algoritmos eficientes. Nesse sentido, a presente tese propõem abordagens lagrangianas capazes de fornecer limites relativamente próximos ao ótimo para instâncias de grande escala. A eficiência dos métodos depende do algoritmo aplicado para resolver duais Lagrangianos, portanto escolhemos entre dois dos solvers mais eficientes da literatura: o Algoritmo de Volume e o Método Bundle, proporcionando uma comparação entre eles. Os resultados mostraram que o Algoritmo de Volume é mais eficiente no contexto considerado, sendo o escolhido para o desenvolvimento do projeto de pesquisa.

Uma primeira heurística Lagrangiana foi desenvolvida para produzir soluções viáveis de boa qualidade para o problema, obtendo resultados muito melhores do que Cplex, para as maiores instâncias. Em relação aos limites inferiores, um algoritmo Relax-and-Cut foi implementado incorporando análise de sensibilidade e uma normalização das restrições, o que melhorou os resultados. Os aumentos nos limites inferiores atingiram 11%, mas em média permaneceram abaixo de 1%.

O algoritmo Relax-and-Cut foi então incluído em um esquema Branch-and-Cut, para resolver programas lineares em cada nó da árvore de busca. Além disso, uma heurística de Feasibility Pump lagrangiana foi implementada para acelerar a busca por boas soluções viáveis. Os resultados obtidos mostraram que o esquema proposto é competitivo com os melhores algoritmos da literatura, e fornece os melhores resultados em contextos de larga escala. Além disso, foi testada uma versão heurística do algoritmo Branch-and-Cut baseado no Feasibility Pump lagrangiano, proporcionando os melhores resultados em geral quando comparada às heurísticas mais eficientes da literatura.

**Palavras-chave: Síntese de rede multi-fluxo, Relaxação lagrangiana, Algorithmo de Volume, Método de Bundle, Relax-and-Cut, Branch-and-Cut.**

# List of Figures

# List of Tables

# Contents

# Chapter 1

# Introduction

A classic topic in combinatorial optimization, network design problems have been attracting much interest for almost six decades in the Operational Research community, according to Minoux [171]. In those problems, networks are usually modeled by graph structures containing nodes and arcs, so design decisions need to be made in order to satisfy a set of requirements, in such a way as to optimize a given function. Here we are especially interested in cases where flow requirements are involved, whereas problems like the Minimum Spanning Tree, Steiner Tree, and Facility Location can also be considered as network design problems. Generally, integer variables are used to model network decisions, which results in *integer programs* (IP), or *mixed integer programs* (MIP).

Typically NP-hard [136, 164], this class of problems arises in various levels of decision-making processes in logistics, transportation, and telecommunications sectors (see [6, 5, 171, 164]). In freight transportation systems, one may have to deal with network problems involving infrastructure, selection of transportation services, and their frequencies [60]. Airline companies, for example, may have to deal with service network design (see [60] for a review on the subject), having to determine the routing pattern for planes and the frequency of flights, given the aircraft and crew availability [135, 154], to satisfy the demand. Similarly, companies like UPS and FedEx have tried to optimize the design of their service networks for delivering express packages [11, 10]. In those contexts, networks may be represented by time-space graphs, each node being a certain moment in a certain location, with the arcs meaning the possible movements.

Moreover, with the advances in manufacturing, and just-in-time supply chain management, the network design has a big impact on logistics activities, in both operational and economical aspects [55]. Recently in the telecommunications field, network design problems arose while planning fiber optical networks [212]. Other applications in telecommunications can be found in the survey [98].

The present work addresses the *fixed-charge multicommodity capacitated network design problem*. In this case, arcs represent facilities (optical fibers, electric lines, roads, pipelines, etc.), each with a given capacity, that may be installed to accommodate the flow of multiple commodities, notably data packets, physical goods, people, etc., given the origin-destination pairs of points [57]. Particularly, only one type of facility is available, and at most one facility is installed per arc.

Given its complexity and characteristics, large-scale instances of the problem present a great number of variables and constraints, which may be very time and memory consuming, especially when approaching it with linear programming methods. Indeed, for some of the instances tested here, even the linear relaxation program (LP) took days to be solved by Cplex (a state-of-the-art linear programming software). In that sense, we tend to look for relaxations and decomposition methods that may help to find a near-optimal feasible solution. In other words, we search for good-quality feasible solutions, and tight bounds to measure their distance from the optimal.

In the literature, much effort has been focused on heuristic approaches to determine near-optimal solutions. However, a bounding procedure is often missing in those kinds of methods. In this regard, cutting-plane algorithms have been proposed, but at the expense of further increasing the size of linear programs. Other methods like Benders decomposition [31] and Dantzig-Wolfe decomposition [69] were applied with relative success [57], but on instances not as large as the ones considered here.

Alternatively, schemes based on Lagrangian relaxation have also been used to generate tight bounds for difficult large scale problems. The early works about Lagrangian relaxation applied to combinatorial problems were published by Held and Karp, for the Traveling Salesman problem [121, 122], and a general theory was developed by Geoffrion in his seminal paper [109]. Since then, such an approach has been proved to be an interesting alternative to deal with hard integer (or mixed-integer) programs, for example in scheduling [84, 85], assignment problems [194, 138], and facility location problems [39, 75]. Other applications can be found in [118, 86], and the references therein.

Given the promising results obtained with Lagrangian schemes in network design and other fields, we developed solution methods for the considered problem, having the Lagrangian relaxation as a central feature. Indeed, it was observed that the use of such a technique was crucial to devise efficient algorithms, especially in terms of time consumption when very large-scale instances were involved.

This thesis has the main goal of producing tight bounds for very large-scale instances of the fixed-charge multicommodity capacitated network design prob-

lem, with the help of efficient Lagrangian schemes. The contributions of this thesis include: *i*) a comparison between two state-of-the-start nondifferentiable optimization algorithms: Volume [26] and Bundle [155] methods, in the context of Lagrangian dual optimization; *ii*) a Lagrangian heuristic capable of providing near-optimal feasible solutions to large-scale instances; *iii*) a relax-and-cut algorithm, with two strategies to enhance performances, along with a new heuristic procedure for cutset separation; and *iv*) a Volume-based branch-and-cut algorithm, combined with a Lagrangian Feasibility Pump heuristic [80].

The following five chapters present the studies carried out during the doctoral program. In chapter 2, the considered problem is presented along with a literature review about solution approaches and closely related problems. A brief discussion about some characteristics of hard instances is presented, based on numerical experiments. At the end of that chapter, the most relevant types of cut inequalities and constraint relaxations for the problem are discussed.

Chapter 3 presents Lagrangian relaxations for the problem, along with a comparison between two state-of-the-art nonsmooth (nondifferentiable) optimization solvers: the Volume Algorithm and the Bundle Method. In fact, the Lagrangian decomposition methods implemented here are based on the fact that, by Lagrangian relaxation, the problem can be decomposed into simpler smaller subproblems. However, such relaxation results in the Lagrangian dual problem, which is concave and generally nonsmooth. In that sense, the choice of efficient nonsmooth optimization algorithms is crucial for the development of efficient Lagrangian schemes. In the same chapter, the Lagrangian heuristic is proposed. A paper entitled 'Lagrangian bounds for large-scale multicommodity network design: a comparison between Volume and Bundle methods' [199] was published in the 'International Transactions in Operational Research' journal, containing the main results presented in the chapter.

Since the Volume Algorithm proved to behave better than the Bundle Method in most large-scale instances, in chapter 4, a Volume-based Relax-and-Cut algorithm is proposed. Sensitivity analysis and constraint scaling are used to improve performances, which indeed provided better results. In chapter 5, the Relax-and-Cut algorithm is then included in a Branch-and-Cut scheme, in an attempt to close optimality gaps. A Volume-based Feasibility Pump heuristic is proposed to accelerate finding better feasible solutions for large-scale instances. Finally, a conclusion closes the thesis in chapter 6, presenting the next steps and suggesting future research.

# Chapter 2

# Fixed-Charge Multicommodity Capacitated Network Design

In this chapter, the Fixed-Charge Multicommodity Capacitated Network Design problem is presented, along with a literature review and an initial discussion about complicating features of the problem. In section 2.1, the problem is described and formulations are presented, followed by the discussion about the problem instances in section 2.2. The literature review is presented in section 2.3, and relaxations and valid inequalities for the problem are given in section 2.4. Finally a conclusion is made in section 2.5.

## 2.1 Introduction

The Fixed-Charge Multicommodity Capacitated Network Design Problem (FCMC) consists in determining a minimal cost network, with enough installed capacity to satisfy the multicommodity flow demands. In the present case, the cost function includes, for each arc and commodity, transportation costs and installation costs on each arc, the latter being associated with a single facility of given capacity.

The FCMC is usually modeled as a MIP, based on either arcs or paths. Considering that all costs, demands, and capacities are nonnegative real values, for a given directed graph $G = (N, A)$, $N$ being the set of nodes, and $A$ the set of arcs, the capacities for each $a \in A$ are denoted by $w_a$, and fixed costs represented by $f_a$. Moreover, for a given set $K$ of commodities, $O(k)$, $D(k)$ and $q^k$ denote respectively, for each $k \in K$, the origin, the destination, and the demand quantity. Finally, let transportation costs for each commodity $k \in K$ on arc $a \in A$ be denoted by $c_a^k$.

For the arc-based formulation we introduce variables $x_a^k \geq 0$ for the amount

of flow $k \in K$ in the arc $a \in A$, and binary variables $y_a$ for all $a \in A$, $y_a = 1$ if the single facility is installed, $y_a = 0$ otherwise. Given that an arc $a = (i, j)$ for $i, j \in N$, $N_i^+ = \{j \in N | (i, j) \in A\}$ is the set of nodes $j$ having an arc arriving from node $i$, and $N_i^- = \{j \in N | (j, i) \in A\}$ is the set of nodes $j$ having an arc arriving into node $i$. The values $b_a^k = min\{w_a, q^k\}$ define upper bounds on the commodity flow $k \in K$ through arc $a \in A$. The model is presented as follows [164]:

$$\text{Minimize} \sum_{k \in K} \sum_{a \in A} c_a^k x_a^k + \sum_{a \in A} f_a y_a \tag{2.1}$$

$$\sum_{j \in N_i^+} x_{ij}^k - \sum_{j \in N_i^-} x_{ji}^k = \begin{cases} q^k, & \text{if} \quad i = O(k) \\ -q^k, & \text{if} \quad i = D(k) \\ 0, & \text{otherwise} \end{cases} \quad \forall i \in N, k \in K \tag{2.2}$$

$$\sum_{k \in K} x_a^k \leq w_a y_a \qquad \forall a \in A \tag{2.3}$$

$$x_a^k \leq b_a^k y_a \qquad \forall a \in A, k \in K \tag{2.4}$$

$$x_a^k \geq 0 \qquad \forall a \in A, k \in K \tag{2.5}$$

$$y_a \in \{0, 1\} \qquad \forall a \in A \tag{2.6}$$

Alternatively, in the path-based formulation the multicommodity flow is modeled by variables representing commodity paths, namely a sequence of arcs $p = \{a_1, \ldots, a_l \ := \ (O(k), j_1), (j_1, j_2), \ldots, (j_{l-1}, D(k))\}$, $k \in K$. Likewise, variables $x_p^k \geq 0$ represent the amount of flow of $k$ in path $p \in P(k)$, the set of all possible paths for commodity $k$.

$$\text{Minimize} \sum_{k \in K} \sum_{p \in P(k)} c_p^k x_p^k + \sum_{a \in A} f_a y_a \tag{2.7}$$

$$\sum_{p \in P(k)} x_p^k = q^k \qquad \forall k \in K \tag{2.8}$$

$$\sum_{k \in K} \sum_{p \in P(k): a \in p} x_p^k \leq w_a y_a \qquad \forall a \in A \tag{2.9}$$

$$\sum_{p \in P(k): a \in p} x_p^k \leq b_a^k y_a \qquad \forall a \in A, k \in K \tag{2.10}$$

$$x_p^k \geq 0 \qquad \forall k \in K, p \in P(k) \tag{2.11}$$

$$y_a \in \{0, 1\} \qquad \forall a \in A$$

The equations (2.2) and (2.8) guarantee the conservation of flow in the network, and inequalities (2.3) and (2.9) define capacity constraints. One can note that the *strong forcing constraints* (2.4) and (2.10) are redundant for the MIPs, but they considerably increase the quality of the lower bound when solving their linear relaxation [105]. Indeed, according to Cornuejols *et al.* [56] those constraints define facets of the convex hull of (2.12), derived from the arc-based formulation

(a similar construction can be made using the path-based formulation). Finally, the domain of the variables are defined in (2.5), (2.11) and (2.6).

$$\{x, y \in \mathbb{R}_+^{|A||K|} \times \mathbb{B}^{|A|} \mid (2.3),\ x_a^k \leq q^k \quad \forall a \in A, k \in K\} \tag{2.12}$$

The relation between the two formulations was studied by Tomlin [204], who stated they are equivalent, and that the path-based model may be viewed as a Dantzig-Wolfe decomposition approach (see [69]) for the arc-based formulation. Such relation was further explored by Jones *et al.* [137]. Nevertheless, it is stated in [105] that in case the problem is uncapacitated, the linear relaxation of the arc-based formulation provides tighter bounds than the path-based one. In addition, works on the polyhedral structure of each model have been published by Balakrishnan *et al.* [20], for the path-based formulation, and by Bienstock and Günlük [38], for the arc-based one. Alternative formulations based on metric inequalities [18] and on cutsets [25], have also been considered when flow costs are not present.

The feasibility of FCMC is completely linked to the existence of a feasible multicommodity flow in the considered network, for the set of flow requirements. Onaga and Kakusho [180] (see also [134]) have discussed propositions on feasibility conditions of multicommodity flows, showing that a multicommodity flow of a given set of demands is feasible if and only if, for every cutset, the capacity is greater or equal to the transpassing flow requirements. More works addressing the feasibility of multicommodity network flows were presented for special cases like the two-commodity case [195, 132, 196], and the case where a planar graph is considered [178, 120].

In summary, two sources of infeasibility may occur: the first related to connectivity and the second related to a lack of capacity, both traduced in the metric inequality (2.13), in which $\alpha_a \geq 0,\ a \in A$ represent the dual values of capacity constraints, and $\pi^k_{D(k)},\ k \in K$, the values of the lengths of the shortest paths between $O(k)$ and $D(k)$, having $\alpha$ as arc lengths.

$$\sum_{a \in A} \alpha_a w_a \geq \sum_{k \in K} q^k \pi^k_{D(k)} \tag{2.13}$$

In that sense, a multicommodity flow is feasible if and only if (2.13) are satisfied for all $\alpha \geq 0$, considering that if no path exists between $O(k)$ and $D(k)$, for some $k \in K$, then $\pi^k_{D(k)}$ is set to infinity. Costa *et al.* [58], defined metric inequalities as a special case of Benders inequalities.

Finally, remark that if a design configuration (topology) is provided, fixing binary variables to 1 or 0, the FCMC is resumed to a Minimum Capacitated Multicommodity Flow (MCMF) problem (see [12, 146] for surveys on the subject).

In that sense, FCMC is feasible if considering a configuration $y_a = 1, \forall\ a \in A$, the corresponding MCMF problem is feasible. Moreover, in the remainder of this document, we say that any given topology is feasible if and only if the respective MCMF problem is feasible too. Such considerations are widely used throughout this document.

Although real-life network design problems often require additional features to be added to the formulation, the FCMC as described above is sufficiently challenging. With *NP-Hard* problems like the *Falicity Location*, as special cases [164, 136], the presented problem is *NP-Hard* itself. Moreover, it contains interesting decomposable substructures, likely to be present in practical situations, and that are quite suitable to the current purpose.

The surveys written by Magnanti & Wong [164], Minoux [171] and Ahuja *et. al.* [6] present a number of practical applications in which the FCMC is involved, as well as similar problems related to network design aspects. Some applications, specifically in the field of rail freight transportation planning, can be found in [208]. Next, a discussion about the difficulty of instances is presented, followed by the benchmark instances most used currently in the literature.

## 2.2 Hard instances

The discussion about difficult features of an instance is itself a complete research area, and an extensive literature on the subject exists for combinatorial problems like the ones surveyed in [201]. More closely related to the FCMC, papers have been published about hard instances for the *knapsack problem* [52, 51, 187, 139], and the facility location problem [152]. In addition, Bienstock [37] has discussed the difficulties while having to deal with a network design problem where node degrees are imposed. However, a clear characterization of hard instances of the FCMC problem is still not available in the literature.

Even though the size of instances may be a difficult aspect at a first sight, other properties may be very complicating, making small-sized instances very challenging too. In this subsection, we try to elucidate some characteristics of a potentially hard instance, while being solved by a state-of-the-art MIP solver. We stress that this is a preliminary numerical experiment and further research is needed on the subject.

In that manner, a series of instances considering complete digraphs of $n = 10$ nodes (and 90 arcs) were conceived with a maximum of 90 commodities (the complete pattern of commodities). On those digraphs, no parallel arcs are allowed, and commodities do not share the same origin-destination pair. The values of

transportations costs, fixed costs, and demands are randomly generated inside an interval given as a parameter. In their turn, arc capacities are set to a fraction $1/u$ of the sum of commodity demands that may pass in that arc, $u$ being a parameter. The origin-destination commodity pairs are generated according to a given parameter $d$, such that in a matrix of binary values $K = [k_{ij}]_{n \times n}, i, j = 1, ..., n$ each column and row have exactly $d$ items equal to 1, while the others are null (the same procedure is adopted for arcs, in [152]). Then each item $k_{ij} = 1$ correspond to a commodity with origin $i$ and destination $j$. Note that $d = 5$ correspond to a number of 50 commodities, $d = 3$ to 30 commodities, and so on.

The instances are solved by the Branch-and-Cut algorithm implemented in Cplex (version 12.8), with all parameters set as default, and a time limit of 24 hours, running in a single thread on an Intel Xeon E7-8890 v3, 2.50 GHz Linux machine, with 30Gb of RAM. The results are shown in Tables 2.1, 2.2 and 2.3, where column '$|K|$' presents the number of commodities of instances in that line (10 instances per line), 'Num Opt' presents the number of instances solved to optimality, 'Avg Gap' presents the final gap on average provided by Cplex, and 'Avg Nodes' presents the average number of evaluated nodes in the Branch-and-Cut tree.

The first testbed aimed at verifying the impact of parameter $u$ to the hardness of an instance. At this first moment, high values of fixed costs were considered, in the interval $[1e5, 2e5]$, while low values of transportation costs were selected, in the interval $[1, 15]$. In their turn, commodity demands were generated in the interval $[1, 10]$.

As shown in the first three lines of Table 2.1, as $u$ increases tightening the arc capacities, the number of nodes needed to solve instances increases too, resulting in more computational time, and in a reduced number of instances solved to optimality. However, at a certain point of tightness, capacities are so tight that one may expect to open almost all arcs, which makes the problem easier, as one can notice in the last three lines of Table 2.1. Observe too that FCMC solution gets close to the min-max solution when congestion increases (see [35] for details). Moreover, it can be seen in lines with $u = 10$ that the number of commodities also impacts the difficulty of instances, since instances got harder as the number of commodities increased. In the remainder of this subsection, all tests were performed with instances built with $u = 10$.

| | $\|K\|$ | Avg Gap (%) | Avg Nodes | Avg Time (sec) | Num Opt |
|---|---|---|---|---|---|
| $u = 2$ | 50 | 0.0 | 2436 | 132 | 10 |
| $u = 5$ | 50 | 0.0 | 189382 | 23450 | 10 |
| $u = 10$ | 50 | 2.9 | 828443 | 70241 | 2 |
| $u = 10$ | 30 | 0.0 | 185417 | 6558 | 10 |
| $u = 10$ | 90 | 4.4 | 318208 | 86400 | 0 |
| $u = 50$ | 90 | 0.3 | 796370 | 61895 | 5 |
| $u \geq 60$ | 90 | 0.0 | 29363 | 1637 | 10 |

Table 2.1: Impact of arc tightness

Provided the same cost and demand intervals, in a second moment we verified the impact of fixed-charges as a complicating feature. In that sense, we test the impact of having some of the arcs with a reduced fixed-charge. So given a parameter value $fc$, and considering the same procedure as for commodities, each column and row of the $\{0, 1\}$-valued matrix $A = [a_{ij}]_{n \times n}$ will present a $fc$ number of items $a_{ij} = 1$, corresponding to the arc $(i, j)$ for which the fixed-charge is reduced. Values are decreased in two degrees of magnitude.

From the results in Table 2.2, one can observe that in general the presence of small fixed-charges makes instances easier, even for a low frequency as for $fc = 1$ (10 arcs). Note that such statements can be made regardless if we refer to instances of 50 or 90 commodities. In addition, once more we notice that a higher number of commodities implies harder instances.

| | $\|K\|$ | Avg Gap (%) | Avg Nodes | Avg Time (sec) | Num Opt |
|---|---|---|---|---|---|
| $fc = 0$ | 50 | 0.0 | 185417 | 6558 | 10 |
| $fc = 1$ | 50 | 0.0 | 46543 | 1280 | 10 |
| $fc = 5$ | 50 | 0.0 | 39259 | 468 | 10 |
| $fc = 0$ | 90 | 4.4 | 318208 | 86400 | 0 |
| $fc = 1$ | 90 | 1.7 | 237080 | 60227 | 4 |

Table 2.2: Impact of fixed charges grid pattern

In a third moment, we tested the impact of intervals in which cost values were generated. The results are presented in Table 2.3, in which three additional columns were added, presenting the intervals for fixed-charges ('Fixed Charge'), transportation costs ('Transp Cost') and demands ('Demands').

It can be observed in Table 2.3 that a generalized reduction on fixed-charges may lead to easier instances since, from the first to the second line, the fixed-charge on all arcs have lost two degrees of magnitude, which resulted in less hard instances. Moreover, observe in the two lines of the middle that the relation

between transportation costs and fixed-charge may be important to define how hard an instance may be. Given the same interval of fixed-charges, instances with bigger demands and transportation costs were easier to be solved. As a matter of fact, that could be expected, since transportation costs are given per unit of flow, so as those costs and the amount of flow become bigger, more importance is given to the linear part of the objective function, rather than to the design (integer) part.

| Fixed Charge | Transp Cost | Demands | $\|K\|$ | Avg Gap (%) | Avg Nodes | Avg Time (sec) | Num Opt |
|---|---|---|---|---|---|---|---|
| [1e5, 2e5] | [1, 15] | [1,10] | 90 | 4.4 | 318208 | 86400 | 0 |
| [1e3, 2e3] | [1, 15] | [1,10] | 90 | 2.4 | 307373 | 73433 | 2 |
| [3e5, 5e5] | [100,500] | [100,500] | 90 | 0.2 | 134519 | 23998 | 8 |
| [3e5, 5e5] | [1,200] | [1,200] | 90 | 4.9 | 321538 | 86400 | 0 |
| [2e5, 1e6] | [1,100] | [1,100] | 90 | 0.1 | 71858 | 20608 | 9 |
| [2e5, 3e5] | [1,100] | [1,100] | 90 | 6.9 | 308976 | 86400 | 0 |

Table 2.3: Impact of cost intervals

In the fifth line, even though fixed-charges could attain the biggest overall values of the testbed, the instances in that configuration were the easiest to be solved. However, note that the range of values in [2e5, 1e6] is much larger, thus the results in that line correspond indeed to what was perceived in Table 2.2, since with a larger interval, instances may present arcs with fixed-charge much smaller than others. That is reinforced by results in the last line, where the upper bound on fixed-charges was significantly reduced and the difficulty increased.

Finally, remark that in fact, even relatively small FCMC instances may be challenging to one of the best solvers in the market. For several instances, Cplex could not solve them before 24 hours of computation. In the following subsection benchmark instances currently used in the literature are presented. In the past, other instances were considered to be hard, but with the advances in computational resources and software implementations, they no longer can be considered in that way [43, 35].

### 2.2.1 Benchmark instances

With the characteristics discussed above, medium to large-sized randomly generated problem instances have been used in the literature. Available for download at `http://www.di.unipi.it/optimize/Data/MMCF.html`, a more detailed description can be obtained in [103, 104, 61]. In short, for a given number $|N|$ of nodes, a given number $|A|$ of arcs, and a given number $|K|$ of commodities, two

nodes are randomly connected until the number of arcs is achieved, parallel arcs not allowed. A similar procedure is adopted for the commodities. In addition, the origin of a commodity is never equal to its destination.

Costs, capacities, and demands are uniformly distributed in intervals given also as parameters. However, costs and capacities are recomputed in order to obtain different difficulty levels among the instances. Two ratios are used to do so: one for the capacities, $C$, and another for fixed costs, $F$. Only integer values are generated.

$$C = |A| \sum_{k \in K} q^k / \sum_{(i,j) \in A} w_{ij}$$

$$F = |K| \sum_{(i,j) \in A} f_{ij} / (\sum_{k \in K} q^k \sum_{k \in K} \sum_{(i,j) \in A} c_{ij}^k)$$

In general, for low values of $C$, the network is lightly constrained, but it becomes more congested as $C$ increases. Moreover, for low values of $F$, fixed costs lose relevance in the objective function, while the converse happens if $F$ is sufficiently increased. Benchmark instances are divided into three groups: Canad-C, Canad-R, and Canad-N, with a number of 43, 153, and 48 instances in each group respectively.

## 2.3 Literature review

In this section we first present exact approaches proposed in the literature for the FCMC, followed by problems closely related to it. At the end of the section, metaheuristics previously implemented for the problem are presented. Here, the focus is on multicommodity flows related to fixed charge design aspects. However, the field of network design is much larger, and we refer the reader to the work of Contreras and Fernández [54], for a wider view on the network design subject.

### 2.3.1 Exact approches

Even though interesting results have been obtained with metaheuristics, finding near-optimal solutions to hard large-scale instances of the FCMC is still very challenging. In a further effort to tackle these instances exact approaches have been proposed, some of them combined with heuristic features.

One of the most widely used methods to solve mixed-integer programs, the Branch-and-Bound [174, 153], was considered by Holmberg and Yuan [130] to solve the considered problem. In essence, in a tree-like structure, the method repeatedly partitions the space of feasible solutions into smaller subsets, and computes bounds on the cost of solutions of each subset. As the subsets become smaller and smaller,

the tendency is for the bounds to get tighter, and converge to an optimal solution. In the work presented by Holmberg and Yuan, a Lagrangian heuristic was used to produce upper bounds, while lower bounds were obtained through Lagrangian relaxation. The results showed that the method struggled to solve large instances, but a heuristic procedure of variable fixing managed to produce good quality solutions quickly. In [129], the authors applied the same procedure to other similar network design problems.

Gendron *et al.* [105, 103, 104] discussed relaxation techniques for the considered problem, including types of linear constraint relaxations (see section 2.4) and Lagrangian relaxations (see section 3.4). A theoretical comparison between two different Lagrangian relaxations is made in [102]. Crainic *et al.* [61] applied the Bundle Method (section 3.3) to solve the Lagrangian dual programs of those same two relaxations. In addition, decomposition methods have been applied to obtain tight bounds for the FCMC through Lagrangian relaxation in [94, 96].

Polyhedral approaches were presented by Chouman *et al.* [46, 47, 48, 49]. In their work, a series of cutting-plane algorithms were implemented, which consists in generating valid inequalities, called cuts, that are violated by the solution of the linear relaxation of the problem. At each iteration, the linear relaxation is solved, and cuts that are violated by the solution are added to the formulation, strengthening the bound given by the relaxed problem. With the use of cutset-based inequalities (to be seen in section 2.4), the authors managed to obtain promising results, what encouraged us to use such inequalities in the present work. Their cutting-plane procedure was afterwards embedded into a matheurisitc described in [45].

Also involving cut generation, Kliewer and Timajev [150] presented a Relax-and-Cut algorithm, which considers Lagrangian relaxation, with cuts being added dynamically, in a Lagrangian relaxed way. Applying the proposed algorithm to solve LPs in a Branch-and-Bound scheme, the authors obtained encouraging results with the use of cover inequalities, and local cuts based on Lagrangian reduced costs.

In fact, cut generation procedures can be embedded into Branch-and-Bound algorithms to improve bounding aspects. Such variation, called Branch-and-Cut, was implemented by, Sellman *et al.* [197] with the addition of local valid cardinality cuts. Still employing such a scheme, Chouman *et al.* [50] applied a filtering process to fix variables while exploring the search tree. Besides the variable fixing feature, using cover inequalities and minimal cardinality inequalities (section 2.4.1), the authors obtained favourable results when compared to Cplex, one of the most popular and efficient mathematical programming solvers in the market.

Similarly, resulting in a Branch-and-Price algorithm, the Branch-and-Bound method may embed column generation procedures to improve computational efforts when solving large scale problems. In fact, column generation is based on the fact that one may not need all the variables of a program to solve it. In that sense, the procedure starts with a restricted program that considers only a subset of variables, and iteratively add to this program only variables that improve the current solution value. Given that improvements are measured by reduced costs, the search for an improving variable (column) corresponds to the pricing problem, which explains the term Branch-and-Price.

Further in that matter, embedding column and cut generation within a Branch-and-Bound scheme, Gendron and Larose [107] proposed a Branch-and-Price-and-Cut algorithm to efficiently solve large-scale FCMC instances, obtaining better performances in comparison to Cplex. In that work, taking the weak arc-based formulation into account (without forcing constraints), the restricted program, solved at each column generation iteration, considered all the design variables, but only a subset of the multicommodity flow variables. In its turn, the cut generation step was used to identify violated forcing inequalities to be added. The same type of procedure was implemented by Barnhart *et al.* [28], for the unsplittable multicommodity flow case.

From another perspective, Costa [57] presented a survey on several works where benders decomposition is applied to approach fixed-charge network design problems (see [190] for a state-of-the-art on the subject). The method, first presented in [31], decomposes the problem into a master program, corresponding to the original one without a group of constraints (usually complicating), and a subproblem, solved to generate optimality and feasibility cuts to the master. In [58], a benders decomposition is applied with the use of metric inequalities. With the aim of improving lower bounds, Sridhar and Park [202] proposed a Benders-and-cut algorithm, adding to the master problem cutset-based inequalities besides Benders cuts. Another alternative to enhance benders decomposition performances was proposed by Costa *et al.* [59], where heuristic solutions were used to generate extra benders cuts per iteration.

Moreover, Hewitt *et al.* [127] combined exact and heuristic approaches to provide upper and lower bounds for the problem, exploiting both arc-based and path-based formulations. Based on local search, at each iteration, a subset of arcs is chosen, and the corresponding restricted integer problem is solved considering the arc-based formulation to obtain a primal solution. On the other side, a cut and column generation procedure was adopted to obtain a lower bound using the path-based formulation. The computational experiments demonstrated that the

proposed approach is a good alternative to quickly produce good quality primal solutions, but the lower bounds obtained were relatively weak.

The same authors present, in [126], a new strategy for solving integer programs, including the FCMC. The approach considers an extended formulation that introduces new columns as restrictions to the model. In addition, binary variables are introduced, modelling the restriction activation in such a way that a solution to the extended formulation leads to a restricted integer program to be solved next. In the FCMC case, a restriction may be a fixed network topology, so one just have to route the multicommodity flow. In that sense, a branch-and-price algorithm is used to solve the extended problem, with restrictions being produced by column generation. When compared to a state-of-the-art MIP solver, the proposed algorithm was able to produce better quality solutions in less time.

More recently, Gendron *et. al* [106] presented a matheuristic based on iterative linear programming, that was capable of finding new best solutions for some hard benchmark instances. The method iteratively solves the linear relaxation of the model, and uses the relaxed solution to select a set of integer variables to be fixed. The corresponding restricted integer model is then solved, and a pseudo-cut is added to the formulation for the next iteration. Such pseudo-cut excludes the solutions that satisfy the fixed values, allowing the algorithm to explore the whole set of solutions.

### 2.3.2 Related problems

We now discuss problems related to the FCMC that may be encountered in the literature. A well-known variant is the *uncapacitated* version, where the values of $w$, for every arc, are greater or equal to the sum of all demands to be routed. Considered as an easier version, notably because of the quasi-integrality of its polytope [124], efficient procedures have been proposed to solve it, for example the dual ascent procedure, by Balakrishnan *et al.* [21]. An attempt to produce a dual ascent procedure for the capacitated case was made by Herrmann *et al.* [125]. However, a note on this work was published by Gendron, showing that the procedure is not effective [100].

Remark that depending on the problem characteristics, it might be preferable to work with a 'normalized' domain of variables. So, instead of having flow variables defined as nominal amounts of flow, one may consider them as percentages of a commodity demand passing through a certain arc or path.

In terms of modeling, the flow variables are bounded in the interval $[0, 1]$, and multiplied by the demand whenever the amount of flow is needed, namely in the cost function and in the capacity constraints (2.9) and (2.3). Moreover, one might

set to 1 the flow bounds $b$ in the forcing constraints (2.4) and (2.10), and the *right-hand-side* of the flow balancing constrainsts (2.2) and (2.7) might be equal to 1, 0 or -1, accordingly.

The use of percentages to represent flow is very common for example in the *unsplittable* variation, in which the flow of a given commodity must follow a single path. In this case, the problem becomes a pure IP, with the flow represented by binary variables. A work on both splittable and unsplittable flow, as a substructure of capacitated network design problems, was published by Atamtürk and Rajan [17].

A different version of the problem also appears when the network is undirected, meaning that, instead of one-way node connections, the links are represented by two-way connections called *edges*. In this case, the installed capacity is shared by the flow in both senses, which means that, for an edge $e = (i, j)$, the capacities constraints in the arc-based formulation might be rewritten as $\sum_{k \in K}(x_{ij}^k + x_{ji}^k) \leq w_e y_e$. Alternatively, edges can be replaced by pairs of arcs in both directions to obtain a directed graph.

Another closely related problem arises when the domain of arc variables $y$, is enlarged to $\mathbb{Z}_+$, so they can assume any nonnegative integer value. In this case, multiple facilities can be installed on the arcs, with different capacities and costs in case more than one type of facility is available. For this variant, generally known as the *network loading problem* [163, 36], several authors have dealt with objective functions based only on fixed costs.

Practical instances of the network loading case were tackled by Barahona [25], using a relaxation approach based on cutset inequalities. With a different perspective, Magnanti *et al.* [162] investigated the polyhedral structure of this problem, presenting a strengthened formulation. Moreover, using binary variables, Frangioni and Gendron [93] formulated network loading as a multiple-choice model with piecewise linear design costs, so on any given arc, each part of its cost function represents the number of facilities to install (similar models were surveyed in [172]). Indeed, Croxton *et al.* [66] reported on three alternative models to this kind of problem, the multiple-choice formulation included, showing that their linear relaxations are equivalent, and that they three approximate the lower convex envelope of the cost function. The same authors, in [67], investigated an extended formulation based on variable disaggregation, obtaining better polyhedral descriptions of the solution space. An alternative, not involving the addition of binary variables is explored by Keha *et al.* in [144, 143]

A number of works have considered commodity-independent transportation costs, that enables the aggregation of commodities sharing the same destination

or the same origin, consequently reducing the problem size. Recently, Chouman *et al.* [49] published a comparison between these two commodity representations: aggregated and disaggregated, finally concluding that the disaggregated formulation presents stronger results for the FCMC. Previous results about commodity aggregation/disaggregation can be found in [37, 191].

Note that an installed capacity may already exist on the network, leading the loading problem to be considered as the *capacity expansion problem* [119]. Minoux considered those types of network loading problems as simplified versions of a larger class of discrete cost multicommodity network optimization problems. In [172], the author discusses a general model that captures aspects like economy of scale, for example. Moreover, existing exact methods to approach those cases are surveyed in the same paper.

Other variants may appear with the addition of side constraints, typically imposing budget and topological restrictions. More precisely, the budget constraints might restrict the total design or routing costs to a limited budget or financial resource, while the topological constraints might impose rules on the network configuration, including relations of precedence and multiple choice [164]. Holmberg and Yuan [131], for example, dealt with side constraints on paths, specifying that commodities must follow paths with a cost less than or equal to a maximum value. Another example can be found in [206], where additional design-balance constraints are imposed.

An interesting textbook about potential function methods applied to minmax or maxmin formulations (minimal congestion or maximum throughput) was given by Bienstock in [35], with theory and practical algorithms. Bektaş *et al.* [29] worked with the possibility of violation of additional constraints at the expense of some penalty costs, resulting in a nonlinear multicommodity network design program. Related to that, another example of nonlinear program was exposed by Paraskevopoulos *et al.*, in [185], where besides transportation and fixed costs, the objective function explicitly adds congestion features, represented by a (nonlinear) delay-cost function. A survey on nonlinear multicommodity flow problems was published by Ouorou *et al.* [181], focused on convex models

A particular case of the capacity expansion problem also consider congestion resulting in nonlinear costs (see [167]). Luna and Mahey [161] provided, along with lower bound results, a modelling framework for this problem, using piecewise-convex arc functions. With such a framework, a heuristic method was proposed by Souza *et al.* in [72], based on local optimality conditions discussed in [166]. In addition, exact approaches were proposed in [79] and [165].

Finally, more specific variations may appear in the literature. In [128], Holm-

berg *et al.* dealt with a problem of transportation planning of empty freight cars in the context of a rail operator, with fixed-charges related to paths. Another particular case arises with capacities and fixed costs concerning nodes instead of links. For those cases, depending on the cost figure, a reduction to an edge capacity problem is possible by splitting nodes into two, forming a link on the network. Belotti *et al.* [30] present a case where such reduction is not possible and propose a method to solve it.

### 2.3.3 Metaheuristics

Given the difficulty of solving large-scale FCMC problems to optimality, much effort has been dedicated to develop efficient heuristic and metaheuristic procedures to find good feasible solutions to realistically sized instances. Crainic *et al.* [63] proposed a *tabu seach* metaheuristic [112], which consists of a local search procedure, and a tabu list serving as memory, to prevent performing repeated movements. The algorithm iteratively tries to improve the current solution by moving to a neighbor one in a defined neighborhood, each visited solution being stored in the tabu list, so the same solution is not visited again for a specific number of iterations. In their work, the authors made use of the path-based formulation and defined a simplex based neighborhood, so the local search is done exchanging path variables in a simplex pivot-like fashion. A second neighborhood is defined relative to the design variables and is used to modify drastically the network configuration and to diversify the search. When compared to the solutions obtained with the software Cplex, the procedure yielded better solutions in significantly less time.

A tabu search was also implemented by Ghamlouche *et al.* [110], however with a neighborhood structure based on low-cost cycles present in the residual graph. The main idea was to reroute the flows around a cycle, eventually modifying the network configuration as a consequence. The results given by this cycle-based tabu search showed a significant improvement over the simplex pivot based one.

In order to explore at best the strength of cycle-based neighborhoods, the same authors embedded such structure in a *path relinking* procedure [111], which provided better performances in comparison with the tabu search. The main idea of such a procedure is to produce a path between initial and guiding solutions present in a reference set. In other words, it produces intermediary solutions, progressively introducing into the current solution, attributes of the guiding one. That may improve the reference set, and eventually the overall best solution.

Both simplex-based and cycle-based neighborhoods were tested with parallel schemes. A multi-thread cooperative parallel procedure was implemented

by Crainic and Gendreau [62], where each process independently executes the simplex-based tabu search, and exchange information with the others, in order to improve the searches and to increase the chances of obtaining high-quality solutions. Indeed, better solutions were found using parallelism. With similar principles, Crainic *et al.* [65] implemented the cycle-based tabu search procedure in a multilevel cooperative algorithm, also improving previous results.

Recently, parallel computational architectures were used by Munguía *et al.* [176], who derived multiple independent neighborhoods that were explored in parallel, by a local search procedure. In the algorithm, once the processes are finished, the improvements identified in parallel are recombined to generate good quality solutions. The authors consider the problem in both splittable and unsplittable cases, claiming that the proposed technique is compatible with both variants.

Similar to the path relinking already mentioned, a *scatter search* procedure was proposed by Alvarez *et al.* [8], for the undirected version of the FCMC. The main idea of the algorithm is to construct a reference set of solutions, that are combined with each other, in order to produce better solutions and then evolve such a reference set (see [113] for more details in path relinking and scatter search). The same authors, in [7], embedded a *greedy randomized adaptive search procedure* (GRASP) [78] into the scatter search to produce a better initial set of solutions. Briefly, at each iteration of GRASP, a greedy solution is constructed, followed by a local search to produce one initial solution. The computational tests showed that the algorithm produces good quality solutions, but tends to be less efficient for instances with a large number of commodities and high fixed costs.

Another heuristic approach, based on *slope scaling*, was proposed by Crainic *et al.* [64] (inspired by the work of Kim and Pardalos [148] for the single-commodity version). The procedure consists in solving the linear multicommodity flow problem associated with the original formulation, with costs iteratively changed in order to reflect the marginal fixed cost and the transportation cost simultaneously. Moreover, intensification and diversification phases were performed using information stored in long-term memory, in order to explore a wider region of the solution space. The results showed that the slope scaling procedure was competitive with other state-of-the-art algorithms.

With a different perspective, Katayama *et al.* [141] presented a *capacity scaling* heuristic. The method iteratively solves the linear relaxation of the FCMC, with arc capacities changed accordingly to the amount of arc flow, present in the relaxed solution of the previous iteration. Using benchmark instances, the computational experiments showed that the capacity scaling algorithm managed to provide one of the best results at that point in the literature. In one of his later work, Katayama

[140] combined the capacity scaling procedure with the *local branching* method, improving the results.

A local branching approach is also proposed by Rodríguez-Martín and Salazar-González [193], providing good results on benchmark instances. The principle behind the method is to explore solution neighborhoods by adding linear inequalities, called local branching constraints, that delimit the neighborhood to be explored, and exclude the current solution. Although the local branching is originally presented as an exact solution strategy, one may interpret it as a heuristic procedure, by setting time and node limits as stopping criteria.

Yaghini *et al.* [209] presented a cutting-plane neighborhood structure that follows the relaxation induced neighborhood search (RINS). As the name suggests, this procedure takes into account the solution given by the current continuous relaxation of the problem, besides the current integer solution. At each iteration, arc variables with the same values in both solutions are fixed, and the sub-MIP on the remaining variables is solved by local branching to obtain a neighbor integer solution. In the work by Yaghini *et al.*, the continuous relaxation problems were strengthened with valid inequalities (discussed in section 2.4), and a tabu search algorithm using the proposed neighborhoods was implemented, providing better results for some benchmark instances when compared to other algorithms in the literature.

Concurrently, Paraskevopoulos *et al.* [184] worked with chains of inefficient arcs to define neighbor solutions, aiming to reroute the flow present in those inefficient arcs. The authors treated the structure as an enhanced cycle-based neighborhood, and implemented an *evolutionary algorithm* consisting of a scatter search phase, and a *iterated local search* phase. The first phase generates descendent solutions of a reference set, and the second tries to increase the quality of the descending solutions, performing a series of local searches and perturbations based on inefficient chains. The computational experiments showed that the proposed algorithm was able to produce high-quality solutions for benchmark instances.

Momeni and Sarmadi [175] implemented a *genetic algorithm* cooperative local branching procedure, presenting better solutions for some benchmark instances. The procedure applied the principles of combining solutions in a reference set, to explore the neighborhood of the current best solution, while local branching was used for diversification, in a RINS fashion.

The *simulated annealing* (see [149]) was also tested for the FCMC. The particularity of this metaheuristic is that, starting from an initial solution, one may accept movements to low-quality neighbor solutions, with a certain probability of acceptance, in order to escape from local minima. In that sense, with the help

of a parameter called *temperature*, the probability of accepting worse solutions is initially high, and then it is decreased throughout iterations. Efficient simulated annealing algorithms were implemented by Yaghini *et al.* in [211], considering the path-based formulation, and by the same first author and others in [210], this time working with simplex-based neighborhoods applied to the arc-based formulation.

## 2.4 Relaxations and valid inequalities

We now discuss cutset-based relaxations, and valid inequalities derived from them, for the FCMC. Here we do not intend to survey the whole literature on cutset polyhedra, but to present what has best worked for the FCMC in particular, based on the work by Chouman *et al.* [46, 47, 48, 49] (we refer to [14] and the references therein for further reading on the subject). In this section, the considerations are related to the arc-based formulation, thus the relaxations refer to the set $P$, the feasible set defined by the constraints (2.2) to (2.6):

$$P = \{(x_a^k, y_a)_{a \in A, k \in K} : (2.2) - (2.6)\}$$

Namely, a cutset is the set of arcs $(S, \bar{S}) = \{(i,j) \in A : i \in S,\ j \in \bar{S}\}$, defined for a given partition of nodes $S \subset N$ and its complement $\bar{S} = N \setminus S$, connecting a node in $S$ to a node in $\bar{S}$. Furthermore, associated with each cutset, there is a set of crossing commodities $K_{S\bar{S}} = \{k \in K : O(k) \in S, D(k) \in \bar{S}\}$, and the reverses: $(\bar{S}, S)$ and $K_{\bar{S}S}$ defined analogously.

With the aim of improving readability, we define the following notations: for any cutset $(S, \bar{S})$ and any subset $L \subseteq K$, the crossing demand is defined as $q^L_{S\bar{S}} = \sum_{k \in K_{S\bar{S}} \cap L} q^k$, and for any arc $a \in A$, $x^L_a = \sum_{k \in L} x^k_a$. Finally, define $b^L_a = min\{w_a, \sum_{k \in L} q^k\}$, $(v)^+ = max\{0, v\}$ and $co(\mathrm{F})$ the convex hull of the solutions in the set F. The equivalent notation is defined also for the reverse cutsets.

Following such notation, given a cutset $(S, \bar{S})$ and commodity set $L \subseteq K$, we define $P_L$ to guide further discussions. Being a cutset-based relaxation itself, $P_L$ is the solution space subjected to (2.14) the sum of flow conservation equalities for all $i \in S$ and $k \in L$, and (2.15) the forcing constraints related to commodities $k \in L$ for all arcs $a \in (S, \bar{S})^+ := (S, \bar{S}) \cup (\bar{S}, S)$, together with integrality of variables $y$. In fact, considering that $x^L_a \leq x^K_a \leq w_a y_a$, one can state that $x^L_a \leq b^L_a y_a$ for all $a \in A$. The next sections present special cases of $P_L$.

$$P_L = \{(x^L_a, y_a)_{a \in (S,\bar{S})^+} : \quad \sum_{a \in (S,\bar{S})} x^L_a - \sum_{a \in (\bar{S},S)} x^L_a = q^L_{S\bar{S}} - q^L_{\bar{S}S}, \tag{2.14}$$

$$0 \leq x^L_a \leq b^L_a y_a,\ y_a \in \{0,1\}, \forall a \in (S,\bar{S})^+ \quad \} \tag{2.15}$$

Remark that constraints (2.14) and (2.15) are redundant for $P$, so adding those constraints to the problem and relaxing constraints (2.2) to (2.6), one obtain the relaxation $P_L$, with a correct change on dimensions.

### 2.4.1 Cutset relaxation

A special case of $P_L$ refers to the fixed charge problems studied by Padberg *et al.* [183]. Indeed, taking $L = K_{S\bar{S}}$, one has $q^L_{\bar{S}S} = 0$ by definition, and (2.14) reduces to $\sum_{a\in(S,\bar{S})} x^L_a = q^L_{S\bar{S}} + \sum_{a\in(\bar{S},S)} x^L_a$. Since flow variables are nonnegative, the resultant equality constraint can be relaxed to inequality (2.16), and $P_L$ can be relaxed to $P_c$.

$$P_c = \{(x^L_a, y_a)_{a\in(S,\bar{S})} : \sum_{a\in(S,\bar{S})} x^L_a \geq q^L_{S\bar{S}} \qquad (2.16)$$
$$0 \leq x^L_a \leq b^L_a y_a,\ y_a \in \{0,1\}, \forall a \in (S,\bar{S}) \quad \}$$

Padberg *et al.* provide valid cover inequalities for $co(P_c)$, considering also the cases with (2.16) as an equality ($P^=_c$), and as a ($\leq$)-inequality ($P^\leq_c$). Moreover, a discussion about the properties relating those polytopes is made, resulting in observations like Proposition 1.

The authors let a set $C \subset (S,\bar{S})$ to be a cover if $\lambda = \sum_{a\in C} b^L_a - q^L_{S\bar{S}} > 0$, and define the valid inequality (2.17) for $C^+ = \{a \in C : (b^L_a - \lambda) > 0\}$, $D \subseteq (S,\bar{S}) \setminus C$ and $\bar{b} = max\{b^L_a : a \in C\}$, which is facet-defining for $co(P^=_c)$ if $0 < \bar{b} - \lambda < b^L_a \leq \bar{b}$ for all $a \in D$.

$$\sum_{a\in C\cup D} x^L_a + \sum_{a\in C^+} (b^L_a - \lambda)(1 - y_a) \leq q^L_{S\bar{S}} + \sum_{a\in D} (\bar{b} - \lambda) y_a \qquad (2.17)$$

For $L = \emptyset$, (2.17) gives origin to the famous *flow cover inequalities* which are also facet defining for $co(P^=_c)$ under certain conditions. Very recently, new inequalities were introduced in [157, 158], that are valid for $P^\leq_c$. Moreover, according to Proposition 1, inequality (2.18) is automaticaly facet-defining for $co(P_c)$, under the same conditions. Hence, it is also a valid inequality for $P$.

$$\sum_{a\in(S,\bar{S})\setminus\{C\cup D\}} x^L_a + \sum_{a\in C^+} (b^L_a - \lambda) y_a + \sum_{a\in D} (\bar{b} - \lambda) y_a \geq \sum_{a\in C^+} (b^L_a - \lambda) \qquad (2.18)$$

**Proposition 1.** [183] *If $\pi x + \mu y \geq v$ is a nontrivial facet of co($P_c^{=}$), then $(\pi - t\mathbf{1})x + \mu y \geq v - tq^L_{S\bar{S}}$ is a nontrivial facet of co($P_c$), where $t = min\{\pi_a : a \in (S, \bar{S})\}$*

From another perspective, one can define a pack $R$ to be such that $\mu = q^L_{S\bar{S}} - \sum_{a \in R} b^L_a > 0$, with $C = (S, \bar{S}) \setminus R$ being its complement. Hence, the inequality (2.19) is valid for $co(P_c)$, since at least one arc in $C$ must be open to meet the demand.

$$\sum_{a \in C} y_a \geq 1 \tag{2.19}$$

Alternatively, considering that $b^L_a y_a \geq x^L_a$ for all arcs in the cutset, one can define the minimum knapsack cutset relaxation $P_{kc}$. With respect to inequality (2.19), If $b^L_a \geq \mu$ for all $a \in C$, the cover is *minimal* and a lifting procedure (discussed in section 4.4.3) can be used to derive facets of $co(P_{kc})$[22]. See [15] for a survey on cover and pack inequalities for other types of knapsack polytopes.

$$P_{kc} = \{(y_a)_{a \in (S,\bar{S})} : \sum_{a \in (S,\bar{S})} b^L_a y^L_a \geq q^L_{S\bar{S}} \quad y_a \in \{0,1\}, \forall a \in (S, \bar{S}) \quad \} \tag{2.20}$$

Apart of *minimal cover inequalities*, Chouman *et al.* [49] and Martello and Toth [169] have worked with *minimum cardinality inequalities* (2.21) (valid for $co(P_{kc})$). The first worked with the FCMC problem, and the second with the 0-1 Knapsack Problem. On the right-hand-side of (2.21), the value $m_{(S,\bar{S})}$ gives the minimum number of arcs that must be opened to satisfy the demand (the computation of $m_{(S,\bar{S})}$ is discussed in section 4.4.2).

$$\sum_{a \in (S,\bar{S})} y_a \geq m_{(S,\bar{S})} \tag{2.21}$$

As discussed in section 4.4, one can strengthen cutset relaxations using a metric to redefine the amount of demand that must cross the cutset, obtaining stronger inequalities for $P$. In the network loading problem case, Avella *et al.* [18] defined metric inequalities that completely define the convex hull of feasible solutions.

Finally, it is worth noting the possibility of working with *k-partitions*, which means to divide the set $N$ into $k$ disjoint sets of nodes, instead of only two as it is considered here. Indeed, several works have used such an approach for cases where an undirected graph is considered, for example [4, 3, 25], deriving similar valid inequalities for that context. The next section presents the cutset flow relaxation and valid inequalities derived from it.

### 2.4.2 Cutset flow relaxation

The cutset flow relaxation for the FCMC is known as the single node flow set, for which several polyhedral studies have been published, including the one written by Van Roy and Wolsey [205] presented as an extension to [183]. Considering any set $L \subseteq K$, and $d^L_{S\bar{S}} = q^L_{S\bar{S}} - q^L_{\bar{S}S}$, such relaxation consists in dropping equality in (2.14) to define $P^f_L$.

$$P^f_L = \{(x^L_a, y_a)_{a\in(S,\bar{S})^+} : \sum_{a\in(S,\bar{S})} x^L_a - \sum_{a\in(\bar{S},S)} x^L_a \leq d^L_{S\bar{S}}, \quad 0 \leq x^L_a \leq b^L_a y_a,\ y_a \in \{0,1\}, \forall a \in (S,\bar{S})^+ \ \} \tag{2.22}$$

In their work, Van Roy and Wolsey presented a generalization of inequalities (2.17), defining a flow cover as a pair of sets $(C_1, C_2)$, such that $\lambda = \sum_{a\in C_1} b^L_a - \sum_{a\in C_2} b^L_a - d^L_{S\bar{S}} > 0$, $C_1 \subseteq (S,\bar{S})$ and $C_2 \subseteq (\bar{S},S)$. For $D_2 \subseteq (\bar{S},S) \setminus C_2$ and $R_2 = (\bar{S},S) \setminus \{C_2 \cup D_2\}$, the (generalized) flow cover inequality is defined by (2.23).

$$\sum_{a\in C_1}(x^L_a + (b^L_a - \lambda)^+(1-y_a)) - \sum_{a\in D_2} min\{b^L_a, \lambda\}y_a \ - \sum_{a\in R_2} x^L_a \leq \sum_{a\in C_2} b^L_a + d^L_{S\bar{S}} \tag{2.23}$$

According to Theorem 6 of [205], if $d^L_{(S,\bar{S})} > 0$, $max_{a\in C_1} b^L_a > \lambda$ and $C_2 = \emptyset$, (2.23) is facet-defining for $co(P^f_L)$. Moreover, Gu *et al.* [116] propose a sequence independent lifting procedure to obtain higher-dimensional inequalities based on superadditive functions (a similar lifting procedure is proposed for inequalities (2.17) too). Recently, three-partition flow cover inequalities were proposed in [16].

Complementarily, Stallaert [203] presents valid inequalities for the case that $\lambda < 0$. Atamtürk [13] later present a special case of those inequalities denoted

*flow pack inequalities*, along with a lifting procedure also sequence independent and using superadditive functions. According to Stallaert, those inequalities must be viewed as cover flow inequalities for a relaxation of $P_L^f$, when a slack variable $s$ is added to (2.22) ($\sum_{a\in(\bar{S},S)} x_a^L - \sum_{a\in(S,\bar{S})} x_a^L - s \leq -d_{S\bar{S}}^L$).

A flow pack is defined as a pair $(C_1, C_2)$, such that $\mu = \sum_{a\in C_2} b_a^L - \sum_{a\in C_1} b_a^L + d_{S\bar{S}}^L > 0$, $C_1 \subseteq (S,\bar{S})$ and $C_2 \subseteq (\bar{S},S)$. For $D_1 \subseteq (S,\bar{S}) \setminus C_1$ and $R_2 = (\bar{S},S) \setminus C_2$, the flow pack inequality is defined by (2.24). According to Atamtürk [13], under certain conditions, flow pack inequalities are facet-defining for the convex hull of the restriction of $P_L^f$ when $y_a = 1 \; \forall \; a \in C_1$.

$$\sum_{a\in C_1} x_a^L + \sum_{a\in D_1} (x_a^L - min\{b_a^L, \mu\} y_a) + \sum_{a\in C_2} (b_a^L - \mu)^+ (1 - y_a) - \sum_{a\in R_2} x_a^L \leq \sum_{a\in C_1} b_a^L \tag{2.24}$$

Finally, cutset inequalities were presented in [46, 49], which served in fact as a first step for the separation of flow cover and flow pack inequalities. Other cutset inequalities have been proposed in [14, 189]. Most recently, in [157], based on the results given by Proposition 1, the authors discussed the possibility of rotating inequalities identified for a knapsack polytope obtained from a restriction of $P_L^f$, to devise valid inequalities for $P_L^f$.

## 2.5 Conclusion

As discussed in the previous sections, much work has been done for the development of the best approaches to solve the FCMC and related problems, but some instances, especially the large ones, remain unsolved by the current methods. As shown in section 2.2, high fixed-charge values when compared to transportation costs, generated in a tight interval may be a complicating feature, as well as sufficiently tight arc capacities. In that sense, efficient methods able to provide tight gaps for hard large-scale FCMC instances are still needed to be devised.

For this purpose, Lagrangian schemes combined with the generation of valid inequalities like the ones presented in section 2.4 seem to be a good alternative. Specifically in chapter 4, following previous works in the literature, the inequalities most successfully implemented for the FCMC, among the ones presented in the last section, are considered for the development of a Relax-and-Cut algorithm.

# Chapter 3

# Lagrangian Bounds and Volume-Bundle Comparison

A part of this chapter content has been published in the official journal of the International Federation of Operational Research Societies (IFORS): *International Transactions in Operational Research*. Here, we compare the performances of the Bundle method and the Volume algorithm (two of the most efficient techniques to obtain accurate Lagrangian dual bounds for hard combinatorial optimization problems). The comparison is made on very large-scale Fixed-Charge Multicommodity Capacitated Network Design problems. The motivation is not only the quality of the approximation of these bounds as a function of the computational time, but also the ability to produce feasible primal solutions and thus to reduce the gap for very large instances for which optimal solutions are out of reach. Feasible solutions are obtained through the use of Lagrangian information in constructive and improving heuristic schemes. We show in particular that, if the Bundle implementation has provided great quality bounds in fewer iterations, the Volume algorithm is able to reduce the gaps of the largest instances, taking profit from the low computational cost per iteration compared to the Bundle method.

## 3.1 introduction

Lagrangian Relaxation has been widely used for long to generate lower bounds for difficult constrained minimization problems and to serve as a basis for developing efficient approximation schemes, see [109, 156, 90] for the basic theory. As the resulting Lagrangian dual functions are generally nonsmooth and concave, the ability to lean on efficient subgradient algorithms is a crucial issue for the success of Lagrangian Relaxation.

The reason for using Lagrangian Relaxation to obtain lower bounds for the

FCMC problem is justified when large-scale instances are involved. These instances will generally be out of reach for the general-purpose MIP solvers like Cplex, even if they are indeed able to exploit the block structure of the underlying Linear Programs (LP) at each node of their branching search tree.

Resuming the main features of Lagrangian Relaxation, we start from a primal problem (P) in $\mathbb{R}^n$, supposed to be linear with mixed-integer variables, defined as:

$$\text{Minimize} \quad c.x \quad \text{s.t.} \quad Ax = b, x \in S$$

where $Ax = b$ represent the difficult constraints we want to relax ($A$ is a $(p \times n)$ matrix). The set $S$ may be discrete and defined by linear constraints. The continuous (or linear) relaxation lower bound is defined as $Z_L = \min_x c.x \quad \text{s.t. } Ax = b,\ x \in \bar{S}$, where $\bar{S}$ is the continous set defined by constraints of $S$ with integrality constraints dropped. Moreover, considering the convex hull $co(S)$ of the set $S$, if $co(S) = \bar{S}$ then the *integrality property* holds [109].

For a given vector of Lagrange multipliers $u \in \mathbb{R}^p$ associated with the difficult constraints, the Lagrangian subproblem defines a lower bound for the optimal value of the primal problem:

$$L(u) = \inf_{x \in S} (c - A^T u).x + b.u$$

The dual problem is thus to search the best lower bound, i.e. to maximize the dual function $L$ on $\mathbb{R}^p$ which is indeed concave on any convex subset of its domain (see [155] for example). That function is generally nonsmooth and piecewise affine (with a huge number of pieces, theoretically up to the number of extreme points of the polyhedral set $co(S)$). This motivates the search for efficient algorithms of nonsmooth optimization. These take profit of the fact that, for any solution $x(u)$ of the Lagrangian subproblem, a subgradient of $L$ at $u$ is easily computed, indeed $g(u) = b - Ax(u) \in \partial L(u)$, where $\partial L(u)$ denotes the set of subgradients of $L$ at $u$. Finally, we recall that the best lower bound $Z_R = \sup_u L(u)$ is finite if the primal problem is feasible and satisfies $Z_L \leq Z_R \leq Z^*$, where $Z^*$ is the optimal value of the primal problem. Moreover, $Z_L = Z_R$ if the Lagrangian subproblem has the integrality property.

Subgradient algorithms have been studied early by Shor and Polyak in the sixties [188] and first applied to the Lagrangian Relaxation of hard combinatorial problems by [122] in their seminal paper about the Traveling Salesman problem. Further improvements have been proposed later, either by the Russian school [147, 200] or by western researchers [123]. These variants try to improve the search direction like in the conjugate gradient method or change the metric of the direction-finding step (Khachian's ellipsoid algorithm is indeed a subgradient

method).

In this chapter, we compare two classical versions of these algorithms, namely the Bundle method, early proposed by Wolfe and Lemaréchal [155], and the Volume algorithm proposed by Barahona and Anbil [26]. As we will see below, Bundle methods and the Volume algorithm share the same strategy as early methods, extending the direction-finding step to more than two former subgradients. Such a comparison effort is inspired by the work of Lemarechal [156] and Frangioni *et al* [92] who contributed to the present success of modern Bundle methods, which remain the most successful approach to treat small or medium scale combinatorial problems by Lagrangian relaxation. Our claim is that the Volume algorithm can do as well or even better when dealing with large-scale network design problems. Other methods like analytic center [114] or proximal cutting planes, deflected [198] or majorize-minimize mean [133] subgradient algorithms, exist in the literature. However, a state-of-the-art review about nondifferentiable optimization (NDO) methods is out of the scope of the present work, and we refer to [40] for a broad history and practical guide on the subject.

Papers comparing nonsmooth optimization algorithms can be found in the literature [42, 95], but a direct comparison of these two algorithms applied to large-scale combinatorial models is missing, and our work is an attempt to fill this gap. An interesting paper was written by Briant *et al.*[42], where the authors compare different algorithms including Bundle, Column Generation, and the Volume, for five different problems. With respect to the Volume-Bundle comparison, the results have shown that they behaved similarly, but Bundle enjoyed more reliable stopping criteria, even though it might be fairly expensive to reach them. According to that paper, the Bundle method obtained better bounds with fewer iterations, though we believe that its average time per iteration is fairly more expensive than the Volume one. Considering that, the present work focus the comparison on the computational total time, rather than on the number of iterations.

The next sections will present an explanation about the considered algorithms. Then, in Section 3.5 constructive and improving heuristic schemes using Lagrangian information is proposed to provide upper bounds for the FCMC, as well as to serve as a basis to compare the primal fractional solutions obtained with the two algorithms being tested. The computational experiments and results are reported in Section 3.6. Finally, conclusions are made in section 3.7.

## 3.2 Volume

Observing that the dual function, as a piecewise affine concave function, can be written as the infimum of a finite set of affine functions associated with the potential solutions of the Lagrangian subproblems $x^j, j \in J$, i.e. the extreme points of $co(S)$, the dual problem can be written as :

$$\begin{array}{lll} \text{Maximize} & Z & \\ \text{s.t.:} & Z \leq c.x^j + u.(b - Ax^j) & \forall j \in J \\ & u \in \mathbb{R}^p, Z \in \mathbb{R} & \end{array} \tag{3.1}$$

a linear program with a generally exponential number of constraints which, dualized in its turn, yields the following so-called master program (of the Dantzig-Wolfe decomposition, see [155]) :

$$\begin{array}{ll} \text{Minimize} & \sum_{j \in J} (c.x^j)\lambda_j \\ \text{s.t.:} & \sum_{j \in J} (Ax^j - b)\lambda_j = 0 \\ & \sum_{j \in J} \lambda_j = 1 \\ & \lambda_j \geq 0, \forall j \in J \end{array} \tag{3.2}$$

Cutting planes algorithms use a subset $J_t$ of $J$ at each outer iteration $t$ corresponding to part of the extreme points already generated by the Lagrangian subproblems, thus yielding a restricted master problem where the primal variables $\lambda_j$ are the weights of the extreme points $x^j, j \in J_t$ in a primal solution $\bar{x} = \sum_{j \in J_t} \lambda_j x^j$, feasible for the polyhedral constraint set $\{Ax = b, x \in co(S)\}$.

The Volume algorithm attempts to find an approximate solution of that master problem by computing at each iteration $t$ a *stability center* $\bar{u}$, a step $s_v^t$ and a subgradient-based direction $d_v^t$. The stability center represents a point that has provided significant improvement with respect to the optimization process. In its turn, the step represents how far one may move in the direction of $d_v^t = (b - A\bar{x})$, so that a new trial point $u^t = \bar{u} + s_v^t \cdot d_v^t$ is obtained. The stability center $\bar{u}$ will be updated whenever $L(u^t) - L(\bar{u}) > 0$ and $d_v^t \cdot (Ax^t - b) > 0$ (we get then a 'green iteration' or 'serious step').

The directions are updated at each iteration according to the primal vector estimate $\bar{x}$ such that :

$$\bar{x} \leftarrow \theta x^t + (1-\theta)\bar{x}$$

The parameter $\theta \in [0\ 1]$ is itself updated in order to force an ascent direction, i.e. such that

$$g^t \cdot (\theta g^t + (1-\theta)d_v^t) \geq 0$$

As stated in [19, 26], at the end of an iteration $t$, the coefficients $\{\theta,\ (1-\theta)\theta,\ (1-\theta)^2\theta,\ ...\ ,\ (1-\theta)^t\theta\}$ can serve as an approximation for the primal variables $\lambda_1$, ... , $\lambda_t$ of the Dantzig-Wolfe's master problem, with respect to the dual constraints. So $\bar{x}$ can be interpreted as an approximate primal solution. Furthermore, those $\lambda$ could be approximated by the volume between the active faces of (3.1) and the current lower bound $\bar{Z}$, which explains the name of the method. The choice of the key parameters is detailed in Section 3.6.2.

## 3.3 Bundle

Bundles were initially presented as extensions of the method of $\epsilon$-subgradients [155, 207], nevertheless recent versions include different backgrounds. It is usual to say that bundle methods are stabilized versions of the cutting plane algorithm [145], since they have the same idea of computing models to approximate functions. Bundle methods usually present better performance than Kelley's algorithm since they avoid going too far from the current point, thanks to the stabilization term added to the objective function. [179] presented a brief survey about it. [89] introduced a generalized Bundle method and a version for cases in which the Lagrangian dual can be decomposed.

The main idea is to gather information throughout iterations in order to build a model to approximate the dual function $L(u)$. Indeed, if $g$ is a subgradient of the concave function $L$ at $\bar{u}$, then $L(u) \leq L(\bar{u}) + g.(u-\bar{u})\ \forall\ u \in \mathbb{R}^p$ (extending the dual value with $-\infty$ if the Lagrangian subproblem is unbounded). Assuming that there exists an initial *bundle* $\beta = \{i \mid g_i \in \partial L(u_i)\}$, $\hat{L}(u)$ is the piecewise affine concave function such that:

$$L(u) \leq \hat{L}(u) := min\{L(u_i) + g_i.(u-u_i) : i \in \beta\} \quad \forall u \in \mathbb{R}^p \tag{3.3}$$

The model at this point is represented by a group of affine functions that together form an easier nondifferentiable optimization problem. The Moreau-Yosida regularization comes then as an alternative to this problem since the function and its regularized function share the same maximum. The regularized concave function is defined by:

$$L_s(\bar{u}) = max_u \quad \hat{L}(u) - \frac{1}{2s_b}||u - \bar{u}||^2 \tag{3.4}$$

where $s_b$ is the step size in the Bundle method.

Assuming the bundle has $m$ parts, thanks to the information transfer property [155], it is convenient to rewrite the bundle in terms of linearization errors computed at $\bar{u}$, such as $e_i := L(u_i) - L(\bar{u}) + g_i.(\bar{u} - u_i) \quad \forall i = 1, ..., m$. Then rewriting the calculation of $L_s(\bar{u})$ as a constrained program with an auxiliary scalar variable $\eta$ :

$$\begin{aligned} \text{Maximize } \eta \ & -\frac{1}{2s_b}||u - \bar{u}||^2 \\ \eta & \leq L(\bar{u}) + g_i(u - \bar{u}) + e_i \quad \forall i \in \beta \\ & u \in \mathbb{R}^p, \eta \in \mathbb{R} \end{aligned} \tag{3.5}$$

Further dualizing (3.5) with the dual coefficients $\alpha_i \geq 0, i \in \beta$, one obtains:

$$(1 - \sum_{i=1}^{\beta} \alpha_i)\eta - \frac{1}{2s_b}||u - \bar{u}||^2 + \sum_{i=1}^{\beta} \alpha_i[L(\bar{u}) + e_i + g_i(u - \bar{u})]$$

Assuming that $(1 - \sum_{i=1}^{\beta} \alpha_i) = 0$, for any given vector $\alpha$, the optimal solution must present $(u - \bar{u}) = s_b \sum_{i=1}^{\beta} \alpha_i g_i$. Hence, the dual of (3.5) may be formulated as:

$$\begin{aligned} \text{Minimize} \quad & \tfrac{s_b}{2}||\sum_{i\in\beta} \alpha_i g_i||^2 + \sum_{i\in\beta} \alpha_i e_i + L(\bar{u}) \\ & \sum_{i\in\beta} \alpha_i = 1 \\ & \alpha_i \geq 0 \ \forall i \in \beta \end{aligned} \tag{3.6}$$

The main search procedure is to get, at each iteration $t$, the solution $\alpha^t$ of (3.6) and a new trial point along the direction of $d_b^t = \sum_{i\in\beta} \alpha_i^t g_i$ with a step of size $s_b^t$. Furthermore, the strong duality property of the pair of quadratic programs allows to estimate the increase in the current solution value if such step is performed, i.e. denoting $u^t$ the current solution of (3.5) :

$$\Delta_\beta = \hat{L}(u^t) - L(\bar{u}) = s_b^t || \sum_{i \in \beta} \alpha_i^t g_i ||^2 + \sum_{i \in \beta} \alpha_i^t e_i \tag{3.7}$$

A weaker version can be obtained considering $L_s(\bar{u})$ the solution value of (3.6) and taking $\delta_\beta = L_s(\bar{u}) - L(\bar{u})$ as an estimation (see Section 3.6.2 for more implementation issues about the role of these estimates).

Bundle methods are now known to be very efficient while solving the Lagrangian dual problem, however with the drawback of the need to solve a quadratic subproblem at each iteration, which can significantly decrease the performance of the method. [87] introduced a specially tailored algorithm to solve such quadratic programs (3.6) in a way to reduce the computational cost and we have used his software in our experiments.

So we conclude that short presentation by observing that both algorithms rely on similar features, namely a combination of subgradients from which we should build at the same time a stability center for the dual problem and a tentative primal solution converging to a feasible (but generally fractional) solution. In addition, note that an approximation for the primal solution can be obtained by setting $\bar{x} = \sum_{i \in \beta} \alpha_i x_i$. Convergence issues will not be discussed here (see [89] for instance for the bundle method). We just mention that the Volume algorithm can be slightly modified to be interpreted as a bundle method and thus inherit its convergence properties [19].

## 3.4 Lagrangian relaxations for the multicommodity network design

Different choices to relax subsets of constraints of FCMC have been considered in the literature. The most common are

- Relaxing the capacity constraints (including the strong inequalities) (2.3)-(2.4) inducing a decomposition by commodity of the pricing step into shortest-path subproblems.
- Relaxing the flow conservation constraints (2.2) inducing a decomposition by arcs into knapsack subproblems.

Most recently Kazemzadeh *et al.* [142] presented node-based Lagrangian relaxations for the FCMC problem, where the Lagrangian subproblem decomposes by nodes. By reformulating the problem, the authors managed to devise three

node-based relaxations which, contrarily to the common ones, do not present the integrality property and may improve over the linear relaxation bound. The results were promising, but the relaxations struggled with time consumption, since the resulting subproblems are hard-to-solve MIPs like the Facility Location. That being said, such node-based relaxations were not tested in the present work, given the size of instances considered (see 3.6.1).

Direct comparisons on the two first alternatives have been made in different works [105, 61, 101] as well as combinations of both, in the so-called 'total' relaxation, which has been used successfully for the 'proximal decomposition' of convex-cost multicommodity flow problems [181] and also considered by [107] for presentation purposes in the present case of FCMC.

Even if it depends on the type of instances, the relaxation of flow conservation constraints has shown a better behaviour, and these comparisons are generally combined with discussions about the possibility to aggregate commodities by origins (or by destinations), or to force other valid inequalities in the model. We will focus on the arc-based model FCMC of Chapter 2, thus only including the strong forcing inequalities (2.4).

We denote the dual multipliers by

- $\alpha_e \geq 0, e \in A$ for the capacity constraints (2.3),
- $\gamma_e^k \geq 0, e \in A, k \in K$ for the strong forcing inequalities (2.4),
- $\pi_i^k \in \mathbb{R}, i \in N, k \in K$ for the flow conservation equations (2.2).

and we need to compare the trade-off between the number of dual variables and the potential splitting into smaller subproblems.

### 3.4.1 Solving the pricing subproblem

As it is generally considered in the literature to be the most effective strategy, we can choose the relaxation of the flow conservation constraints for illustration purposes. In section 3.6.3, previous results were resumed to reinforce our choice.

The Lagrange multipliers associated with each node $i$ and each commodity $k$ are denoted by $\pi_i^k \in \mathbb{R}, \forall\ i \in N, k \in K$. The dual space is thus in $\mathbb{R}^{|N|\times|K|}$ which can be huge for a full set of commodities (of order $|N|^2$). The complete Lagrangian dual function is given by :

$$L(\pi) = \min_y \sum_{e\in A}[f_e + g_e(\pi)]y_e + \sum_{k\in K} q^k(\pi^k_{D(k)} - \pi^k_{O(k)}) \tag{3.8}$$

$$y_e \in \{0,1\},\ \forall e \in A \tag{3.9}$$

where, for each arc $e = (ij) \in A$ :

$$g_e(\pi) = \min_x \sum_{k \in K} (c_{ij}^k + \pi_i^k - \pi_j^k) x_{ij}^k \tag{3.10}$$

$$\sum_{k \in K} x_{ij}^k \leq w_{ij} \tag{3.11}$$

$$0 \leq x_{ij}^k \leq b_{ij}^k, \forall k \in K \tag{3.12}$$

Thus, the computation of $g_e(\pi)$ is given by a continuous knapsack problem defined separately for each arc $e \in A$, and very simple to be solved. It suffices to fill up the arc with the commodities having the most negative reduced costs $(c_e^k + \pi_i^k - \pi_j^k) \leq 0$ if any, until the arc flow equals the capacity. The Lagrangian dual problem is then defined as the maximization of the Lagrangian function $L(\pi)$ on $\pi \in \mathbb{R}^{N \times K}$ as discussed in the previous sections.

## 3.5 Upper bounds

We propose a heuristic scheme to obtain upper bounds for the FCMC problem. The main idea, employed repeatedly, is to select arcs to compose a topology and then optimize the routing of multicommodity flows over the topology set down. The heuristic makes use of (i) perturbations on a given topology to build a pool of topologies, and (ii) arc combinations between pairs of topologies from the pool. A key element of the heuristic is the use of information collected throughout the Lagrangian optimization process to guide topology construction, in particular the frequency in which an arc is opened when solving the subproblem (3.8) - (3.12) at each Lagrangian iteration.

### 3.5.1 Initial solution

We now describe the construction of a first feasible topology, which is done after the procedure to obtain the Lagrangian lower bound. Let $y^{cf} \in \mathbb{R}^{|A|}$, with values $0 \leq y_e^{cf} \leq 1, \forall e \in A$, be the frequency vector of opening arcs, i.e., the ratio between the number of Lagrangian iterations where arc $e$ was opened, and the total number of Lagrangian iterations.

Algorithm 1 describes the procedure to get an initial solution. After solving the Lagrangian dual, we use the frequency value trying to identify attractive arcs to compose a topology. Thus, all arcs with frequency $y_e^{cf} \geq 0.3$ are fixed to 1, forming the topology $A_0$, and all arcs with $y_e^{cf} \leq 0.001$ are discarded. The set $\Phi$ contains the unfixed arcs that will be later evaluated according to their frequency in further solutions of restricted Lagrangian subproblems. Preliminary

computational experiments have shown that topologies composed strictly with arcs such that $y_e^{cf} \geq 0.3$ often do not support feasible routing of the multicommodity flows. So, in the while-loop we try to identify attractive arcs that remained in set $\Phi$. We successively solve Lagrangian subproblems (3.8) - (3.12) with $y_e$ fixed to 1 if $e \in A_0$, $y_e$ unfixed if $e \in \Phi$, and $y_e$ fixed to 0 if $e \notin A_0 \cup \Phi$. Since restricted subproblems have fewer arcs and some of them are already set to 1, we expect that some unfixed arcs may no longer be necessary, while others may become more requested. We get a new vector of frequencies $y^f$ for arcs in $\Phi$ solving the FCMC Lagrangian dual with respect to $\Phi \cup A_0$. The new frequencies serve as basis to fix arcs in $\Phi$, such that the arc is opened if $y_e^f \geq 0.3$, or closed if $y_e^f < b$, where $b$ is a threshold value. We increase the value of $b$ every time there is no change in $\Phi$.

**Algorithm 1** First Feasible Topology

Given $y^{cf}$, the vector of frequency
$A_0 = \{e \in A : y_e^{cf} \geq 0.3\}$
$\Phi = \{e \in A : 0.001 \leq y_e^{cf} < 0.3\}$
$b = 0.01$
**while** $b \leq 0.05$ **do**
    Set $y_e = 1$, $\forall e \in A_0$
    Solve Lagrangian Dual for set $\Phi \cup A_0$ (i.e., (3.1) if using Volume, or (3.5) if using Bundle)
    Given $y^f$, the vector of frequency
    $A_0 = A_0 \cup \{e \in \Phi : y_e^f \geq 0.3\}$
    $\Phi = \Phi \setminus \{e : y_e^f < b \text{ or } y_e^f \geq 0.3\}$
    **if** No changes in $\Phi$ **then**
        $b = b + 0.01$
Check/Restore feasibility of $A_0$
Solve the multicommodity flow problem over topology $A_0$
**return** $A_0$

Given the topology $A_0 \subseteq A$ obtained at the end of the while-loop, we try to satisfy all the multicommodity demands subject to the installed arc capacities. If such demand can be satisfied, the topology is feasible, otherwise either there is no path for a flow to go from its origin to its destination, or there are paths for all $k \in K$ but the capacities installed are not sufficient. The checking procedure adds artificial flow variables $x_{od}^k \geq 0$ with high transportation costs for each commodity $k \in K$, such that $o = O(k)$ and $d = D(k)$. The simplex algorithm is applied until the first basic feasible solution is obtained (Phase I of the method). If all artificial variables have value zero, the topology is feasible and the optimization phase takes

place to solve the multicommodity flow problem using column generation (similar to what is described in [107]). Otherwise, a restoring procedure takes place.

To restore feasibility, the procedure consists in rerouting the flow present in each artificial variable through the shortest path between the origin and the destination of the corresponding commodity, using Dijkstra's algorithm. To ensure that, for a given $k$, the computed path can accommodate $q = x_{od}^k > 0$ units of flow, only arcs with enough residual capacity are considered, i.e., the original arc capacity minus the total flow in the arc must be greater than or equal to $q$. The costs are recomputed accordingly to the amount of flow to be routed, so for a given $k$, if $e \in A_0$ then $c_e^{k'} = c_e^k q$, otherwise $c_e^{k'} = c_e^k q + f_e(1 - y_e^{cf})$ $(e \in A \setminus A_0)$. To summarize, for each commodity with a positive artificial slack, the capacities and costs are recomputed and an O-D shortest path is obtained. The flow is then rerouted and the same process is repeated until all artificial variables have zero value. Finally, all arcs not in $A_0$, but carrying some flow, are added to the topology and the optimal flow is computed using column generation. The commodities are examined in an increasing order of $q$. Once the solution is obtained, opened arcs $e \in A_0$ with no flow are deleted from the topology and its fixed cost subtracted from the solution value.

Although in our computational experiments we have always found a feasible solution with this procedure of successive shortest-paths, it is important to note that there is no guarantee for a feasible solution to be found. Since the shortest-paths are computed successively, it may occur that the only possible path for a certain commodity has been already blocked by previously routed ones. If the procedure to restore feasibility fails, an alternative would be to set $A_0 = A$ which is likely to produce a high-cost solution. As mentioned, we did not face that situation in our computational experiments.

The cost of a feasible topology $A_0$ returned by Algorithm 1 is given by the sum of the fixed charges of the arcs in $A_0$, plus the transportation costs of routing the multicommodity flow through that topology.

### 3.5.2 Searching for a high quality solution

The search for a high-quality solution is performed in two phases. In the first phase, we apply perturbations to the topology leading to the best solution found so far to build a pool of topologies. In a second phase, we generate combinations of arcs from pairs of topologies belonging to the pool in an attempt to obtain improved solutions. The heuristic performs 10 rounds of such scheme, and returns the best solution found.

Algorithm 2 presents the heuristic. The topology leading to the best solution

found so far is kept as $A_{best}$, which is initially set to the first feasible solution obtained with Algorithm 1 described in the previous section. Let $c^*(A')$ be the optimal cost of a multicommodity flow problem over a topology $A' \subseteq A$ (optimal routing cost), and $UB$ is the best upper bound known so far.

The perturbation phase builds a new pool in each round by applying perturbations to $A_{best}$. A perturbation consists in replacing a percentage of the arcs in $A_{best}$. The procedure of perturbation will be described later. For each value $p = 2\%, 4\%, 6\%, 8\%, 10\%$ of $|A_{best}|$ we generate $n = 1, \ldots, 5$ topologies $A_n$. Let $L(A_n)$ be the value computed with Lagrangian relaxation for the optimal routing cost over topology $A_n$, $n = 1, \ldots, 5$. Lagrangian relaxation is used to compute the routing cost in order to speed up the algorithm. The corresponding Lagrangian dual is obtained with the relaxation of the flow conservation constraints, and the subproblem is defined by (3.8) - (3.12) with variables $y_e$ fix to 1, if $e \in A_n$, 0 otherwise. For each value of $p$, we add to the pool the topology of minimum cost among the 5 generated. Thus, at the end of the perturbation phase, the pool has 5 topologies, each one generated with a different value of $p$.

The combination phase generates topologies considering pairs of topologies from the pool. For each unordered pair $\{A', A''\}$ of the pool we generate $n = 1, 2, 3$ topologies $A_n$, such that $A_n \subseteq A' \cup A''$ and $A_n \supseteq A' \cap A''$. The frequency vector $y^{cf}$ defined above is used as the probability for an arc $e \in A' \triangle A''$ to be inserted in $A_n$, so arcs with a high frequency have more chances to be selected. Again, Lagrangian relaxation is used to compute the routing cost over $A_n$, and $A_{round}$ is updated if it is the case.

At the end of each round we have a new topology $A_{round}$. If its cost improves $UB$, we apply the procedure to check and restore feasibility if needed as described in the previous section. The actual routing cost over the feasible topology $A_{round}$ is computed, and $A_{best}$ and the corresponding $UB$ are updated if it is the case. The heuristic returns $A_{best}$ and $UB$, which is used to compute an optimality gap with respect to the lower bound provided by Lagrangian relaxation.

**Algorithm 2** Two Phase Searching

Set $A_{best} = A_0$ and $UB = \sum_{e \in A_0} f_e + c^*(A_0)$
**for** round = 1 to 10 **do**
$Pool \leftarrow \emptyset$

**for** $p = 2\%, 4\%, 6\%, 8\%, 10\%$ of $|A_{best}|$ **do** ▷ Perturbation Phase
**for** $n = 1, \ldots, 5$ **do**
Generate $A_n$ by perturbation of $p$ arcs to $A_{best}$ ▷ Algorithm 3
Compute $L(A_n)$
$Pool = Pool \cup \{\arg\min_{n=1,\ldots,5}\{\sum_{e \in A_n} f_e + L(A_n)\}\}$

$A_{round} = \arg\min_{A' \in Pool}\{\sum_{e \in A'} f_e + L(A')\}$

**for** each unordered pair $\{A', A''\}$ of $Pool$ such that $A' \neq A''$ **do** ▷ Combination Phase
**for** $n = 1, 2, 3$ **do**
$A_n = A' \cap A''$
**for all** $e \in A' \triangle A''$, i.e., in the symmetric difference of the sets $A'$ and $A''$ **do**
Select a random number $q \in [0, 1]$
**if** $q \leq y_e^{cf}$ **then**
$A_n = A_n \cup \{e\}$
Compute $L(A_n)$
Update $A_{round}$ if $\sum_{e \in A_n} f_e + L(A_n) < \sum_{e \in A_{round}} f_e + L(A_{round})$

**if** $\sum_{e \in A_{round}} f_e + L(A_{round}) < UB$ **then**
Check/Restore feasibility of $A_{round}$
Solve the multicommodity flow problem over $A_{round}$
Update $A_{best}$ and $UB$ if $\sum_{e \in A_{round}} f_e + c^*(A_{round}) < UB$

**return** $A_{best}$ and $UB$

Algorithm 3 describes the perturbation phase. The inputs are the topology $A_{best}$, the frequency vector $y^{cf}$, and the number $p$ of arcs to be replaced. We use a threshold of 0.05, as in Algorithm 1, to select a set $\Phi \subseteq A \setminus A_{best}$ of unfixed arcs. Initially, topology $A_n$ is set to $A_{best}$, and then in the while-loop $p$ arcs are removed from $A_n$. Given a randomly chosen arc $e \in A_n$, the frequency value $y_e^{cf}$ is used as the probability for $e$ to be kept in $A_n$, so arcs with a high frequency have less chance to be removed. The procedure to insert arcs in $A_n$ is similar to

the one used in Algorithm 1. We compute new arc frequencies $y^f$ while solving the FCMC Lagrangian dual defined over $\Phi \cup A_n$ with the arcs in $\Phi$ unfixed. Note that the $p$ arcs that were removed from $A_n$ do not belong to $\Phi$. We select at most $p$ arcs with the highest $y^f$ values to be included in $A_n$, given that it must not be inferior to 0.3.

**Algorithm 3** Perturbations

Given $A_{best}$, the frequency vector $y^{cf}$, and $p$
$\Phi = \{e \in A : y_e^{cf} \geq 0.05\} \setminus A_{best}$
$A_n = A_{best}$
**while** less than $p$ arcs were removed from $A_n$ **do**
    Randomly choose an arc $e \in A_n$
    Select a random number $q \in [0, 1]$
    **if** $q \geq y_e^{cf}$ **then**
        $A_n = A_n \setminus \{e\}$
Set $y_e = 1$, $\forall e \in A_n$
Solve Lagrangian Dual for set $\Phi \cup A_n$ (i.e., (3.1) if using Volume, or (3.5) if using Bundle)
Given $y^f$, the frequency vector
Insert in $A_n$ at most $p$ arcs of $\Phi$ with the largest values of $y^f$, $y^f \geq 0.3$
**return** $A_n$

## 3.6 Computational experiments

To solve the Lagrangian dual, the two algorithms were implemented in C++, compiled with g++ version 4.8.5, using the flag -O3. Tests were run on a CentOS Linux 7 3.1 machine, Intel Xeon E5-2687W v3 3.10GHz, with 60 Gb RAM. The linear programs were solved with Cplex 12.7.0.0. The Volume implementation has been provided by the COIN-OR project `https://projects.coin-or.org/Vol` and the Bundle implementation by [91].

### 3.6.1 Instances

Instances were elaborated using the same generator used to generate benchmark instances of section 2.2.1, available at `http://www.di.unipi.it/optimize/Data/MMCF.html`. Eight groups of very-large-scale instances were generated for experiments in this work. Table 3.1 describes the features of each group of instances. The first column identifies the group. The second to the fourth columns show the network size. The fifth column shows the (C - F) ratios considered, 5 randomly

instances were generated for each (C - F) ratio. For example, Group A has 15 instances of 100 nodes, 1000 arcs, and 2000 commodities, 5 of them for each of the three cluster (C - F) ratios. Then, the sixth to the eighth columns show the number of binary variables, of continuous variables, and constraints, respectively, generated in the model given by (2.1) - (2.6) for each instance of the group. The goal has been to test large scale instances with different levels of difficulty. The higher are the ratios, the higher is the expectation to get a difficult instance, due to the large importance of fixed charges and capacity restrictions.

| Group | Nodes | Arcs | Comm. | (C - F) ratios | Binary Var | Continuous Var | Constraints |
|---|---|---|---|---|---|---|---|
| A | 100 | 1000 | 2000 | (10 - 0.5), (8 - 0.5), (14 - 0.5) | 1.0e3 | 2.0e6 | 2.2e6 |
| B | 100 | 1000 | 500 | (10 - 10), (6 - 10), (14 - 10) | 1.0e3 | 5.0e5 | 5.5e5 |
| C | 100 | 1000 | 800 | (2 - 10), (2 - 0.001), (14 - 0.001) | 1.0e3 | 8.0e5 | 8.8e5 |
| D | 100 | 1200 | 1000 | (1 - 20), (14 - 12), (20 - 0.001) | 1.2e3 | 1.2e6 | 1.3e6 |
| E | 100 | 2000 | 2000 | (1 - 20), (1 - 0.001), (20 - 0.001) | 2.0e3 | 4.0e6 | 4.2e6 |
| F | 100 | 5000 | 7000 | (1 - 20), (1 - 0.001), (20 - 20) | 5.0e3 | 3.5e7 | 3.6e7 |
| G | 100 | 8000 | 8000 | (20 - 0.001), (1 - 0.001), (20 - 20) | 8.0e3 | 6.4e7 | 6.5e7 |
| H | 200 | 12000 | 10000 | (1 - 20), (1 - 0.001), (20 - 20) | 1.2e4 | 1.2e8 | 1.2e8 |

Table 3.1: Characteristics of the instances generated for this study

In addition, the benchmark group Canad-N with 48 instances was also tested, corresponding to medium-size randomly generated problem instances. These instances were introduced by [96] and are available for download at `http://www.di.unipi.it/optimize/Data/MMCF.html`. In particular, there are 12 different network sizes, and 4 instances with different capacities and fixed-charge ratios were generated for each size. Table 3.2 presents these instances, a more detailed description can be found in [96]. Analogously to the previous table, in Table 3.2 the first column identifies the group. The second to the fourth columns show the network size, and the fifth to the seventh columns show the number of binary variables, of continuous variables, and constraints, respectively, generated in the model given by (2.1) - (2.6) for each instance of the group.

| Instances | Nodes | Arcs | Commodities | Binary Var | Continuous Var | Constraints |
|---|---|---|---|---|---|---|
| cN/ 01-04 | 20 | 299 | 100 | 3.0e2 | 3.0e4 | 3.2e4 |
| cN/ 05-08 | 20 | 298 | 200 | 3.0e2 | 6.0e4 | 6.4e4 |
| cN/ 09-12 | 20 | 297 | 400 | 3.0e2 | 1.2e5 | 1.3e5 |
| cN/ 13-16 | 20 | 300 | 800 | 3.0e2 | 2.4e5 | 2.6e5 |
| cN/ 17-20 | 30 | 599 | 100 | 6.0e2 | 6.0e4 | 6.3e4 |
| cN/ 21-24 | 30 | 599 | 200 | 6.0e2 | 1.2e5 | 1.3e5 |
| cN/ 25-28 | 30 | 598 | 400 | 6.0e2 | 2.4e5 | 2.5e5 |
| cN/ 29-32 | 30 | 597 | 800 | 6.0e2 | 4.8e5 | 5.0e5 |
| cN/ 33-36 | 50 | 1200 | 100 | 1.2e3 | 1.2e5 | 1.3e5 |
| cN/ 37-40 | 50 | 1200 | 200 | 1.2e3 | 2.4e5 | 2.5e5 |
| cN/ 41-44 | 50 | 1200 | 400 | 1.2e3 | 4.8e5 | 5.0e5 |
| cN/ 45-48 | 50 | 1200 | 800 | 1.2e3 | 9.6e5 | 1.0e6 |

Table 3.2: Characteristics of the benchmark instances considered in this study

### 3.6.2 Computational settings

The subgradient algorithms were tuned in order to obtain the best trade-off between good-quality bounds and fast convergence. We have set a limit of 1000 iterations and a time limit of 10000 seconds. Different calibrations were tested to keep the best one on average. The chosen configuration was then applied to the whole testbed. Parameters not discussed here were not crucial for the performances of the algorithms and so they were left as its default value.

To calibrate the Bundle implementation, the discussion made by [61] was taken as reference. In that paper, a few suggestions are made for parameters related to the bundle administration and the stepsize choice. Indeed, the settings proposed by the authors performed well, even for the newly generated instances. Furthermore, despite the absence of a similar study for the Volume, the best settings were, somehow, similar to the Bundle ones as precised below.

Concerning the Volume algorithm, the stepsizes are computed by $s_v^t = \rho(L_* - L(\bar{u}))/||d_v^t||^2$ as stated by [188], given that $L_*$ is a target value for the Lagrangian bound $L_* > L(\bar{u})$, which is updated as the bound $L(\bar{u})$ increases. An initial value is given to $\rho$, which is then updated accordingly to each iteration label, namely red, yellow, or green. In short, if no improvement is detected, the iteration is labeled red, on the contrary, the scalar $\mu_v = g^t \cdot d_v^t$ is computed. If $\mu_v \geq 0$ the iteration is called green, otherwise it is called yellow. On the other hand, in the case of the Bundle method, the step $s_b^t$ is set to an initial value and updated accordingly to the type of iterations classified as null or serious steps. Normally, a serious step happens if the real increase in the current bound matches some fraction of the predicted one, i.e. if $L(u^t) - L(\bar{u}) \geq 0.1\delta_\beta$, and a null step occurs in the opposite case. Good results were obtained fixing $s_b^0 = 1.0$ and $\rho^0 = 0.1$, with the slightly better bounds on average.

Usually, stepsizes are reduced after a series of iterations without improvements and can be enlarged when a profitable search direction is available. In the specific case of the Bundle implementation, the decrease also depends on whether $e^t \leq 0.3 \sum_{i \in \beta} \alpha_i e_i$, or not. If the condition is true, the decrease is inhibited, assuming that accurate first-order information has been gathered.

A minimum of 4 consecutive iterations without improvement before reducing the stepsize was established in both algorithms. It means that $\rho$ is diminished after $nr_v = 4$ red iterations and $s_b^t$ after $nn_b = 4$ null steps. Inversely, both $\rho$ and $s_b^t$ are increased after every green iteration and serious step. Even though yellow iterations in the Volume algorithm represent a gain in the current solution, the scalar product $\mu_v < 0$ indicates that the current subgradient may not be that interesting, so then, a more cautious rule might be preferable. It was also set a

minimum of 4 consecutive yellow iterations before $\rho$ is increased. Moreover, it might be important to point out the fact that, in both implementations, better bounds were obtained setting higher values for $nr_v$ and $nn_b$ (for example 10), but significantly more computational time was needed to achieve those bounds.

With respect to the procedure of increasing/reducing stepsizes, the Volume algorithm simply multiplies $\rho$ by a factor $0 < r < 2$, which can be 0.66, 1.1 and 2, for red, yellow and green iterations respectively, while the Bundle method presents a more complex rule, see [88] for a detailed explanation. Indeed, to update $s_b^t$, we have used the maximizer of the quadratic one-dimensional function $\phi(s) = L(\bar{u} + sd_b^t)$ which satisfies $\phi(0) = L(\bar{u})$, $\phi(s_b^t) = L(u^t)$ and $\phi^{'}(s_b^t) = g^t \cdot d_b^t = \mu_b$. That maximizer will be greater than $s_b^t$ if and only if $\mu_b > 0$, which resembles the yellow step policy present in the Volume algorithm. Practically the same function is used to perform stepsize decreases, but with the derivative in the current point $\phi^{'}(0) = \Delta_\beta$.

The red-yellow-green scheme for stepsize updates has been successfully adapted to classical subgradient-based methods, providing them better performances in terms of solution quality and number of iterations. In a stochastic optimization context, [76] obtained interesting results applying such a scheme to the Subgradient Algorithm. When compared to other NDO solvers, it managed to provide the best performance overall.

Still concerning the Bundle method, useful information can be derived from the solution of the quadratic problem (3.6), which can be helpful in the choice of $s_b^t$. Sophisticated heuristics exploiting those informations have been developed by [88]. For the present work, the heuristic implemented tends to increase $s_b^t$ whenever $\Delta_\beta$ is not great enough, compared to an ideal improvement: $\Delta^* = s^* || \sum_{i \in \beta} \alpha_i^t g_i ||^2 + \sum_{i \in \beta} \alpha_i^t e_i$, given the parameter $s^*$, considered as the longest step it can be taken. In other words, if $\Delta_\beta < 0.001\Delta^*$ the latest step performed is now useless and a larger one might be preferable. Sensitive analysis is used in order to ensure that $\Delta_\beta \geq \Delta^*$ in the next iteration.

In addition to step-control parameters, we need to consider too the ones related to the search direction management, which are closely related to how the bundle is updated in the Bundle method and how the value of $\theta$ is chosen throughout the Volume algorithm. Concerning the bundle $\beta$, a maximum size has been set to 10 items. In order to respect that limit, some old subgradients need to be discarded when a new one is proposed; indeed, a subgradient $g_i$ is called active if the dual variable $\alpha_i$ belongs to the optimal basis of the dual subproblem (3.6). Thus, we automatically discard items after 20 consecutive iterations being inactive. Moreover, If there is no removable item and an active one must be replaced, an

aggregation is done so that the information is not lost. In other words, a convex linear combination of the $|\beta|$ items is kept, using the current optimal vector $\alpha$ as multiplier.

Back to the Volume algorithm, a heuristic is used to choose the value for $\theta_t$ at each iteration $t$. In the intention to obtain an ascent direction, $\theta_t$ is set in order to have $(\theta_t g^t + (1-\theta_t)d_v^t) \cdot g^t \geq 0$ which can be estimated compared to $\theta_0 \|g^t\|^2$. This is done keeping $\theta_t \in [0,1]$ by:

$$\theta_t = \begin{array}{ll} \theta_0, & \text{if } \tau > 1 \\ \theta_0/10, & \text{if } \tau < 0 \\ \tau, & \text{otherwise} \end{array} \qquad \tau = max\{\theta_0, \frac{\theta_0 ||g^t||^2 - d_v^t g^t}{||g^t||^2 - d_v^t g^t}\} \tag{3.13}$$

The parameter $\theta_0$ is initially set to 1.0 and then decreased. More precisely, $\theta_0$ is multiplied by a factor 0.3, every time the current solution value has not improved by at least 1%. Accordingly to [26], the reductions are made in order to enhance the precision of the primal solution. A lower bound set to 0.01 allows the algorithm to stop decreasing $\theta_t$.

Finally, a third stopping criterium as stated by [95] was put in practice for both algorithms. At an iteration $t$, if the current search direction $d^t$ and its respective linearization error $\hat{e}^t$, w.r.t. the current point $\bar{u}$, are such that $s^*||d^t||^2 + \hat{e}^t \leq \epsilon|L(\bar{u})|$, the algorithm stops. As described before, $s^*$ is an estimate of the largest step that can be performed along $d^t$, and an interesting value for it was found to be one or two degrees of magnitude from the initial value set to $s_b^t$. Such observation has been made within the Bundle method context. Nevertheless, $s^*$ was set to 10 and $\epsilon$ to $10^{-4}$ in both implementations. The rationale behind this criterium is to stop the iterations if the maximum estimated improvement is less than a fraction of the current solution value. Alternatively, one can say that the condition indicates that both $||d^t||^2$ and $\hat{e}^t$ are relatively small enough. Although some modifications needed to be done in the Volume algorithm for the computation of the linearization error, they did not represent a significant loss in performance.

The settings concerning the Lagrangian heuristic discussed in Section 3.5 were almost all presented during the explanation of the algorithm. In general, a priority was given to the computational time performance, due to the very large scale instances. Therefore, the combinations were limited to 3 for each pair of topology, and for each percentage $p$ of arcs, only 5 perturbations are performed. The number of rounds was set to 10, with a computational time limit of 24h for the heuristic to run.

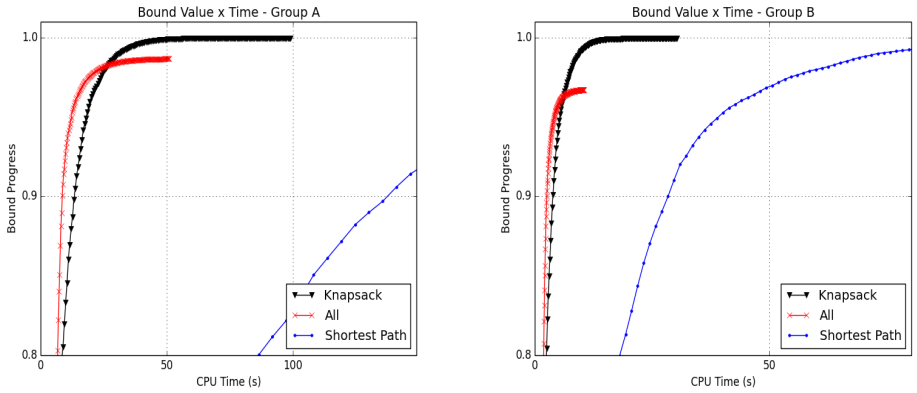
Still aiming the time consumption, the results obtained with the first relaxation (presented in 3.6.4) showed that the most crucial steps were usually performed in the first quarter of the optimization process, which led us to set a reduced limit

of 250 iterations to the NDO optimizers to solve each routing subproblem. When the solution value is not important and the solvers are only used in the interest of computing the frequency of unfixed arcs, the limit is set to 100 iterations. In addition, attempting to boost those secondary Lagrangian optimizations, the first dual solution obtained with the FCMC Lagrangian relaxation was used as hotstart.

### 3.6.3 Choice of relaxation

The main reason for choosing the 'knapsack' relaxation is that it provided the best overall numerical results in terms of time and quality solution. Using the Volume algorithm to solve Lagrangian duals, Figures 3.1 and 3.2 exemplify the results of previous tests that lead us to that choice. The same calibration, discussed previously, was used for the three relaxation approaches. In those figures, the average bound progression for each group of large-scale instances is given, related to the best bound obtained for each instance. The 'shortest path' relaxation provided good quality bounds, but it spent much more computational time, which damages the performances when dealing with very large instances as one can see in Figure 3.2d, where the values did not attain 80% of the best bound within the limit of time.

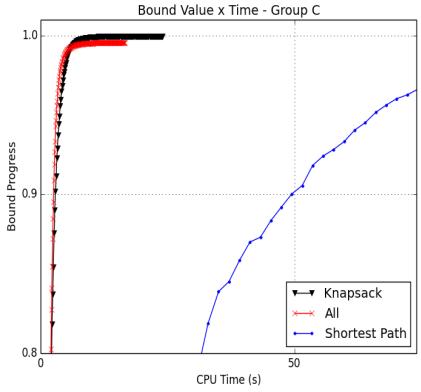
With respect to the 'total' relaxation, the computational times were lower than the 'knapsack' relaxation ones, for the largest instances. However, the quality of bound was worse in general, especially for the Bundle method. Furthermore, the same observations made in terms of the methods comparison with the 'knapsack' relaxation were also valid for the 'total' relaxation. See appendix A for additional information.



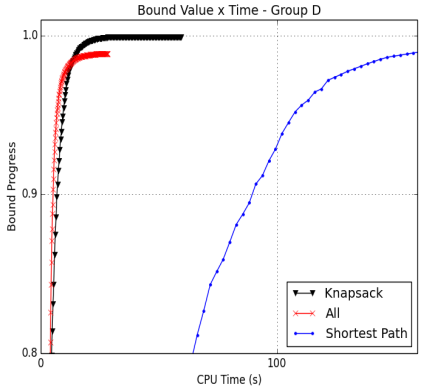

(a) Group A




(b) Group B

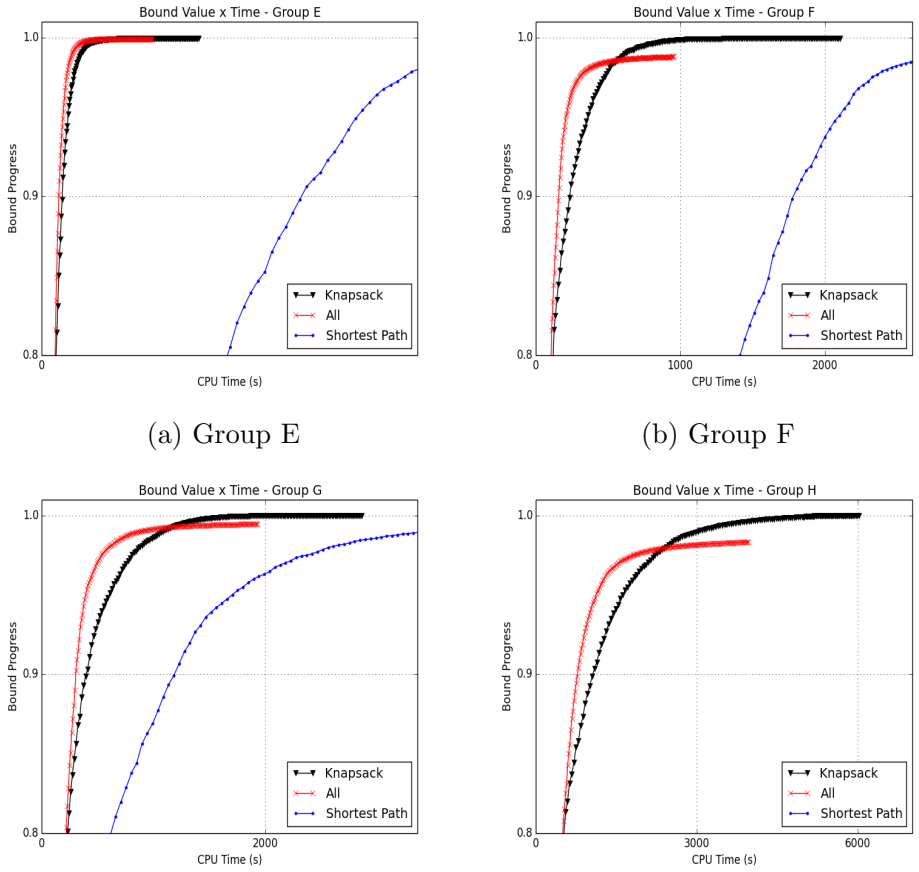



(c) Group C




(d) Group D

Figure 3.1: Average bound progression with respect to computation time, using Volume Algorithm for large instances of groups A-D




(a) Group E

(b) Group F

(c) Group G

(d) Group H

Figure 3.2: Average bound progression with respect to computation time, using Volume Algorithm for large instances of groups E-H

### 3.6.4 Computational results

We first analyse the computational performance of the Volume and Bundle methods to obtain lower bounds for the FCMC. Then, we analyse optimality gaps we can get with the lower and upper bounds procedures.

#### Computational performance of the Volume and Bundle methods

We compare Volume and Bundle methods in terms of the quality of the lower bound they provide, and in terms of time consumption. Because the Lagrangian subproblem resulting from the flow conservation constraints relaxation, i.e., (3.8) - (3.12), has the integrality property, the Lagrangian bound is theoretically equal to the linear relaxation bound. Thus, it means that in case the linear program is solved to optimality, one has a good quality proof for the lower bounds. We ran the Cplex Dual-Simplex algorithm in an attempt to get the linear relaxation

value. Unfortunately, we were not able to solve the linear relaxation for the large instances with 2000 or more commodities. Thus, we have the linear relaxation bound for instances Canad-N and for the groups B, C, and D. The linear relaxation of instances with low fixed costs C 02_001, C 14_001, and D 20_001, could be solved in a few minutes, whereas it can take a few days to solve the others.

A brief comparison in terms of memory consumption was also made. As expected, the Volume algorithm is more efficient in terms of RAM consumption, since the Bundle might have approximately $|\beta|$ times more data to store. Especially for group H, while the Volume algorithm consumed 3Gb on average for that group, the Bundle needed 14Gb of RAM, which can be an obstacle depending on the machine available to solve problem instances of that size.

Although [61] discusses the dynamic generation of Lagrangian variables aiming to decrease levels of time and RAM consumption, the same scheme can also be adapted to the Volume, as discussed by [95]. Moreover, results have shown that such a mechanism is not efficient for the decomposition considered here. The procedure tries to work only with the variables related to violated relaxed constraints, which leaves the possibility of saving time and memory, excluding variables that would have zero value throughout most part of the optimization process. That works very well when the capacity inequalities are being relaxed, but in the present case, it is very expected that all variables might be generated, given that equality constraints are more likely to be violated.

Table 3.3 presents average computational results. The first column indicates the group of instances. The line of instances Canad-N corresponds to average results for 48 instances, while the remaining lines correspond to average results for 5 instances with each (C - F) ratio in each group, see Table 3.1. Then, for each method, we report in column 'Deviation' the average deviations in percentage between the lower bound obtained with the method and the best lower bound obtained. In the case of instances of groups Canad-N, B, C, and D, the best lower bound is the linear relaxation value. Otherwise, the best lower bound is the best one given by either Volume or Bundle. Thus, for instances A, E, F, G, and H, a 0.00% deviation means that the method obtained the best lower bounds for all 5 instances with the given (C - F) ratio in that group. The best average results for each group of instances are highlighted in boldface. We report in column 'Time' the computational time in seconds.

For the experiments conducted, Table 3.3 shows that, on average, the Volume algorithm had better performance in terms of lower bound quality and time consumption than the Bundle method. The differences in computation times become evident when large instances are involved, which confirms that the fact of

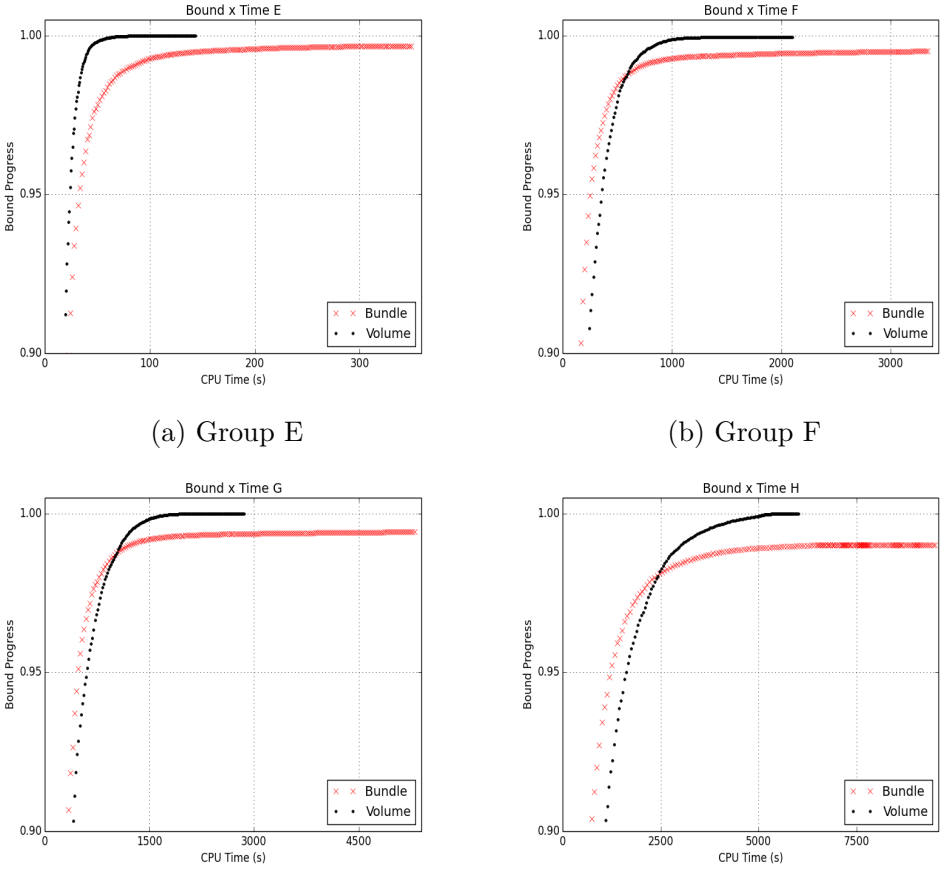
| Group (C_F) | Bundle | | Volume | |
|---|---|---|---|---|
| | Deviation(%) | Time(s) | Deviation(%) | Time(s) |
| Canad-N | 0.16 | 26.8 | **0.10** | 11.3 |
| A 08_05 | 0.66 | 171.8 | **0.00** | 89.3 |
| A 10_05 | 0.69 | 154.7 | **0.00** | 96.0 |
| A 14_05 | 0.80 | 161.2 | **0.00** | 111.6 |
| B 14_10 | 2.07 | 46.1 | **0.26** | 31.4 |
| B 10_10 | 2.53 | 46.0 | **0.28** | 30.6 |
| B 06_10 | 3.02 | 45.3 | **0.26** | 29.0 |
| C 02_10 | 1.59 | 65.0 | **0.22** | 40.2 |
| C 02_001 | **0.03** | 45.0 | 0.05 | 16.0 |
| C 14_001 | **0.04** | 46.6 | 0.05 | 16.1 |
| D 14_12 | 0.99 | 108.4 | **0.22** | 85.9 |
| D 20_001 | 0.10 | 73.9 | **0.05** | 26.2 |
| D 01_20 | 1.43 | 108.2 | **0.22** | 66.4 |
| E 01_001 | 0.09 | 300.9 | **0.01** | 100.2 |
| E 20_001 | 0.06 | 307.1 | **0.01** | 100.6 |
| E 01_20 | 0.81 | 438.4 | **0.00** | 229.1 |
| F 01_20 | 0.47 | 3508.3 | **0.00** | 2376.9 |
| F 20_20 | 0.40 | 4070.7 | **0.14** | 2989.5 |
| F 01_001 | 0.59 | 2436.3 | **0.00** | 947.0 |
| G 20_001 | 0.75 | 4717.4 | **0.00** | 2024.1 |
| G 20_20 | 0.10 | 6567.1 | **0.00** | 4430.3 |
| G 01_001 | 0.87 | 4614.1 | **0.00** | 2115.0 |
| H 01_20 | 0.60 | 9624.8 | **0.00** | 7217.5 |
| H 20_20 | 0.53 | 10006.8 | **0.00** | 9078.0 |
| H 01_001 | 1.62 | 9493.7 | **0.00** | 4152.1 |

Table 3.3: Average computational performance of the Lagrangian methods.

dealing with a quadratic problem at each iteration can be a bottleneck for the Bundle method. For example in group H, since the average time per iteration is much higher, the method performed fewer iterations due to the time limit, what consequently impacted the bound quality.

In general, taking as reference the cases in which the linear optimal could be obtained, the Lagrangian bounds were very close to the theoretical one, but without reaching the required precision. For almost all the testbed, both algorithms stopped because of either the time or the iteration limits were attained.

We plot in Figure 3.3 the bound progression by the computation time for the larger instances. The curves indicate the ratio between the current value $LB_x^t$ and $LB_{best}$ measured every 5 iterations and the respective computation time at that precise moment. The curves for groups E-H (Figures 3.3a-3.3d, see appendix B for the other groups of instances) confirm the already known fast convergence of the Bundle method. However, even though Bundle provides higher values in the very beginning, the Volume manages to follow and rapidly overtake it. In others words, we observed that the gain obtained at each step of the Bundle method is quite worthful at the beginning of the optimization process, but thanks to the faster iterations, the Volume manages to perform more steps in the same period of time, presenting better solutions already in the first quarter of the total elapsed time.




(a) Group E

(b) Group F

(c) Group G

(d) Group H

Figure 3.3: Average bound progression for the largest instances with respect to computation time

### Optimality gaps

From another perspective, we consider the impact of embedding the Lagrangian methods in the heuristic schemes proposed in Section 3.5 to produce feasible primal solutions for the FCMC. Table 3.4 presents results for the benchmark set of instances Canad-N. Since, to the best of our knowledge, there are no upper bounds reported in the literature for instances Canad-N, we ran Cplex with a time limit of 24 hours of CPU time, using 10 threads and all other parameters set as default. The first column indicates the instance. Then, for each method, we report (i) the optimality gap $(UB - LB)/UB$ in percentage between the upper bound $UB$ obtained with Algorithm 1 (the first feasible solution) and the lower bound $LB$ obtained with the Lagrangian method, (ii) the time in seconds to obtain the first upper bound, (iii) the optimality gap in percentage between the best upper bound with Algorithm 2 and the lower bound obtained with the Lagrangian method, (iv) the time in seconds to obtain the best upper bound, and (v) the deviation

$(UB - UB_{Cplex})/UB_{Cplex}$ in percentage of the best upper bound $UB$ obtained with the method with respect to the upper bound $UB_{Cplex}$ obtained with Cplex in 24 hours. Negative values of deviation mean that the heuristic managed to provide a better upper bound than the one obtained with Cplex, which is clearly outperformed for the largest Canad-N instances. Moreover, it is important to note that the computational times reported include the elapsed CPU time for the resolution of the FCMC Lagrangian dual to obtain the lower bound, plus the CPU time spent in the heuristic procedure. Thus, the total time to obtain the optimality gap reported.

It is interesting to see that for the largest Canad-N instances, from cN/29 to cN/48, the optimality gaps between the first feasible solution and the Lagrangian bound, obtained in few minutes, are much better than those obtained with Cplex in 24 hours for almost all such instances. The use of perturbations and combinations in Algorithm 2 lead much better solutions, but at the expense of higher computational times. Nevertheless, in less than 4 hours, the heuristic with both Volume and Bundle obtained optimality gaps of less than 15% for all instances, whereas Cplex in 24 hours obtained optimality gaps of more than 20% for 18 out of 48 instances, including gaps larger than 50% for 8 instances, and up to 84%. Therefore, regardless of the Lagrangian method used, the heuristic has been able to provide good quality solutions, being 6 times faster than Cplex in the worst case.

With respect to the comparison of the heuristic versions, the running times were relatively close. In terms of solution quality, the Bundle version did better for half of the instances, while the Volume version provided the best solutions for the other half of the Canad-N group. Indeed, the differences between each method become evident while working with groups A-H, when the large (and very large) instances are considered.

The results for the groups of large instances are presented in Tables 3.5, 3.6, and 3.7, using the same column nomenclatures as in Table 3.4, except for the columns of deviation with respect to Cplex. Given the very large size of instances in these groups, Cplex was not able to provide feasible solutions for them, even after 24 hours, so the optimality gap is the only solution quality measure.

Very small optimality gaps were obtained for instances with low values of F-ratio, i.e. C/2_001, C/14_001, and D/20_001. In addition, especially with the Volume version, good quality bounds were provided for lightly congested instances, i.e., D/1_20, F/1_001, and almost all instances of group E, since optimality gaps between 5% and 13% were obtained. Inversely, the hardest instances were the ones having high levels of F and C ratios.

| Instance | Bundle | | | | | Volume | | | | | Cplex |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1st Sol Gap(%) | Time (s) | 2nd Sol Gap(%) | Time (s) | Dev (%) | 1st Sol Gap(%) | Time (s) | 2nd Sol Gap(%) | Time (s) | Dev (%) | Gap (%) |
| cN/01 | 15.75 | 1 | 4.11 | 25 | 1.98 | 7.78 | 2 | **2.56** | 23 | 0.36 | 0.00 |
| cN/02 | 17.50 | 1 | 3.24 | 23 | 2.42 | 3.67 | 1 | **2.01** | 27 | 1.13 | 0.00 |
| cN/03 | 17.02 | 2 | 3.58 | 38 | 2.24 | 7.20 | 1 | **2.63** | 23 | 1.30 | 0.00 |
| cN/04 | 16.37 | 2 | 4.67 | 38 | 1.48 | 8.02 | 2 | **4.61** | 27 | 1.51 | 0.00 |
| cN/05 | 17.43 | 4 | **5.23** | 95 | 0.31 | 12.15 | 4 | 5.29 | 88 | 0.40 | 3.15 |
| cN/06 | 18.52 | 4 | 7.14 | 102 | 1.71 | 6.26 | 3 | **5.71** | 62 | 0.31 | 3.32 |
| cN/07 | 20.91 | 6 | **6.75** | 95 | 0.21 | 12.82 | 4 | 7.59 | 70 | 1.21 | 4.17 |
| cN/08 | 19.36 | 5 | 7.23 | 129 | 2.37 | 8.88 | 4 | **5.34** | 77 | 0.37 | 3.25 |
| cN/09 | 25.07 | 14 | **5.79** | 293 | 0.07 | 18.58 | 10 | 7.31 | 218 | 1.78 | 5.07 |
| cN/10 | 27.19 | 12 | 9.21 | 326 | 4.30 | 16.04 | 9 | **7.31** | 243 | 2.20 | 4.32 |
| cN/11 | 12.70 | 18 | 5.15 | 452 | 2.34 | 12.51 | 12 | **4.58** | 335 | 1.74 | 2.42 |
| cN/12 | 11.19 | 16 | 6.02 | 477 | 1.45 | 10.94 | 13 | **4.32** | 368 | -0.35 | 4.20 |
| cN/13 | 12.58 | 34 | **3.53** | 1098 | 0.10 | 11.27 | 30 | 3.76 | 747 | 0.35 | 3.10 |
| cN/14 | 17.42 | 43 | 3.33 | 1098 | -1.33 | 9.94 | 30 | **3.18** | 803 | -1.47 | 4.37 |
| cN/15 | 13.35 | 42 | **3.07** | 1355 | 0.36 | 12.54 | 30 | 3.41 | 948 | 0.72 | 2.50 |
| cN/16 | 12.08 | 45 | 6.19 | 1287 | 0.40 | 11.17 | 35 | **4.27** | 1103 | -1.62 | 5.57 |
| cN/17 | 15.59 | 6 | 8.43 | 84 | 2.38 | 14.31 | 4 | **8.23** | 80 | 2.19 | 5.51 |
| cN/18 | 13.39 | 7 | **6.47** | 83 | 1.47 | 8.63 | 3 | 6.87 | 59 | 2.03 | 3.56 |
| cN/19 | 20.53 | 5 | **9.53** | 69 | 2.25 | 16.24 | 3 | 10.09 | 65 | 3.01 | 6.43 |
| cN/20 | 14.76 | 8 | 6.08 | 75 | 2.27 | 6.61 | 3 | **5.81** | 51 | 2.11 | 2.11 |
| cN/21 | 26.62 | 13 | **9.25** | 232 | -2.98 | 19.45 | 10 | 10.23 | 195 | -1.86 | 11.42 |
| cN/22 | 26.59 | 12 | 14.25 | 254 | 4.19 | 20.59 | 10 | **11.93** | 204 | 1.50 | 10.19 |
| cN/23 | 19.12 | 15 | **8.61** | 351 | 0.44 | 15.70 | 13 | 9.28 | 288 | 1.24 | 7.73 |
| cN/24 | 18.52 | 14 | 13.04 | 361 | 5.61 | 15.19 | 12 | **8.87** | 293 | 0.84 | 7.69 |
| cN/25 | 10.42 | 62 | 3.82 | 1576 | -20.02 | 10.13 | 62 | **3.60** | 944 | -20.23 | 22.89 |
| cN/26 | 9.19 | 70 | **3.41** | 1880 | -2.37 | 9.68 | 67 | 3.90 | 1180 | -1.91 | 5.44 |
| cN/27 | 9.39 | 63 | **3.35** | 1536 | 0.43 | 10.47 | 53 | 3.55 | 942 | 0.62 | 2.72 |
| cN/28 | 10.28 | 61 | **3.64** | 1586 | 0.41 | 9.24 | 59 | 3.69 | 1026 | 0.43 | 3.01 |
| cN/29 | 16.99 | 110 | **5.01** | 2106 | -83.51 | 12.06 | 93 | 5.29 | 1843 | -83.46 | 84.33 |
| cN/30 | 12.79 | 139 | **3.65** | 2729 | -25.20 | 10.66 | 116 | 4.53 | 2081 | -24.49 | 27.85 |
| cN/31 | 17.57 | 101 | 4.22 | 2940 | -72.29 | 10.42 | 100 | **4.15** | 2125 | -72.31 | 73.44 |
| cN/32 | 10.91 | 130 | **4.19** | 2364 | -81.52 | 10.98 | 103 | 4.33 | 1891 | -81.48 | 82.27 |
| cN/33 | 17.84 | 22 | **11.27** | 386 | 1.10 | 16.35 | 13 | 12.21 | 266 | 2.19 | 8.07 |
| cN/34 | 15.43 | 28 | 11.62 | 402 | -16.42 | 14.87 | 17 | **10.45** | 262 | -17.51 | 24.14 |
| cN/35 | 15.22 | 26 | **11.51** | 522 | -0.79 | 14.51 | 17 | 11.60 | 346 | -0.70 | 10.24 |
| cN/36 | 15.68 | 33 | 10.10 | 518 | -1.76 | 15.07 | 18 | **9.90** | 303 | -1.98 | 9.50 |
| cN/37 | 24.44 | 28 | 17.18 | 560 | -31.48 | 19.07 | 24 | **12.26** | 426 | -35.27 | 42.74 |
| cN/38 | 15.67 | 56 | 9.69 | 536 | -27.52 | 13.36 | 29 | **8.90** | 375 | -28.04 | 34.01 |
| cN/39 | 29.32 | 22 | 19.83 | 431 | -37.01 | 19.14 | 22 | **14.51** | 352 | -40.83 | 49.13 |
| cN/40 | 15.62 | 88 | 9.30 | 657 | -48.77 | 10.53 | 55 | **8.14** | 335 | -49.30 | 53.20 |
| cN/41 | 12.09 | 185 | 5.94 | 3452 | -27.22 | 11.28 | 185 | **5.47** | 2461 | -27.51 | 31.29 |
| cN/42 | 21.39 | 66 | 12.62 | 1418 | -42.44 | 15.44 | 57 | **10.28** | 1101 | -43.85 | 49.46 |
| cN/43 | 16.02 | 95 | 8.04 | 1778 | -39.64 | 15.15 | 78 | **8.01** | 1582 | -39.58 | 44.29 |
| cN/44 | 19.51 | 85 | 13.82 | 1913 | -38.62 | 14.41 | 85 | **9.36** | 1659 | -41.48 | 46.82 |
| cN/45 | 17.23 | 358 | **6.44** | 6087 | -83.68 | 16.04 | 250 | **6.44** | 5050 | -83.67 | 84.71 |
| cN/46 | 11.80 | 743 | **4.16** | 13098 | -66.97 | 10.94 | 655 | 4.34 | 10775 | -66.90 | 68.30 |
| cN/47 | 15.13 | 725 | **4.85** | 9138 | -80.28 | 12.45 | 719 | 5.06 | 8573 | -80.23 | 81.21 |
| cN/48 | 14.85 | 471 | **5.31** | 10840 | -82.86 | 14.32 | 738 | 6.06 | 9655 | -82.71 | 83.74 |

Table 3.4: Feasible solutions for benchmark instances Canad-N.

| Instance | Bundle | | | | Volume | | | |
|---|---|---|---|---|---|---|---|---|
| | 1st Sol Gap(%) | Time (s) | 2nd Sol Gap(%) | Time (s) | 1st Sol Gap(%) | Time (s) | 2nd Sol Gap(%) | Time (s) |
| A/8_05a | 33.30 | 547 | 20.94 | 8349 | 27.60 | 213 | **18.11** | 4884 |
| A/8_05b | 33.49 | 477 | **17.69** | 8513 | 31.06 | 220 | 18.70 | 4941 |
| A/8_05c | 33.39 | 539 | 19.04 | 8498 | 29.86 | 221 | **18.00** | 5292 |
| A/8_05d | 31.73 | 284 | **23.59** | 7857 | 37.58 | 218 | 26.58 | 5558 |
| A/8_05e | 27.30 | 262 | **15.90** | 7975 | 28.88 | 208 | 17.11 | 5654 |
| A/10_05a | 33.89 | 541 | **20.09** | 8841 | 31.69 | 252 | 20.52 | 5775 |
| A/10_05b | 33.32 | 642 | **20.15** | 9015 | 32.67 | 283 | 21.25 | 6053 |
| A/10_05c | 33.68 | 557 | **20.72** | 9237 | 32.65 | 273 | 21.51 | 6071 |
| A/10_05d | 33.80 | 306 | 26.46 | 9081 | 39.08 | 240 | **26.36** | 5980 |
| A/10_05e | 29.12 | 340 | 20.70 | 8537 | 32.09 | 272 | **18.93** | 6130 |
| A/14_05a | 32.01 | 587 | **24.08** | 10660 | 38.77 | 296 | 25.75 | 7279 |
| A/14_05b | 33.10 | 585 | **23.06** | 10577 | 38.12 | 357 | 25.22 | 7274 |
| A/14_05c | 33.31 | 637 | 25.68 | 11218 | 39.88 | 308 | **25.01** | 7337 |
| A/14_05d | 39.53 | 673 | **28.08** | 9568 | 40.32 | 432 | 29.87 | 7391 |
| A/14_05e | 36.99 | 389 | 20.42 | 9540 | 32.48 | 497 | **19.60** | 7448 |
| Avg | 33.20 | 491 | **21.77** | 9164 | 34.18 | 286 | 22.17 | 6204 |
| | | | | | | | | |
| B/14_10a | 43.40 | 75 | 34.24 | 1542 | 36.59 | 57 | **28.78** | 1248 |
| B/14_10b | 45.11 | 63 | 31.02 | 1531 | 43.47 | 52 | **27.11** | 1308 |
| B/14_10c | 40.90 | 60 | 35.71 | 1830 | 34.22 | 62 | **27.68** | 1276 |
| B/14_10d | 46.18 | 62 | 36.03 | 1941 | 45.12 | 51 | **30.28** | 1281 |
| B/14_10e | 44.88 | 67 | 37.51 | 1800 | 41.38 | 53 | **30.48** | 1302 |
| B/10_10a | 45.51 | 61 | 39.21 | 1735 | 40.88 | 50 | **28.67** | 1130 |
| B/10_10b | 37.97 | 60 | 32.98 | 1208 | 42.90 | 49 | **26.14** | 1113 |
| B/10_10c | 36.94 | 58 | 34.39 | 1192 | 43.27 | 46 | **29.37** | 1190 |
| B/10_10d | 44.06 | 65 | 35.44 | 1702 | 38.87 | 52 | **28.22** | 1163 |
| B/10_10e | 42.73 | 63 | 35.76 | 1662 | 38.97 | 62 | **27.67** | 1220 |
| B/6_10a | 39.11 | 57 | 34.80 | 1512 | 36.11 | 47 | **24.01** | 1044 |
| B/6_10b | 33.72 | 57 | 31.80 | 1018 | 37.41 | 46 | **18.92** | 987 |
| B/6_10c | 36.76 | 57 | 28.99 | 1127 | 38.09 | 44 | **23.15** | 996 |
| B/6_10d | 37.89 | 59 | 30.34 | 1455 | 39.90 | 44 | **23.34** | 1055 |
| B/6_10e | 38.67 | 57 | 33.60 | 1262 | 40.40 | 52 | **21.49** | 1081 |
| Avg | 40.92 | 61 | 34.12 | 1501 | 39.84 | 51 | **26.35** | 1160 |
| | | | | | | | | |
| C/2_10a | 26.10 | 89 | 19.77 | 1215 | 21.74 | 69 | **9.73** | 1372 |
| C/2_10b | 25.48 | 84 | 8.36 | 1744 | 18.86 | 62 | **4.39** | 1495 |
| C/2_10c | 28.57 | 91 | 24.40 | 1514 | 27.79 | 73 | **15.27** | 1568 |
| C/2_10d | 28.88 | 90 | 12.36 | 1978 | 24.01 | 67 | **8.17** | 1305 |
| C/2_10e | 39.39 | 84 | 34.55 | 1614 | 29.14 | 65 | **21.53** | 1519 |
| C/2_001a | 6.03 | 207 | **0.39** | 4320 | 0.86 | 70 | 0.45 | 2146 |
| C/2_001b | 5.79 | 162 | **0.29** | 4197 | 0.76 | 84 | 0.36 | 2218 |
| C/2_001c | 5.99 | 235 | 0.48 | 4394 | 0.88 | 67 | **0.29** | 2170 |
| C/2_001d | 5.35 | 101 | 1.74 | 4366 | 2.01 | 136 | **1.19** | 3540 |
| C/2_001e | 3.26 | 190 | **0.29** | 4026 | 0.72 | 88 | 0.31 | 3064 |
| C/14_001a | 5.34 | 249 | 0.57 | 4311 | 1.10 | 96 | **0.37** | 2704 |
| C/14_001b | 5.70 | 188 | 0.32 | 4012 | 0.87 | 103 | **0.30** | 2446 |
| C/14_001c | 5.52 | 190 | 0.34 | 4295 | 0.77 | 97 | **0.19** | 2393 |
| C/14_001d | 5.49 | 98 | 1.55 | 4657 | 2.00 | 74 | **1.28** | 3313 |
| C/14_001e | 2.81 | 165 | **0.25** | 4563 | 0.70 | 100 | 0.38 | 3188 |
| Avg | 13.31 | 148 | 7.04 | 3414 | 8.81 | 83 | **4.28** | 2296 |

Table 3.5: Computational results for large scale instances with 100 nodes and 1000 arcs.

| Instance | Bundle | | | | Volume | | | |
|---|---|---|---|---|---|---|---|---|
| | 1st Sol Gap(%) | Time (s) | 2nd Sol Gap(%) | Time (s) | 1st Sol Gap(%) | Time (s) | 2nd Sol Gap(%) | Time (s) |
| D/14_12a | 44.76 | 166 | 35.95 | 4384 | 39.07 | 174 | **25.64** | 2793 |
| D/14_12b | 42.39 | 213 | 29.07 | 3490 | 40.72 | 157 | **27.35** | 2631 |
| D/14_12c | 41.07 | 164 | 34.82 | 2746 | 26.45 | 217 | **24.10** | 2292 |
| D/14_12d | 38.94 | 229 | 33.44 | 2878 | 40.70 | 170 | **29.88** | 2436 |
| D/14_12e | 35.56 | 132 | 35.14 | 2486 | 39.64 | 166 | **26.70** | 2758 |
| D/20_001a | 5.99 | 317 | 0.52 | 6161 | 1.25 | 155 | **0.47** | 3564 |
| D/20_001b | 5.86 | 382 | 0.39 | 5761 | 1.12 | 129 | **0.24** | 3020 |
| D/20_001c | 5.13 | 263 | 0.34 | 5387 | 0.48 | 102 | **0.20** | 2862 |
| D/20_001d | 4.73 | 181 | 1.83 | 5988 | 2.41 | 201 | **1.75** | 5646 |
| D/20_001e | 2.59 | 312 | 0.55 | 6383 | 0.89 | 188 | **0.31** | 3972 |
| D/1_20a | 19.77 | 129 | 16.79 | 1325 | 15.63 | 93 | **5.19** | 1454 |
| D/1_20b | 21.27 | 127 | 16.49 | 1462 | 16.79 | 95 | **4.93** | 1464 |
| D/1_20c | 25.51 | 128 | 21.60 | 1474 | 15.29 | 95 | **6.47** | 1422 |
| D/1_20d | 24.32 | 125 | 16.37 | 1403 | 14.56 | 91 | **4.42** | 1399 |
| D/1_20e | 24.03 | 129 | 16.06 | 1429 | 17.35 | 95 | **9.20** | 1441 |
| Avg | 22.80 | 200 | 17.29 | 3517 | 18.16 | 142 | **11.12** | 2610 |
| | | | | | | | | |
| E/1_001a | 10.47 | 872 | 4.98 | 17327 | 5.22 | 321 | **4.03** | 8848 |
| E/1_001b | 10.83 | 1022 | 2.25 | 13132 | 2.19 | 246 | **1.02** | 5152 |
| E/1_001c | 13.37 | 1503 | 2.79 | 16202 | 2.16 | 269 | **1.33** | 5848 |
| E/1_001d | 10.58 | 1580 | 4.48 | 17786 | 3.66 | 637 | **3.25** | 11458 |
| E/1_001e | 14.87 | 800 | 6.73 | 20827 | 5.00 | 285 | **3.97** | 8143 |
| E/20_001a | 10.46 | 936 | 4.89 | 16657 | 5.32 | 424 | **3.76** | 9224 |
| E/20_001b | 8.36 | 1256 | 2.11 | 13114 | 2.64 | 317 | **1.51** | 6378 |
| E/20_001c | 13.04 | 1269 | 2.85 | 15457 | 2.47 | 242 | **1.61** | 6130 |
| E/20_001d | 9.91 | 1615 | 4.45 | 18267 | 3.62 | 912 | **3.23** | 13154 |
| E/20_001e | 13.75 | 903 | 6.20 | 20110 | 5.09 | 294 | **3.71** | 8090 |
| E/1_20a | 33.06 | 469 | 23.17 | 4043 | 17.37 | 322 | **8.49** | 3046 |
| E/1_20b | 34.00 | 479 | 23.90 | 4108 | 21.98 | 309 | **13.41** | 3043 |
| E/1_20c | 31.88 | 463 | 24.19 | 3904 | 15.60 | 317 | **6.10** | 3092 |
| E/1_20d | 33.43 | 449 | 21.17 | 3856 | 15.77 | 314 | **9.05** | 3089 |
| E/1_20e | 49.55 | 401 | 41.52 | 5220 | 24.74 | 257 | **19.55** | 2969 |
| Avg | 19.84 | 934 | 11.71 | 12667 | 8.86 | 364 | **5.60** | 6511 |

Table 3.6: Computational results for large scale instances with 100 nodes and 1000 and 1200 arcs.

| Instance | Bundle | | | | Volume | | | |
|---|---|---|---|---|---|---|---|---|
| | 1st Sol Gap(%) | Time (s) | 2nd Sol Gap(%) | Time (s) | 1st Sol Gap(%) | Time (s) | 2nd Sol Gap(%) | Time (s) |
| F/1_20a | 39.40 | 5190 | 24.03 | 25285 | 23.91 | 3115 | **12.74** | 18025 |
| F/1_20b | 36.39 | 5575 | 28.37 | 24076 | 17.11 | 3771 | **9.28** | 18502 |
| F/1_20c | 30.76 | 4691 | 23.63 | 19545 | 23.57 | 3984 | **12.78** | 20478 |
| F/1_20d | 54.91 | 3947 | 42.46 | 30138 | 22.87 | 2516 | **21.10** | 15588 |
| F/1_20e | 53.40 | 4188 | 41.74 | 30421 | 26.68 | 2797 | **21.17** | 16941 |
| F/20_20a | 53.50 | 7533 | 46.00 | 64773 | 46.82 | 7573 | **39.07** | 45601 |
| F/20_20b | 57.76 | 5583 | 47.09 | 73930 | 61.81 | 5945 | **46.01** | 53075 |
| F/20_20c | 49.10 | 8214 | 39.07 | 88295 | 43.27 | 7389 | **31.29** | 65425 |
| F/20_20d | 55.91 | 6374 | 47.85 | 52991 | 59.87 | 6047 | **47.69** | 45614 |
| F/20_20e | 54.59 | 6082 | 46.42 | 52949 | 49.85 | 5864 | **39.74** | 42237 |
| F/1_001a | 21.89 | 5509 | 11.10 | 61139 | 10.58 | 2431 | **9.87** | 27260 |
| F/1_001b | 21.41 | 6140 | 11.71 | 71669 | 9.97 | 2429 | **9.91** | 27203 |
| F/1_001c | 18.70 | 6522 | 9.52 | 49269 | 7.86 | 2163 | **7.31** | 22922 |
| F/1_001d | 21.89 | 4758 | 10.92 | 55667 | 10.58 | 2282 | **9.87** | 25699 |
| F/1_001e | 23.11 | 9239 | 13.47 | 92924 | 10.93 | 16835 | **10.63** | 82305 |
| Avg | 39.52 | 5354 | 29.56 | 46677 | 28.38 | 5009 | **21.90** | 35125 |
| G/20_001a | 29.37 | 12259 | **14.01** | 100695 | 15.87 | 5036 | 15.19 | 37794 |
| G/20_001b | 27.99 | 11625 | **14.62** | 101960 | 17.49 | 5979 | 17.24 | 45540 |
| G/20_001c | 49.45 | 8115 | 31.82 | 87603 | 16.58 | 7403 | **15.85** | 57861 |
| G/20_001d | 35.54 | 12027 | **17.87** | 89948 | 20.79 | 5294 | 20.03 | 40918 |
| G/20_001e | 43.86 | 9253 | 23.51 | 86630 | 21.55 | 8014 | **20.59** | 61858 |
| G/20_20a | 61.62 | 10484 | 53.00 | 73758 | 55.75 | 10605 | **44.54** | 53434 |
| G/20_20b | 50.13 | 14308 | 43.20 | 91850 | 49.46 | 11756 | **40.90** | 70321 |
| G/20_20c | 58.04 | 10021 | 47.05 | 93149 | 54.08 | 6705 | **43.86** | 46828 |
| G/20_20d | 57.31 | 12358 | 48.36 | 89119 | 61.14 | 11047 | **43.25** | 73745 |
| G/20_20e | 56.79 | 9380 | 45.60 | 71617 | 45.37 | 8538 | **36.86** | 46055 |
| G/1_001a | 31.36 | 8826 | **15.69** | 74962 | 25.86 | 5453 | 23.88 | 37899 |
| G/1_001b | 31.17 | 11190 | **15.46** | 77377 | 25.27 | 4477 | 24.16 | 33981 |
| G/1_001c | 27.95 | 10582 | **13.97** | 77408 | 20.65 | 4483 | 18.53 | 32896 |
| G/1_001d | 27.95 | 10855 | **13.91** | 76142 | 20.65 | 4576 | 18.53 | 32530 |
| G/1_001e | 42.34 | 7417 | 24.85 | 90230 | 16.75 | 5439 | **16.29** | 57802 |
| Avg | 42.06 | 10580 | 28.19 | 85497 | 31.15 | 6987 | **26.65** | 48631 |
| H/1_20a | 36.85 | 10642 | 35.07 | 105030 | 17.25 | 11086 | **13.11** | 58245 |
| H/1_20b | 29.47 | 10775 | 10.95 | 96943 | 13.70 | 11161 | **7.99** | 57296 |
| H/1_20c | 39.34 | 10761 | 37.26 | 71769 | 15.30 | 11090 | **13.40** | 59788 |
| H/1_20d | 60.46 | 10447 | 54.78 | 87719 | 22.40 | 11455 | **22.40** | 62955 |
| H/1_20e | 62.99 | 9511 | 59.82 | 88810 | 23.63 | 9362 | **23.63** | 56172 |
| H/20_20a | 48.73 | 22538 | 45.55 | 86904 | 48.37 | 20747 | **43.09** | 90358 |
| H/20_20b | 46.82 | 37007 | 37.82 | 90910 | 44.54 | 22426 | **35.52** | 90787 |
| H/20_20c | 45.43 | 12363 | **42.75** | 89430 | 52.11 | 14826 | 46.22 | 92250 |
| H/20_20d | 60.05 | 10923 | 54.96 | 87368 | 51.63 | 14838 | **45.37** | 88430 |
| H/20_20e | 67.52 | 11109 | 63.59 | 88644 | 52.41 | 13720 | **49.81** | 92306 |
| H/1_001a | 57.13 | 19209 | 53.71 | 96211 | 38.60 | 19091 | **36.88** | 90135 |
| H/1_001b | 53.72 | 26888 | 50.04 | 100843 | 21.31 | 14250 | **20.48** | 86912 |
| H/1_001c | 60.18 | 21270 | 54.21 | 94343 | 45.42 | 16653 | **45.19** | 88491 |
| H/1_001d | 39.88 | 33522 | 32.31 | 87016 | 31.70 | 16574 | **29.97** | 88221 |
| H/1_001e | 41.50 | 23836 | 37.88 | 96031 | 17.24 | 76101 | **17.24** | 89783 |
| Avg | 50.00 | 18053 | 44.71 | 91198 | 33.04 | 18892 | **30.02** | 79475 |

Table 3.7: Computational results for large scale instances with 100 and 200 nodes and up to 12000 arcs.

Time consumption can be considered satisfactory having in mind the size of the instances. At each round, a total of 55 Lagrangian duals may be optimized (5 per perturbation, being 5 different percentages of perturbation and 3 per combination, being 10 combinations given a pool of 5 topologies) with 250 iterations allowed, what would have been impracticable if one had to deal with LPs. Moreover, we remind that after the first phase (Algorithm 1), arcs with low frequency were neglected, leading to a reduced set of unfixed variables to be explored, what also reduced computational times. We observe that the Volume version of the heuristic is noticeably faster, taking the average as reference (lines 'Avg'). We note that, except for group H, it was able to perform all the rounds before the time limit, while the Bundle version struggled to complete the rounds already for instances in group G and F (with smaller instance sizes than in H).

With respect to the gaps obtained, the Volume version of the heuristic clearly outperforms the Bundle version. Except for group A, and some instances of group G, the heuristic with the Volume algorithm obtained better results on average and quite often significantly smaller gaps. The Volume version of the heuristic was able to provide optimality gaps within 30% for all instances of groups A, B, C, D, and E. For the very large instances of group F, G, and H, the average optimality gaps obtained by the Volume version of the heuristic are within 30%, and for less than half of instance in these groups, 17 out of 45, the optimality gaps exceeded 30% remaining below 50%. We remark that, to the best of our knowledge, this is the first study in the literature to deal with such large instances, which are by far larger than the benchmark instances Canad-N. We also remark that the gap between lower and upper bounds include the potentially high 'natural' gap between the linear relaxation value and the optimal one. Therefore, even though we have gaps reaching 40%, the deviation to the optimal value is possibly smaller. For example, the heuristic solution using Volume for cN/01 in Table 3.4 has a gap of 2.56%, but it is only 0.36% away from the optimal.

Finally, we make some remarks about the constructive Algorithm 1 and the operations put in practice on the search for an improving solution. With respect to the procedure implemented to build a first feasible solution, the results obtained showed that it is able to produce in a fast manner a solution within 50% for almost all instances. We point out the case of instance H/1_001e, for which the Volume version of Algorithm 1 ran for a much longer period, leaving practically no time for further improvements in phase two. However, in this case the Volume version of Algorithm 1 obtained a good quality solution with an optimality gap of 17%. We conclude this section by noticing that in the improving phase Algorithm 2 managed to significantly reduce gaps, especially when the time limitation was not

an issue.

### Comparisons with state-of-the-art heuristics

A third experiment was conducted to compare the performance of the heuristic schemes proposed in Section 3.5 with state-of-the-art heuristics from the literature for the FCMC. For such purpose, sets Canad-C and Canad-R of instances introduced by [63] were used, since these smaller instances have been served as benchmark for several heuristics proposed in the literature. Groups Canad-C and Canad-R have 43 and 153 instances, respectively. These instances are also available for download at `http://www.di.unipi.it/optimize/Data/MMCF.html`.

We compare our approach with the following heuristics and metaheuristics: (i) CYCLE, the cycle-based tabu search by [110]; (ii) RELINK, the path relinking by [111]; (iii) MULTI, the multilevel cooperative search by [65]; (iv) SCALE, the capacity scaling heuristic by [141]; and (v) LCBR, the local branching heuristic by [193].

Table 3.8 presents results for the group Canad-C. The results are presented as the gap in percentage between the upper bound obtained with the heuristic and the best upper bound among all the 7 heuristics used in this comparison. A gap of zero means that the heuristic obtained the best result, and are highlighted in boldface. The first column in Table 3.8 identifies the instances. The size corresponds to the number of nodes $|N|$, arcs $|A|$, and commodities $|K|$. The letters stand for tight (T) or loose (L) capacities, and high fixed charges are indicated by F and the contrary by V. Then, the second to the sixth columns present results for each heuristic from the literature. The solution values obtained with CYCLE, RELINK, MULTI, and SCALE are reported in [141], and the ones obtained with LCBR are reported in [193]. The seventh and eighth columns present results of the best solutions obtained with the heuristic schemes proposed in Section 3.5 using Volume and Bundle to solve the Lagrangian problems, respectively. The last line of Table 3.8 presents the average gap. Table 3.9 presents results for instances of group Canad-R. For each size, there is a number of instances with different capacity and cost ratios. As far as we know, only [110] and [141] have reported solution values for this group. Table 3.9 presents, for instances of the same size, average gaps in percentage between the upper bound obtained with the heuristic and the best upper bound among the 4 heuristics used in this comparison.

For group Canad-C, average and maximal running times of the heuristic scheme with Volume were 90 and 439 seconds, respectively, and with Bundle 150 and 709, respectively. For group Canad-R, average and maximal running times with Volume were 30 and 337 seconds, respectively, and with Bundle 29 and 299, respectively.

The proposed heuristic scheme had a similar performance using Volume or Bundle, the former obtaining better results on average for group R and the latter for group C. Results have shown that both versions of the proposed heuristic scheme are competitive with the state-of-the-art-heuristics. Indeed, it was able to find new best upper bounds for some cases and outperformed most of them in terms of average gaps to the best solution. We note that SCALE due to [141] comes out of our experiments as the best performing heuristic in terms of average gaps to the best solution.

| Instance | CYCLE | RELINK | MULTI | SCALE | LCBR | Volume | Bundle |
|---|---|---|---|---|---|---|---|
| 20,230,40, VL | 0.22 | 0.13 | 0.67 | 0.05 | **0.00** | 0.26 | **0.00** |
| 20,230,40, VT | 0.11 | 0.09 | **0.00** | 0.12 | **0.00** | 0.10 | 0.04 |
| 20,230,40, FT | 0.43 | 0.39 | 1.51 | 0.22 | **0.00** | 0.27 | 0.35 |
| 20,230,200, VL | 4.80 | 6.13 | 4.40 | **0.00** | 1.10 | 0.91 | 0.54 |
| 20,230,200, FL | 6.07 | 6.99 | 3.85 | **0.00** | 4.05 | 0.51 | 0.87 |
| 20,230,200, VT | 6.48 | 6.42 | 3.98 | **0.00** | 0.07 | 0.28 | 0.23 |
| 20,230,200, FT | 7.64 | 7.74 | 3.58 | **0.00** | 3.54 | 1.15 | 0.68 |
| 20,300,40, VL | 0.03 | **0.00** | 0.10 | **0.00** | **0.00** | **0.00** | **0.00** |
| 20,300,40, FL | 1.22 | 0.74 | 1.26 | 0.29 | **0.00** | 0.46 | 0.59 |
| 20,300,40, VT | 0.05 | **0.00** | 0.32 | 0.01 | **0.00** | 0.03 | **0.00** |
| 20,300,40, FT | 0.48 | 0.95 | 2.42 | **0.00** | **0.00** | **0.00** | **0.00** |
| 20,300,200, VL | 7.31 | 4.18 | 4.22 | **0.00** | 1.91 | 0.14 | 0.09 |
| 20,300,200 , FL | 6.13 | 6.24 | 5.06 | **0.00** | 2.82 | 0.79 | 0.39 |
| 20,300,200, VT | 5.42 | 4.52 | 2.52 | **0.00** | 1.14 | 0.09 | 0.91 |
| 20,300,200, FT | 6.08 | 5.34 | 3.28 | 0.31 | 2.08 | 0.65 | **0.00** |
| 30,520,100, VL | 1.70 | 1.60 | 3.10 | 0.11 | **0.00** | 0.35 | 0.16 |
| 30,520,100, FL | 4.80 | 7.11 | 5.03 | **0.00** | 1.51 | 0.15 | 0.47 |
| 30,520,100, VT | 1.62 | 1.67 | 2.58 | 0.29 | **0.00** | 0.37 | 0.95 |
| 30,520,100 , FT | 6.43 | 6.97 | 3.65 | 0.11 | 2.34 | **0.00** | 0.29 |
| 30,520,400, VL | 6.47 | 5.50 | 2.44 | **0.00** | 1.33 | 0.42 | 0.64 |
| 30,520,400, FL | 7.23 | 8.38 | 4.57 | **0.00** | 5.25 | 0.91 | 0.63 |
| 30,520,400, VT | 5.71 | 4.60 | 5.24 | **0.00** | 0.52 | 0.22 | 0.20 |
| 30,520,400, FT | 9.05 | 6.68 | 4.66 | **0.00** | 9.38 | 0.53 | 0.85 |
| 30,700,100, VL | 1.64 | 2.30 | 2.59 | 0.07 | **0.00** | 0.10 | 0.07 |
| 30,700,100, FL | 3.64 | 4.59 | 5.59 | **0.00** | 0.13 | 0.73 | 1.46 |
| 30,700,100, VT | 2.38 | 2.76 | 3.27 | 0.57 | **0.00** | 0.83 | 0.59 |
| 30,700,100, FT | 4.81 | 2.60 | 3.17 | 0.46 | **0.00** | 0.24 | 0.60 |
| 30,700,400, VL | 8.25 | 6.80 | 4.54 | **0.00** | 5.60 | 0.67 | 0.70 |
| 30,700,400, FL | 9.32 | 6.87 | 6.20 | **0.00** | 20.44 | 0.45 | 0.43 |
| 30,700,400, VT | 6.26 | 5.84 | 3.92 | **0.00** | 1.42 | 0.45 | 0.85 |
| 30,700,400, FT | 8.85 | 7.70 | 5.87 | **0.00** | 10.20 | 0.41 | 1.25 |
| 25,100,10, FL | **0.00** | **0.00** | **0.00** | 0.64 | **0.00** | 6.64 | 6.64 |
| 25,100,10, FT | **0.00** | **0.00** | 0.08 | 1.72 | **0.00** | 2.67 | 2.85 |
| 25,100,10, VL | **0.00** | **0.00** | **0.00** | **0.00** | **0.00** | **0.00** | **0.00** |
| 25,100,30, FL | 0.69 | 0.87 | 0.75 | 0.39 | **0.00** | 4.18 | 1.33 |
| 25,100,30, FT | 0.89 | 1.04 | 1.08 | 0.32 | **0.00** | 0.40 | **0.00** |
| 25,100,30, VT | 0.03 | 0.03 | 0.03 | **0.00** | **0.00** | 0.40 | 0.11 |
| 100,400,10, FL | **0.00** | 0.30 | 0.30 | 2.09 | 3.00 | 8.60 | 2.23 |
| 100,400,10, FT | 2.59 | **0.00** | 1.52 | 11.27 | 3.09 | 7.98 | 10.35 |
| 100,400,10, VL | 0.89 | 0.22 | 0.46 | 0.01 | **0.00** | 0.28 | 0.28 |
| 100,400,30, FL | 3.26 | 2.83 | 1.16 | 4.01 | **0.00** | 2.77 | 3.40 |
| 100,400,30, FT | 2.61 | **0.00** | 2.99 | 2.05 | 0.19 | 5.22 | 2.72 |
| 100,400,30, VT | 0.18 | 0.03 | 0.12 | 0.02 | **0.00** | 2.34 | 3.13 |
| Avg | 3.53 | 3.19 | 2.61 | **0.58** | 1.89 | 1.25 | 1.11 |

Table 3.8: Comparisons for benchmark instances of group C.

| $\|N\|$ | $\|A\|$ | $\|K\|$ | Number of Instances | CYCLE | SCALE | Volume | Bundle |
|---|---|---|---|---|---|---|---|
| 10 | 25 | 10 | 6 | **0.00** | 0.62 | 0.07 | **0.00** |
| 10 | 25 | 25 | 6 | 0.77 | 0.11 | 0.02 | **0.00** |
| 10 | 25 | 50 | 6 | 1.59 | 0.07 | **0.02** | **0.02** |
| 10 | 50 | 10 | 9 | **0.08** | 0.42 | 0.78 | 1.06 |
| 10 | 50 | 25 | 9 | **0.23** | 0.27 | 0.66 | 0.73 |
| 10 | 50 | 50 | 9 | 2.26 | **0.06** | 0.10 | 0.37 |
| 10 | 75 | 10 | 9 | 0.05 | 0.88 | **0.06** | 1.18 |
| 10 | 75 | 20 | 9 | 0.60 | 0.48 | **0.45** | 0.70 |
| 10 | 75 | 50 | 9 | 2.65 | **0.20** | 0.37 | 0.30 |
| Avg | | | | 0.93 | 0.36 | **0.31** | 0.54 |
| | | | | | | | |
| 20 | 100 | 40 | 9 | 2.22 | **0.21** | 0.53 | 0.93 |
| 20 | 100 | 100 | 9 | 2.44 | **0.02** | 0.12 | 0.27 |
| 20 | 100 | 200 | 9 | 4.62 | **0.00** | 0.32 | 0.43 |
| 20 | 200 | 40 | 9 | 2.51 | **0.40** | 0.66 | 0.95 |
| 20 | 200 | 100 | 9 | 5.98 | **0.06** | 0.75 | 0.75 |
| 20 | 200 | 200 | 9 | 6.83 | **0.04** | 0.49 | 0.96 |
| 20 | 300 | 40 | 9 | 2.28 | **0.17** | 0.73 | 0.79 |
| 20 | 300 | 100 | 9 | 5.25 | **0.01** | 0.68 | 0.76 |
| 20 | 300 | 200 | 9 | 9.34 | **0.17** | 0.80 | 1.07 |
| Avg | | | | 4.61 | **0.12** | 0.56 | 0.77 |

Table 3.9: Comparisons for benchmark instances of group R.

## 3.7 Conclusion

In summary, the Bundle and the Volume have both provided good quality bounds. Both methods computed similar 'best' lower bounds for large to very large network instances, but they mostly reached the time limit or the max iteration counter set by the user without the possibility to prove convergence within a given accuracy. In other words, the question of defining a reliable stopping criterion which takes profit of the behavior of both algorithms remains a difficult issue.

With respect to the comparison of the lower bounds, one can say that, for the tests put in practice, the Volume algorithm has performed well no matter the instance characteristics, while the Bundle has performed worse for a specific group. One can conclude that the Volume algorithm manages to be more robust, in the sense that it might be successful for a wider range of different instance configurations. Moreover, in a large scale context, the time consumption may be a bottleneck for the Bundle method, but its ability to provide good solutions in the early iterations can be very profitable for small problems. We remark that such conclusions are drawn upon results obtained with Volume and Bundle for the FCMC, which is a structured problem, and therefore should not be immediately extrapolated for other types of problems. As future research, it is suggested to extend the computational comparisons to a set of non-structured problems.

Regarding the estimation of good upper bounds, one can say that the results obtained with the Volume were better on average. Finally, we provide the first feasible solutions for the very large scale instances considered and not solved in the literature. Furthermore, the heuristic using Lagrangian information within a

perturbation and combination scheme could obtain good quality solutions (with reduced gaps) for some benchmark instances.

Provided that, for the largest instances, the best performances were obtained with the Volume Algorithm, in the next chapter we present a Relax-and-Cut algorithm that uses the Volume as NDO solver for the 'knapsack' relaxation. This same solver is used for the development of a feasibility pump heuristic in chapter 5.

# Chapter 4

# Volume-Based Relax-and-Cut

Here, we present the Relax-and-Cut algorithm implemented using the Volume Algorithm. We considered the cover inequalities, minimum cardinality inequalities, and the flow pack and flow cover inequalities of section 2.4 to be added *on-the-fly* to Lagrangian dual in a relaxed way. Unfortunately, the gains with the addition of such inequalities were not expressive on average, but it can be worthful for some instances. Moreover, two additional features were considered to enhance performances, which indeed provided improvements in all cases.

## 4.1 Introduction

Based on Lagrangian relaxation, Relax-and-Cut algorithms attempt to improve Lagrangian bounds, by dynamically introducing valid inequalities as violated constraints, to strengthen the current relaxed model. The addition of such inequalities can be done after the relaxation is solved (and the problem is then reoptimized), or during the optimization process, what Lucena [160] called, Delayed Relax-and-Cut and Non-Delayed Relax-and-Cut, respectively. Moreover, the new constraints are usually relaxed in a Lagrangian fashion, to avoid having to deal with harder subproblems.

The main principles of the Relax-and-Cut were earlier discussed by Balas and Christofides [23] for the *traveling salesman problem*, and later by Gavish [97], for a *centralized network design problem*. Since then, successful Relax-and-Cut implementations have been proposed for different problems in the literature (see for example [44, 68, 77, 159, 70, 151, 9, 83]). Moreover, Guignard [117] discusses the efficiency of violated cuts in improving Lagrangian bounds, concluding that cuts using integer information are more likely to be helpful, while surrogate constraints cannot improve bounds.

As mentioned in section 2.3.1, Kliewer and Timajev [150] proposed, for the

FCMC problem, a Relax-and-Cut procedure dedicated to solve node LPs in Branch-and-Bound schemes. In their work, only cover inequalities were used, with the addition of another type of locally valid cuts. In the present work, however, we implement 4 types of globally valid inequalities, with the intention of producing a bounding procedure on its own. Moreover, we propose two alternatives to boost performances and an effective cutset separation procedure.

In the implemented algorithm, cutset based inequalities valid for the FCMC problem are identified (see section 2.4) and added to the Lagrangian problem. Since the number of cutsets is exponential, and so is the number of inequalities, efficient cutset and constraint separation procedures are necessary. We start describing the Relax-and-Cut algorithm itself, then the following four sections explain the separation procedures for cutsets and inequalities. In sequence, we present the two propositions to enhance performances, and at the end of this chapter, the results obtained with the proposed algorithm are shown, followed by a conclusion.

## 4.2 The Relax-and-Cut algorithm

In the present case, violated cover, minimal cardinality, flow pack, and flow cover inequalities (described in section 2.4) are dynamically added to the knapsack relaxation (see 3.4) every iteration where a better dual solution (in terms of bound value) is found. Although other types of valid inequalities could be separated, here we follow the results obtained in [47, 49], specifically for the case of the FCMC problem.

According to Lucena [159], the addition of inequalities in a non-delayed routine is often preferable, being our natural choice for implementation. In this work, the strong arc formulation (2.1)-(2.6) is considered throughout the whole discussion, except in section 4.8.4, where the weak formulation (which excludes the strong forcing constraints (2.4)) is used to compare the cutset generation capability of producing strong cover inequalities, in comparison to the one proposed in [49].

Moreover, considering that the Volume Algorithm provides us a linear relaxation approximate solution $(\tilde{x}, \tilde{y})$, we also test the case where, to be introduced, inequalities must be violated by both the current Lagrangian subproblem solution $(\hat{x}, \hat{y})$ (which is mixed-integer), and the approximate one (which is continuous). In this later case, a simple inspection procedure takes place after the inquality is identified, to check if it is violated or not by $(\tilde{x}, \tilde{y})$. Hence, the procedures to identify violated inequalities (separation procedures) take into account the mixed-integer vector $(\hat{x}, \hat{y})$, only.

**Algorithm 4** The Volume-based Relax-and-Cut Algorithm

Given an inicial $\bar{\lambda}$

Compute $L(\bar{\lambda})$, to get solution $\hat{x}$ and subgradient $g^0$.

$\tilde{x} \leftarrow \hat{x}$, $t \leftarrow 1$, $h \leftarrow g^0$

**do**

Compute stepsize $s^t \leftarrow \rho \frac{UB - L(\bar{\lambda})}{||h||^2}$

Take step: $\lambda^t \leftarrow \bar{\lambda} + s^t h$

Solve $L(\lambda^t)$ to get the solution $\hat{x}$

Sensitivity Analysis:

If($flag$) then maximize $L(\lambda)$ given $\hat{x}$, to get $\lambda^{t'}$

else $\lambda^{t'} \leftarrow \lambda^t$

Separate and add violated cuts.

Compute subgradient $g^t$

Check if *green*, *yellow* or *red* iteration

Update $\rho$ and $UB$ target

**if** $L(\lambda^{t'}) > L(\bar{\lambda})$ **then**

$\bar{\lambda} \leftarrow \lambda^{t'}$; $L(\bar{\lambda}) \leftarrow L(\lambda^{t'})$

Compute $\alpha$

$\tilde{x} \leftarrow \alpha\hat{x} + (1 - \alpha)\tilde{x}$

$h \leftarrow \alpha g^t + (1 - \alpha)h$

Look for removable cuts.

$t \leftarrow t + 1$

**while** Stopping criteria not reached

The implemented Relax-and-Cut scheme is formally presented in the Algorithm 4. The procedure is quite the same as the Volume Algorithm described in section 3.2: for a given initial dual point, the Lagrangian subproblem is solved, and a subgradient is obtained, being used as first search direction. At each iteration, the subproblem is solved for a new trial point, obtained with a previously computed stepsize and search direction. Once the subproblem is solved, a *sensitivity analysis* procedure (to be discussed in section 4.7) may take place, according to the $flag$, considering $flag = true$ if sensitivity analysis must be executed, or $flag = false$ if it must not.

After obtaining the solution for the Lagrangian subproblem, cutsets and cutset-based inequalities are separated. The violated inequalities are then added to the problem. In addition, the new entries of vector $h$ (the search direction) are set to the value obtained from the difference between the left and right-hand sides of the newly inserted inequalities; that, considering vector $\tilde{x}$, the linear relaxation

approximate solution. Afterwards, the usual Volume operations take place. At the end of each iteration, based on what is discussed in [68] about active sets of dual variables, cuts that have not been violated after a certain number of consecutive iterations are deleted, along with their respective dual variable.

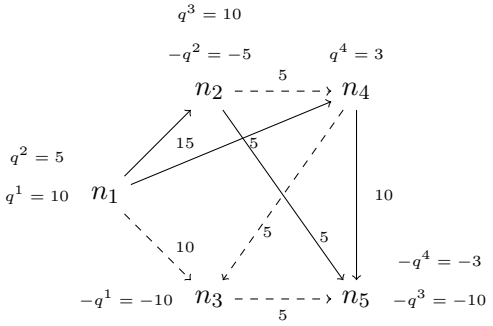
## 4.3 Cutset generation

Recall that a cutset is defined as a set of arcs $(S, \bar{S}) = \{(i,j) \in A : i \in S,\ j \in \bar{S}\}$, given a partition of nodes $S \subset N$ and its complement $\bar{S} = N \setminus S$. Furthermore, associated with each cutset, there is a set of crossing commodities $K_{S\bar{S}} = \{k \in K : O(k) \in S, D(k) \in \bar{S}\}$, and the reverses: $(\bar{S}, S)$ and $K_{\bar{S}S}$, defined analogously.

The cutsets are generated during the Volume iterations, using the solutions from the Lagrangian subproblem. In fact, considering an interval of 50 iterations, we use the best solution $(\hat{x}, \hat{y})$, in terms of value, produced in that interval, to generate new cutsets to be added to a collection $CS$. The procedure for the cutset generation is described by Algorithm 5. The main goal of this heuristic separation is to obtain cutsets with low installed capacity and high amounts of crossing demand, increasing the chances of finding violated valid inequalities.

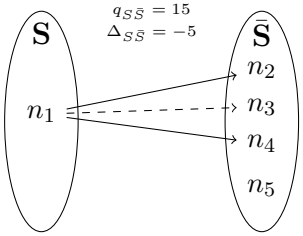
The algorithm starts with a single node (singleton), having its respective installed capacity $U_{S\bar{S}} = \sum_{a \in (S,\bar{S}):\hat{y}_a = 1} w_a$ and transpassing demand $q_{S\bar{S}} = \sum_{k \in K_{S\bar{S}}} q^k$, recalling that $\hat{y} \in \{0,1\}^{|A|}$. The goal is to move nodes from $\bar{S}$ to $S$, in order to obtain a cutset with high $\Delta_{S\bar{S}} = q_{S\bar{S}} - U_{S\bar{S}}$.

Node movements are done accordingly to its benefit to the value of $\Delta_{S\bar{S}}$, namely how much it may increase that value. To measure $\delta_j$ the benefit of node $j \in \bar{S}$, we compute $q^+_{S\bar{S}}$: the amount of demand to be added to $q_{S\bar{S}}$, minus the amount of demand to be retrieved from it, once moving $j$ from $\bar{S}$ to $S$. The same is done for the installed capacity, in order to obtain $U^+_{S\bar{S}}$, and to compute $\delta_j = q^+_{S\bar{S}} - U^+_{S\bar{S}}$. Then, the most increasing node is moved, and the values of $q_{S\bar{S}}$ and $U_{S\bar{S}}$ are updated. This procedure is repeated until no increasing node is encountered. An example reproducing a node movement is given in Figure 4.1.



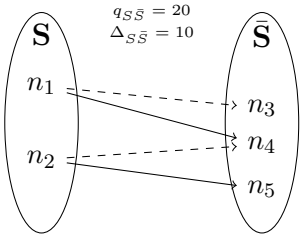

An example of 5 nodes, 8 arcs, and 4 commodities. The demands $q^k$ are given next to the origins, while their negative value is given next to their destinations. The capacities are given next to the arcs. The current Lagrangian subproblem solution is represented by dashed (closed) and solid (opened) arcs.

(a) Graph example




We first set an initial singleton. The candidate nodes are $n_2$ and $n_4$. For $n_2$ one have $q^+_{S\bar{S}}(n_2) = 10 - 5$ and $U^+_{S\bar{S}}(n_2) = 5 - 15$, so $\delta_{n_2} = 15$

(b) Initial cutset




Note that $n_2$ is chosen, since for $n_4$ one have $q^+_{S\bar{S}}(n_4) = 3$ and $U^+_{S\bar{S}}(n_4) = 10 - 5$, so $\delta_{n_4} = -2$

(c) Cutset after first iteration

Figure 4.1: Example of an iteration of the cutset generation, Algorithm 5

Note that the nodes to be evaluated are only the ones corresponding to the endpoint of an installed arc in the cutset. Moreover, as the movements are performed, every current configuration presenting positive $\Delta_{S\bar{S}}$ is added to the cutset collection. However, if only $\Delta_{S\bar{S}}$-negative configurations are obtained, the last one generated is kept.

Once a cutset $(S, \bar{S})$ is identified, the reverse cutset $(\bar{S}, S)$ is also added to collection $CS$, since for a node separation $S$ and $\bar{S}$, the cutset containing arcs with start-point in $\bar{S}$ and end-point in $S$ is automatically available. Furthermore, the same procedure of Algorithm 5 is implemented from the perspective of $\bar{S}$, that is to say it starts with $\bar{S}$ as a singleton and $S = N \setminus \bar{S}$. It suffices to modify the inner 'for all' loop to

$$\textbf{for all } j \in S : \exists\ i \in \bar{S}, \hat{y}_{ji} = 1 \textbf{ do}$$

and to multiply by $-1$ the computed values: $q^+_{S\bar{S}}$ and $U^+_{S\bar{S}}$. The $S$ and $\bar{S}$ updating steps are then restated accordingly to traduce the nodes movements from $S$ to

$\bar{S}$. In that way, we intend to generate a larger and more diversified collection of cutsets.

**Algorithm 5** Cutset Generation

Given the collection of cutsest $CS$
Given the solution $(\hat{x}, \hat{y})$
**for all** $n \in N$ **do**
  $S \leftarrow \{n\}$, $\bar{S} \leftarrow N \setminus S$
  $q_{S\bar{S}} \leftarrow \sum_{k \in K_{S\bar{S}}} q^k$
  $U_{S\bar{S}} \leftarrow \sum_{a \in (S,\bar{S}):\hat{y}_a = 1} w_a$
  **while** true **do**
    **for all** $j \in \bar{S} : \exists\, i \in S, \hat{y}_{ij} = 1$ **do**
      $q^+_{S\bar{S}} \leftarrow \sum_{k \in K:O(k)=j, D(k) \in \bar{S}} q^k - \sum_{k \in K:O(k) \in S, D(k)=j} q^k$
      $U^+_{S\bar{S}} \leftarrow \sum_{v \in \bar{S}:\hat{y}_{jv}=1} u_{jv} - \sum_{i \in S:\hat{y}_{ij}=1} u_{ij}$
      $\delta_j \leftarrow q^+_{S\bar{S}} - U^+_{S\bar{S}}$
    Get $j^*$ with the greatest $\delta_j, \delta_j \geq 0$
    **break** if no such $j^*$
    $\bar{S} \leftarrow \bar{S} \setminus \{j^*\}$
    $S \leftarrow S \cup \{j^*\}$
    $q_{S\bar{S}} \leftarrow q_{S\bar{S}} + q^+_{S\bar{S}}(j^*)$
    $U_{S\bar{S}} \leftarrow U_{S\bar{S}} + U^+_{S\bar{S}}(j^*)$
    $\Delta_{S\bar{S}} \leftarrow q_{S\bar{S}} - U_{S\bar{S}}$
    **if** $\Delta_{S\bar{S}} > 0$ **then**
      $CS \leftarrow CS \cup \{(S, \bar{S})\}$
      $CS \leftarrow CS \cup \{(\bar{S}, S)\}$
Add the current $(S, \bar{S})$ (and $(\bar{S}, S)$) to $CS$ if not added yet.
**return** $CS$

In the following three sections we discuss the cutset-based inequalities separation procedures. We first describe the separation of minimal cardinality inequalities and cover inequalities. Then, the two following sections describe the separation of flow cover and flow pack inequalities, respectively.

## 4.4 Minimal cardinality and cover separation methods

Before discussing about the constraint separation procedures, we note that given a cutset $(S, \bar{S})$, one can redefine the set of transpassing commodities $K_{S\bar{S}}$

with the use of a metric, setting arc lengths equal to 1, if in $(S, \bar{S})$, otherwise equal to 0. If the shortest path between the origin and destination of commodity $k \in K$ has value $\pi^k > 0$, the commodity is included in $K_{S\bar{S}}$. Consequently, the total demand value may be modified to $q^{\pi}_{S\bar{S}} = \sum_{k \in K_{S\bar{S}}} \pi^k q^k$. One must consider this redefinition in the remainder of this section.

### 4.4.1 Cover inequalities

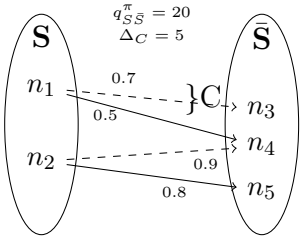
Given the total crossing demand $q^{\pi}_{S\bar{S}}$, for any cutset $(S, \bar{S})$, a cover can be found, or proven to be nonexistent by solving (4.1), where $z \in \{0,1\}^{|(S,\bar{S})|}$ represents the characteristic vector of the cover $C$.

$$\begin{aligned} Z = min \sum_{a \in (S,\bar{S})} \hat{y}_a z_a \\ \sum_{a \in (S,\bar{S})} w_a (1 - z_a) < q^{\pi}_{S\bar{S}} \\ z_a \in \{0,1\}, \forall\ a \in (S,\bar{S}) \end{aligned} \tag{4.1}$$

Since, $\hat{y}$ is binary vector, the solution of the separation problem is easily computed by a greedy algorithm. We first define $R = \{a \in (S,\bar{S}) : \hat{y}_a = 0\}$ and $C_1 = \{a \in (S,\bar{S}) : \hat{y}_a = 1\}$, and compute $\Delta_C = q^{\pi}_{S\bar{S}} - \sum_{a \in C_1} w_a$. Note that a cover $C$ must correspond to a subset of $R$, since a violated cover inequality must present $\hat{y}_a = 0\ \forall\ a \in C$, and $\Delta_C$ must be positive.

With the aim of producing inequalities (2.19) more likely to be violated throughout the Volume iterations, thus to keep a non-zero Lagrangian multiplier, a restriction over $R$ is applied. Such restriction is performed by artificially opening arcs in the set, i.e., moving arcs from $R$ to $C_1$, guaranteeing that $\Delta_C$ remains positive. In the present case, we chose to sequentially open arcs with $\tilde{y}_a \geq 0.9, a \in (S,\bar{S})$, in a nonincreasing order of $\tilde{y}$. Remind that $\tilde{y}$ comes from the linear relaxation approximate solution provided by the Volume, and $\hat{y}$ comes from the solution of the Lagrangian subproblem.

Finally, to form a minimal cover C, the arcs in the restricted set $R$ are sequentially examined, also in a nonincreasing order of $\tilde{y}$, so that an arc $a \in R$ with $w_a > \Delta_C$ is included in the cover C. Otherwise it is moved to $C_1$, which consequently decreases, by the arc capacity, the value of $\Delta_C$ that is updated. Figure 4.2 presents an example of cover, based on the cutset obtained in Figure 4.1c.




Values of $\tilde{y}_a$ are given next to the arcs. Applying the cover generation Algorithm 6 on the cutset of Figure 4.1c, one obtains the cover C identified in this figure.

Figure 4.2: Example of cover obtained from the cutset of Figure 4.1c

The Algorithm 6 describes de whole separation procedure. Note that, only minimal covers are considered, and that a lifting procedure must take place for global validity, since inequalities are defined for restrictions on the cutsets. The lifting procedures are discussed below, in subsection 4.4.3.

**Algorithm 6** Cover generation

Given $(\tilde{x}, \tilde{y})$, the current linear relaxation approximate solution.
Given $(\hat{x}, \hat{y})$, the solution to the Lagrangian subproblem.
Let $CI$ be the collection of covers
Let $(S, \bar{S})$ be the cutset
Let $R = \{a \in (S, \bar{S}) : \hat{y}_a = 0\}$
Let $C_1 = \{a \in (S, \bar{S}) : \hat{y}_a = 1\}$
Let the cover $C \leftarrow \emptyset$
$\Delta_C \leftarrow q^{\pi}_{S\bar{S}} - \sum_{a \in C_1} w_a$
**if** $\Delta_C < 0$ **then return** $CI$
**for all** $a \in R$ in nonincreasing order of $\tilde{y}_a$ **do**
  **if** $\tilde{y}_a \geq 0.9$ and $\Delta_C - w_a > 0$ **then**
    $R \leftarrow R \setminus \{a\}$
    $C_1 \leftarrow C_1 \cup \{a\}$
    $\Delta_C \leftarrow \Delta_C - w_a$
**for all** $a \in R$ in nonincreasing order of $\tilde{y}_a$ **do**
  **if** $w_a \geq \Delta_C$ **then**
    $C \leftarrow C \cup \{a\}$
  **else**
    $C_1 \leftarrow C_1 \cup \{a\}$
    $\Delta_C \leftarrow \Delta_C - w_a$
$CI \leftarrow CI \cup C$
**return** $CI$

### 4.4.2 Minimal cardinality inequalities

The separation procedure for minimal cardinality inequalities (2.21) is generally trivial. It suffices to compute $m_{(S,\bar{S})} = 1 + max\{i : \sum_{j=1...i} w_{a_j} < q^{\pi}_{S\bar{S}}\}$, having arcs $a_j \in (S,\bar{S}), j = 1, ..., |(S,\bar{S})|$, in a nonincreasing order of $w_{a_j}$, and to verify whether $\sum_{a\in(S,\bar{S})} \hat{y}_a - m_{(S,\bar{S})} \geq 0$. Nevertheless, the same restriction as for cover inequalities is performed in this case too.

In other words, the sets $R$ and $C_1$ are defined as for the cover inequalities, and the same restriction procedure is applied on $R$, originating the set $C$. Then, the minimal cardinality $m_C$ is defined over the restricted set $C$, giving $\sum_{a\in C} y_a \geq m_C$. As well as in the cover case, the separated minimal cardinality inequalities must be lifted for them to be globally valid.

### 4.4.3 Lifting

In the present context, the lifting technique consists in elevating an inequality to a higher dimension by introducing outer variables into it. Consider the general valid inequality (4.2), where $C$ may represent either a cover or a restricted cutset. In the case of cover inequalities $b = 1$, while $b = m_C$ for minimal cardinality inequalities.

$$\sum_{a\in C} y_a \geq b \tag{4.2}$$

According to Proposition 2, given the set $C$ and set $C_1 = (S,\bar{S}) \setminus C$, with arcs in $C_1$ considered to be open, the following inequality is also valid:

$$\sum_{l\in C_1} \gamma_l y_l + \sum_{a\in C} y_a \geq b + \sum_{l\in C_1} \gamma_l \tag{4.3}$$

**Proposition 2.** *Consider* $P = \{x \in \mathbb{B}^n : Ax \geq b\}$, $L = P \cap \{x \in \mathbb{B}^n : x_1 = 0\}$, $S = P \cap \{x \in \mathbb{B}^n : x_1 = 1\}$, *and that* $\sum_{i=2}^{n} \pi_i x_i \geq \pi_0$ *is valid for* $S$. *If* $L = \emptyset$, $x_1 \geq 1$ *is valid for* $P$. *If* $L \neq \emptyset$, *then for any* $\gamma_1 \leq Z - \pi_0$,

$$\gamma_1 x_1 + \sum_{i=2}^{n} \pi_i x_i \geq \pi_0 + \gamma_1$$

*is valid for* $P$, *where* $Z = min\{\sum_{i=2}^{n} \pi_i x_i : x \in L\}$.

*Proof.* For a solution $\bar{x} \in P$, If $\bar{x} \in L$, since by definition $\gamma_1 \leq Z - \pi_0$:

$$\gamma_1 \bar{x}_1 + \sum_{i=2}^{n} \pi_i \bar{x}_i = \sum_{i=2}^{n} \pi_i \bar{x}_i \geq Z \geq \pi_0 + \gamma_1.$$

If $\bar{x} \in S$, since $\sum_{i=2}^{n} \pi_i \bar{x}_i \geq \pi_0$ is valid for S:

$$\gamma_1 \bar{x}_1 + \sum_{i=2}^{n} \pi_i \bar{x}_i = \gamma_1 + \sum_{i=2}^{n} \pi_i \bar{x}_i \geq \pi_0 + \gamma_1.$$

□

As discussed in [24], one has to solve a sequence of binary knapsack problems to define coefficients $\gamma_l, l \in C_1$, for the lifted inequalities (4.3). Namely, for a set of already lifted variables $B$, and $M = C_1 \setminus \{B\}$, in case of lifting a variable $y_l$, $l \in M$, the following integer program need to be solved:

$$\begin{gathered} Z = min \sum_{a \in B} \gamma_a y_a + \sum_{a \in C} y_a \\ \sum_{a \in B \cup C} w_a y_a \geq q^{\pi}_{S\bar{S}} - \sum_{a \in M} w_a + w_l \\ y_a \in \{0,1\}, \forall\ a \in C \cup B \end{gathered} \tag{4.4}$$

Still according to Proposition 2, the coefficient $\gamma_l = Z - b - \sum_{a \in B} \gamma_a$, if a solution is found. However, if no feasible solution exists, $y_l = 1$ is valid, which is equivalent to set $\gamma_l = |C|$ (as stated in [46]). Indeed, if one rewrite inequality (4.2) for the group of already lifted variables $B$, and the variable $y_l$ being lifted, the inequality (4.5) is obtained. Since $\gamma_l = |C| \geq (\sum_{a \in B} \gamma_a y_a + \sum_{a \in C} y_a - \sum_{a \in B} \gamma_a)$, $y_l$ must be 1, so (4.5) can be satisfied.

$$\gamma_l y_l + (\sum_{a \in B} \gamma_a y_a + \sum_{a \in C} y_a - \sum_{a \in B} \gamma_a) \geq b + \gamma_l \tag{4.5}$$

Remark that the inequality resulting from the lifting procedure described depends on the sequence in which variables are lifted. According to Gu *et al.* [115], given a feasible fractional point $y^*$, and a minimal cover $C$, the problem of deciding if there exists a violated lifted cover inequality is NP-complete. In the present case, a binary point is provided, and only already violated inequalities are considered for lifting. Moreover, note that the variables to be lifted can only contribute to the violation.

In that sense, the lifting order is defined heuristically, using the approximate solution $\tilde{y}$ given by the Volume, to drive the sequence. Variables $y_l$, $l \in C_1$ are

lifted in a nonincreasing order of $\tilde{y}_l$. Moreover, even though the sequential lifting procedure is more commonly employed for the present case, in [213], Zemel discussed the simultaneous lifting procedure, which adds the possibility to generate more than one lifted cover inequality at a time, but presenting a higher computational cost.

Despite the fact that 0-1 knapsack programs must be solved, several dynamic programming algorithms, for example the ones proposed in [186, 168], exist in the literature, that can efficiently tackle these problems. This technique consists in breaking down a problem into simpler subproblems, solving them recursively from the smallest to the biggest, while storing the solutions to avoid recomputations.

In the present case, a dynamic programming algorithm based on the recursion (4.6) was implemented, considering $u$ the right-hand-side of the knapsack constraint, its maximal value $U$, and that $\gamma_a = 1,\ \forall\ a \in C$. The Algorithm 7 details the procedure. By abuse of notation, consider $a_m = m$, so $w_m$ stands for $w_{a_m}$, for example.

$$\Phi_m(u) = \begin{cases} min\{\gamma_m, \Phi_{m-1}(u)\}, & \text{if } u = 1, ..., w_m \\ min\{\Phi_{m-1}(u), \Phi_{m-1}(u - w_m) + \gamma_m\} & \text{if } u = w_m + 1, ..., U \end{cases} \tag{4.6}$$

**Algorithm 7** Dynamic Programming Lifting Procedure

Let $n = |C|$ the number of cover variables, with indexes $\{1, ..., n\}$

Let $n_1 = |C_1|$ the number of variables to be lifted, with indexes $\{n+1, ..., n_2 = n + n_1\}$ in nonincreasing order of $\tilde{y}$;

Let $max\_w = max\{w_l : n + 1 \leq l \leq n_2\}$

Let $max\_u = q^{\pi}_{S\bar{S}}$

$L \leftarrow \emptyset$; $m \leftarrow 1$; $W \leftarrow w_m$; $v \leftarrow min\{w_m, max\_u\}$

**for** $u = 1$ **to** $v$ **do**

$\Phi_m(u) \leftarrow \gamma_m$

**for** $u = v + 1$ **to** $min\{W + max\_w, max\_u\}$ **do**

$\Phi_m(u) \leftarrow \infty$

$strt \leftarrow 2$; $end \leftarrow n$

**for** $l = n + 1$ **to** $n_2$ **do**

**for** $m = strt$ **to** $end$ **do**

$W \leftarrow W + w_m$

$v \leftarrow min\{w_m, max\_u\}$

$v_2 \leftarrow min(W, max\_u)$

$v_3 \leftarrow min(W + max\_w, max\_u)$

**for** $u = 1$ **to** $v$ **do**

$\Phi_m(u) \leftarrow min\{\gamma_m, \Phi_{m-1}(u)\}$

**for** $u = v + 1$ **to** $v_2$ **do**

$\Phi_m(u) \leftarrow min\{\Phi_{m-1}(u), \Phi_{m-1}(u - w_m) + \gamma_m\}$

**for** $u = v_2 + 1$ **to** $v_3$ **do**

$\Phi_m(u) \leftarrow \infty$

$U = q^{\pi}_{S\bar{S}} - \sum_{a \in C_1 \setminus B} w_a + w_l$

**if** $U \leq 0$ **then** $Z \leftarrow 0$

**else if** $W \geq U$ **then** $Z \leftarrow \Phi_{end}(U)$

**else** $Z \leftarrow -1$

**if** $Z \geq 0$ **then** $\gamma_l = Z - b - \sum_{a \in B} \gamma_a$

**else** $\gamma_l = |C|$

$B \leftarrow B \cup \{l\}$

$strt \leftarrow end \leftarrow l$

**return** Lifted coefficients $\alpha_l, l \in L$

The main idea of the algorithm is to start considering only one variable of index $m = 1$ in the 0-1 knapsack problem, and to solve it for all possible values of $u$, one by one, that is $u = 1, ..., w_1$. For higher values of $u$, the problem becomes currently infeasible, and the solution value is set to $\infty$. Then, the second variable

of index $m = 2$ is inserted, and the problem is solved for $u = 1, ..., w_1 + w_2$, which will need the already computed values of $m = 1$. In that manner, variables keep being inserted sequentially, until all of them have been considered. The insertion of a variable $m$ in the problem corresponds to a stage, and $\Phi_m(u)$ correspond to the solution value of stage $m$ for a certain $u$.

Note that to solve the first lifting problem, one has to compute the values of stages $m = 1, ..., |C|$, while only stage $m = |C| + 1$ need to be computed, to solve a second lifting problem. In other words, lifting a variable $l_1 \in C_1$ means to automatically solve subproblems of the lifting problem of variable $l_2 \in \{C_1 \setminus \{l_1\}\}$. Finally, observe that the complexity of the algorithm in the worst case is $O(nq^{\pi}_{S\bar{S}})$, $n$ being the total number of variables.

## 4.5 Flow cover separation method

To identify violated flow cover inequalities (2.23), given a mixed-integer point $(\hat{x}, \hat{y})$, one have to separate a flow cover $(C_1, C_2)$, $C_1 \subseteq (S, \bar{S})$ and $C_2 \subseteq (\bar{S}, S)$, such that $\mu = \sum_{a \in C_1} b^L_a - \sum_{a \in C_2} b^L_a - d^L_{S\bar{S}} > 0$. In this work, we follow what was proposed by Nemhauser and Wolsey in [177]. As stated by the authors in section II.6.4, separating flow cover inequalities is equivalent to solving a series of equality knapsack problems, which are hard to solve. Hence, a heuristic solution may be preferable: it consists of solving (4.7), a relaxation of the separation problem, taking $\alpha \in \{0,1\}^{|(S,\bar{S})|}$ and $\beta \in \{0,1\}^{|(\bar{S},S)|}$ as the characteristic vectors of sets $C_1$ and $C_2$ respectively.

$$\begin{gathered} max \sum_{a \in (S,\bar{S})} (\hat{y}_a - 1)\alpha_a + \sum_{a \in (\bar{S},S)} \hat{y}_a \beta_a \\ \sum_{a \in (S,\bar{S})} b^L_a \alpha_a - \sum_{a \in (\bar{S},S)} b^L_a \beta_a > d^L_{S\bar{S}} \\ \alpha \in \{0,1\}^{|(S,\bar{S})|}, \beta \in \{0,1\}^{|(\bar{S},S)|} \end{gathered} \tag{4.7}$$

The set $L \subseteq K$ is defined as all commodities $\{k \in K : \hat{x}^k_a > 0, a \in (S, \bar{S})\}$, to increase the chances of finding a violated inequalities. Then, solving (4.7) is equivalent to put into $C_1$, all arcs $\{a \in (S, \bar{S}) : \hat{y}_a = 1\}$, and to sequentially insert into $C_2$, arcs $\{a \in (\bar{S}, S) : \hat{y}_a = 1\}$ in a nondecreasing order of $b^L_a$, while $\mu > 0$. Finally, $D_2 = \{a \in (\bar{S}, S) \setminus C_2 : b^L_a > \mu\}$, and if the obtained inequality is violated, the sequence independent lifting procedure proposed by Gu *et al.* [116] is applied, to produce a lifted flow cover inequality, according to the Theorem 12 of that same paper.

## 4.6 Flow pack separation method

As well as for flow cover inequalities, as discussed by Stallaert [203], the separation problem for flow pack inequalities (2.24) is equivalent to an equality knapsack problem. Therefore, a similar heuristic procedure as for flow covers is applied to separate flow packs $(C_1, C_2)$, $C_1 \subseteq (S, \bar{S})$, $C_2 \subseteq (\bar{S}, S)$ and $\mu < 0$. Such a heuristic consist in solving problem (4.7), however with constraint (4.8) insuring $\mu < 0$ instead of $\mu > 0$.

$$\sum\nolimits_{a \in (S,\bar{S})} b_a^L \alpha_a - \sum\nolimits_{a \in (\bar{S},S)} b_a^L \beta_a < d_{S\bar{S}}^L \tag{4.8}$$

In that manner, given the point $(\hat{x}, \hat{y})$, $C_2 = \{a \in (\bar{S}, S) : \hat{y}_a = 1\}$, and arcs $\{a \in (S, \bar{S}) : \hat{y}_a = 1\}$ are sequentially inserted into $C_1$, in a nondecreasing order of $b_a^L$, while $\mu < 0$. Also with the aim to increase the chances of finding violated inequalities, the set $L \subseteq K$ is defined as all commodities $\{k \in K : \hat{x}_a^k > 0, a \in (S, \bar{S})\}$. Finally, $D_1 = \{a \in (S, \bar{S}) \setminus C_1 : b_a^L > -\mu\}$.

If the obtained inequality is violated, the sequence-independent lifting procedure proposed by Atamtürk [13] is applied, to produce a lifted flow pack inequality. The procedure is based in a superadditive lifting function, which can also be used to obtain a lifted flow cover inequality, however weaker than the one proposed in [116].

## 4.7 Volume and Sensitivity Analysis for Relax-and-Cut scheme

In this section, we present the sensitivity analysis step of Algorithm 4, which aims to further explore the dual space of new dualized valid constraints, identified in Relax-and-Cut schemes. Indeed, the motivation for this new procedure comes from the fact that, even though some interesting cover and minimal cardinality inequalities were being identified, the benefits to the bound value were not being perceived.

### 4.7.1 Sensitivity analysis

The sensitivity analysis, in this case, is based on the fact that, given a primal solution for the Lagrangian subproblem, one has a range of values that the

Lagrangian multipliers may assume, so that the solution remains optimal. For example, let consider the integer problem $P : min\{f(x) \mid c(x) \geq 0,\ x \in X\}$, $X$ the set of integer solutions, $f(x)$ the objective function and $c(x) \geq 0$ a complicating constraint, which can be dualized, resulting in (4.9), for $\lambda \geq 0$. Moreover, for ease of simplicity, assume that the convex hull $co(X)$ has integrality property.

$$L(\lambda) = min_{x \in X}\ f(x) - \lambda c(x) \tag{4.9}$$

Figure 4.3 illustrates the set of all pairs

$$\Omega_X = \{(c(x), f(x)) \mid x \in X\}$$

supposed to be finite (given by the black points), and its convex hull $co(\Omega_X)$, shown with thick black lines. A similar geometric interpretation was proposed in section 5.3 of [41].

The feasible solutions of $P$ are inside the shaded region, satisfying $c(x) \geq 0$. The point named *opt* represents the optimal pair $(c(x^*), f(x^*))$ for problem $P$, and $lp^*$ is the optimal value for the linear relaxation of $P$, i.e. the best lower bound associated with the relaxation of the constraint $c(x) \geq 0$.

For a given multiplier $\lambda_1$, the dual function value $L(\lambda_1)$, obtained solving (4.9), corresponds to point A= $(c(x_{\lambda_1}), f(x_{\lambda_1}))$ since $c(x)$ has been relaxed, and defines a supporting hyperplane at A

$$H_1 = \{(\alpha, \beta) \in \mathbb{R}^2 \mid \beta - \lambda_1 \alpha = L(\lambda_1)\}$$

with a positive slope $\lambda_1$ ($H_1$ is orthogonal to vector $[-\lambda_1\ 1]^T$). The dual function value, $L(\lambda_1) \leq lp^*$ is obtained for $\alpha = 0$ at point $v_1$ in Figure 4.3.

Observe now that a greater scalar $\lambda_1^{'} > \lambda_1$ may increase the lower bound as long as the relaxed solution $x(\lambda_1^{'})$ does not move. Indeed, the orthogonal vector to hyperplane $H_1'$, i.e. $u = [-\lambda_1'\ 1]$ turns to the left when slightly increasing the multiplier from $\lambda_1$ to $\lambda_1'$ and the 'best' possible increase occurs when $H_1'$ supports the polyhedron on the whole segment [A B], B defining an alternative solution $x(\lambda_1')$, with the corresponding lower value at a higher value $v_1'$. Observe too, however, that a still higher slope for $\lambda_1'' > \lambda_1'$ can correspond to another solution at point C with a smaller dual value $v_1'' < v_1$.

On practical instances, careless modifications of Lagrangian multipliers are very likely to produce only worse bound values. Therefore, the sensitivity analysis comes as an alternative to modify the current Lagrangian multipliers, so that $L(\lambda)$ can be maximal for a given fixed primal point.

Figure 4.4 illustrates the same example from a dual point of view. It is well known that the dual function $L(\lambda)$ is concave and non-differentiable, and each piece corresponds to a primal point in Figure 4.3. In addition, $g_A$ and $g_B$ are the subgradients of $L$ associated with the supporting hyperplanes $H_A, H_B$ of the corresponding segments. As it can be seen in Figure 4.4, one can maximize $L(\lambda)$ restricted to segment $A$, moving from $\lambda_1$ to $\lambda_1'$, which is also in segment $B$. That new dual point is related to a non-differentiable point of the Lagrangian function $L(\lambda)$, and more than one subgradient can be derived from it, that is to say $g_A$, $g_B$ and all convex combinations of them.

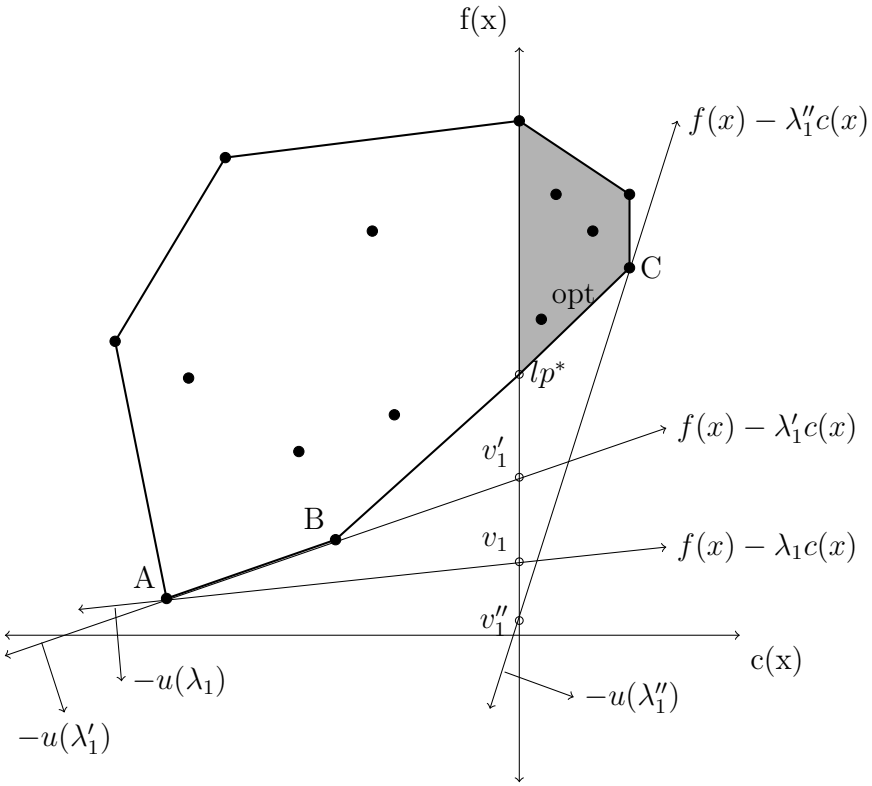



Figure 4.3: Primal view

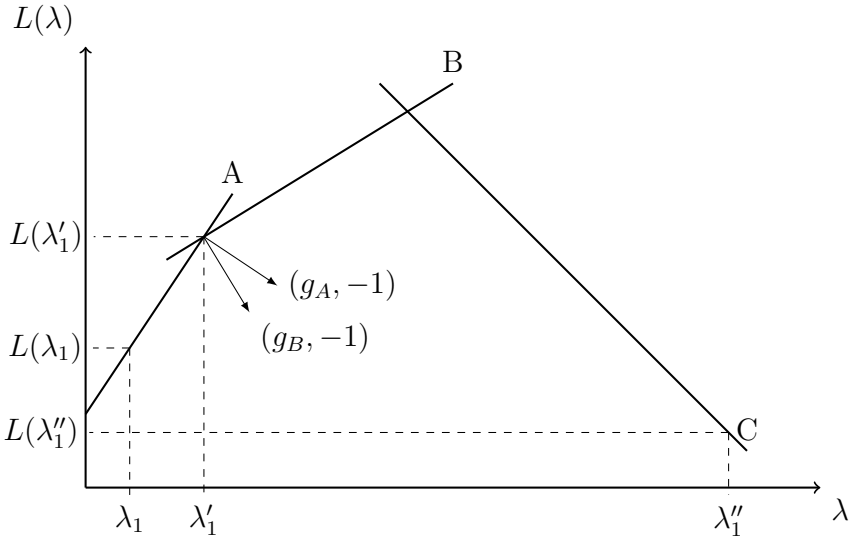



Figure 4.4: Dual view

Hence, the procedure can be applied in every iteration of the Volume Algorithm to reach possibly better bounds, as well as to identify possibly better subgradients for the next search direction. That may result in better lower bounds for the original problem and might accelerate the optimization process.

### 4.7.2 Implementation issues

Considering what has been said, we embed the sensitivity analysis into the Volume Algorithm (described by Algorithm 4). More precisely, at each iteration $t$, after solving the subproblem for the new trial point $\lambda^t$, we fix the solution $\hat{x}$ and recompute $L(\lambda)$, maximizing it over the $\lambda$-space, to obtain $\lambda^{t'}$ the redefined trial point.

In the case of the FCMC, the problem of $max_{\lambda \geq 0}\{L(\lambda) : x = x^t\}$ may be described by (4.10)-(4.20). The dual variables $\lambda_i^k, i \in N, k \in K$ correspond to the flow constraints (2.2), while dual variables $\mu_i, i \in CI$ correspond to inequalities in collection $CI$, the set of cover and minimal cardinality inequalities, with the general form: $\sum_{a \in C_i} \gamma_a y_a \geq b$.

Assuming $A^+$ the set of opened arcs, and $A^-$ the set of closed arcs, the reduced costs $\bar{f}_a$ (defined in (4.16)) of arcs $a \in A^+$ must be nonpositive, while for arcs $a \in A^-$, $\bar{f}_a$ must be nonnegative. The notation $CI(a)$ means the set of inequalities $i \in CI$ that include arc $a \in A$.

In terms of flow, for all arcs $a \in A$, consider $K^+(a)$ the set of commodities $k$ with positive flow in arc $a$, and $K^-(a)$ the set of commodities $k$ with no flow in that arc $a$. When $\sum_{k \in K^+(a)} q^k \geq w_a$, a critical commodity $\kappa(a) = \text{argmax}_{k \in K^+(a)}\{\bar{c}_a^k : \bar{c}_a^k \leq 0\}$ exists, otherwise $\kappa(a) = 0$. In the first case, the non-critical commodities

in $K^+(a)$ must present reduced costs smaller than or equal to $\bar{c}_a^{\kappa(a)}$, while commodities $k \in K^-(a)$ must present reduced costs greater than or equal to $\bar{c}_a^{\kappa(a)}$. Moreover, $\bar{c}_a^{\kappa(a)}$ must be nonpositive in case $\kappa(a) > 0$.

$$\text{Max} \sum_{a \in A^+} \bar{f}_a + \sum_{i \in CI} b\mu_i + \sum_{k \in K} q^k(\lambda^k_{D(k)} - \lambda^k_{O(k)}) \tag{4.10}$$

$$\bar{c}^k_{ij} = c^k_{ij} + \lambda^k_i - \lambda^k_j \qquad \forall\ (i,j) \in A,\ k \in K \tag{4.11}$$

$$\bar{c}^k_a \leq \bar{c}^{\kappa(a)}_a \qquad \forall\ a \in A,\ k \in K^+(a) \setminus \{\kappa(a)\} \tag{4.12}$$

$$\bar{c}^k_a \geq \bar{c}^{\kappa(a)}_a \qquad \forall\ a \in A,\ k \in K^-(a) \tag{4.13}$$

$$\bar{c}^{\kappa(a)}_a \leq 0 \qquad \forall\ a \in A,\ \kappa(a) > 0 \tag{4.14}$$

$$\bar{c}^{\kappa(a)}_a = 0 \qquad \forall\ a \in A,\ \kappa(a) = 0 \tag{4.15}$$

$$\bar{f}_a = (f_a + \sum_{k \in K^+(a)} \bar{c}^k_a \hat{x}^k_a - \sum_{i \in CI(a)} \gamma_a \mu_i) \qquad \forall\ a \in A \tag{4.16}$$

$$\bar{f}_a \leq 0 \qquad \forall\ a \in A^+ \tag{4.17}$$

$$\bar{f}_a \geq 0 \qquad \forall\ a \in A^- \tag{4.18}$$

$$\lambda^k_n \in \mathbb{R} \qquad \forall\ n \in N,\ k \in K \tag{4.19}$$

$$\mu_i \geq 0 \qquad \forall\ i \in CI \tag{4.20}$$

Remark that solving sensitivity analysis problem (4.10)-(4.20) may be, computationally, as expensive as solving the linear relaxation of the original problem, which is pointless to our purposes. Instead, one may solve it for the $CI$ dual variables exclusively, leaving $\lambda = \lambda^t$. In addition, since there is no need for an optimal solution, the sensitivity analysis problem might be solved heuristically, in order to further reduce computational times. For this purpose, we propose a heuristic focused on the $CI$ dual variables only.

As explained in subsection 4.7.1, it might be interesting to decrease potentially high-valued dual variables; therefore the heuristic tries to decrease some values of $\mu_i, i \in CI$, then increase others. Algorithm 8 describes how to decrease variable values, and Algorithm 9 how to increase them. Indeed, decreases are performed on dual values of satisfied inequalities, while increases are performed on dual values of violated inequalities.

Assume $CI^-$ to be the index set of satisfied inequalities, $CI^+$ the index set of violated inequalities, $AC^-$ the set of arcs present in some inequality of index $i \in CI^-$, and $AC^+$ the set of arcs present in some inequality of index $i \in CI^+$. Note that sets $CI^+$ and $CI^-$ are disjoint sets, so a decreased multiplier is not increased afterwards, and vice-versa. As for $CI(a)$, $CI^+(a)$ correspond to the

index sets of inequalities $i \in CI^+$ that include arc $a \in A$, being $CI^-(a)$ defined analogously.

First, Algorithm 8 is applied on the current dual vector $[\lambda,\ \mu]$, more specifically on the variables corresponding to satisfied inequalities. In this procedure, we aim to increase $\bar{f}_a \leq 0, a \in AC^-$, by decreasing values of $\mu_i, i \in CI^-$. That is because, for satisfied cover inequalities: $\mu_i(b - \sum_{a \in C_i} \gamma_a \hat{y}_a) \leq 0, i \in CI^-$, what deteriorates the objective function (4.10).

**Algorithm 8** Decrease cover multipliers

Let vector $[\lambda, \mu]$ be the current trial point.
Let $\bar{c}_a^k = c_a^k + \lambda_i^k - \lambda_j^k \ \forall\ a = (i,j) \in A,\ k \in K$
$\tilde{f}_a = f_a + \sum_{k \in K^+(a)} \bar{c}_a^k \hat{x}_a^k \ \forall\ a \in AC^-$
$\bar{f}_a \leftarrow (\tilde{f}_a - \sum_{i \in CI(a)} \gamma_a \mu_i) \ \forall\ a \in AC^-$
$AC_1^- = \{a : a \in AC^-; \bar{f}_a < 0; \tilde{f}_a > 0\}$
$AC_2^- = \{a : a \in AC^-; \bar{f}_a < 0; \tilde{f}_a \leq 0\}$
$m_a \leftarrow \sum_{i \in CI^-(a)} \mu_i \ \forall\ a \in AC^-$
**for all** $a \in AC_1^-$ **then for all** $a \in AC_2^-$ **do**
  **if** $a \in AC_1^-$ **then** $m \leftarrow \bar{f}_a$
  **else** $m \leftarrow \bar{f}_a - \tilde{f}_a$
  **for all** $i \in CI^-(a)$ **do**
    $\delta \leftarrow m(\mu_i/(m_a \gamma_a))$
    **if** $\mu_i + \delta < 0$ **then** $\delta \leftarrow \mu_i$
    $e = argmax_{e \in C_i}\{\bar{f}_e : \bar{f}_e < 0\}$
    **if** $\bar{f}_e - \gamma_e \delta > 0$ **then** $\delta \leftarrow \bar{f}_e/\gamma_e$
    $m_a \leftarrow m_a - \mu_i$
    $\mu_i \leftarrow \mu_i - \delta$
    Update $\bar{f}_e, \forall e \in C_i$
Return $[\lambda, \mu]$

Decreases must happen in such a way that $\bar{f}_a \leq 0, a \in AC^-$ do not become positive. Moreover, two situations may occur: either $f_a + \sum_{k \in K^+(a)} \bar{c}_a^k \hat{x}_a^k$ is by itself nonpositive, or the nonpositive value of $\bar{f}_a$ is due to the decrement of $CI$ dual values to that sum. In the first case, values will remain below or equal to zero, no matter how much one decreases $CI$ duals values, what contrasts with the second case, which needs more attention. Therefore, the algorithm starts iterating over arcs of the second case, passing to the other arcs in a second moment.

The decrement is initially set to a fraction of the current value of $\bar{f}_a$. If such a decrease produces an infeasible $\mu_i < 0$, the decrement is set to the current value (the maximal decrease). Afterwards, we check if such a decrement is possible for all

arcs present in the corresponding inequality, whose dual value is being decreased. In other words, we exclude the possibility of increasing too much the $\bar{f}_a$ value of some opened arc of that inequality, to the point it becomes positive. In that sense, it suffices to check for the maximal value $\bar{f}_a$ such that $\bar{f}_a < 0, a \in C_i$, if $\bar{f}_a - \gamma_a \delta > 0$, for which case the decrement $\delta$ is set to $\bar{f}_a/\gamma_a$.

Finally, Algorithm 9 is applied on the current dual vector $[\lambda,\ \mu]$, more specifically on the variables corresponding to violated $CI$ inequalities. In this procedure, we aim to increase values of $\mu_i, i \in CI^+$. That is because, for violated inequalities: $\mu_i(b - \sum_{a \in C_i} \gamma_a \hat{y}_a) > 0, i \in CI^+$, what raises the objective function (4.10).

In their turn, increases must happen in such a way that $\bar{f}_a \geq 0, a \in AC^+$ remain nonnegative, since higher values of $\mu_i, i \in CI^+$, mean reductions on $\bar{f}_a, a \in AC^+$. For this purpose, we compute $l_i$, the minimum value of $\bar{f}_a$ for each inequality of index $i \in CI^+$, that correspond to the maximum increase that $\mu_i$ can assume without resulting in a negative $\bar{f}_a, a \in C_i$.

The algorithm starts setting the increment values $\delta_i$ to their maximum: $\delta_i = l_i, \forall\ i \in CI^+$, then checks for each arc $a \in AC^+$ if such increment is possible. If not, it sets $\delta_i$ equal to a fraction of $\bar{f}_a$, trying to keep the same proportions between the $\mu_i, i \in CI^+(a)$, as it was originally. In the end, the increases to $\mu_i$ are made, and the new dual vector is entire returned.

**Algorithm 9** Increase cover multipliers

Let vector $[\lambda, \mu]$ be the modified trial point returned by Algorithm 8.
Let $\delta_i \leftarrow 0$ the value to be added to $\mu_i, i \in CI^+$
$\bar{f}_a \leftarrow (f_a + \sum_{k \in K^+(a)} \bar{c}_a^k \hat{x}_a^k - \sum_{i \in CI(a)} \gamma_a \mu_i)\ \forall\ a \in AC^+$
$m_a \leftarrow \sum_{i \in CI^+(a)} \mu_i\ \forall\ a \in AC^+$
$\delta_i \leftarrow l_i = min\{\bar{f}_a : a \in C_i, \bar{f}_a > 0\}\ \forall\ i \in CI^+$
**for all** $a \in AC^+ : \bar{f}_a > 0$ **do**
    $M \leftarrow \sum_{i \in CI^+(a)} \gamma_a \delta_i$
    **if** $\bar{f}_a - M < 0$ **then**
        **for all** $i \in CI^+(a)$ **do**
            $\delta_i \leftarrow min\{\delta_i, \bar{f}_a(\mu_i/(m_a \gamma_a))\}$
**for all** $i \in CI^+$ **do**
    $\mu_i \leftarrow \mu_i + \delta_i$
Update $\bar{f}_a, \forall a \in AC^+$
Return $[\lambda, \mu]' \leftarrow [\lambda, \mu]$

Observe that the changes in the $CI$ dual variables may produce reduced costs $\bar{f}_a = 0$ for some arcs $a \in A$, enabling those arcs to be opened or closed without changing the value of the Lagrangian function, what explains why one has optimal

solutions A and B, on Figure 4.3. Therefore, changing the current status of an arc with zero reduced cost implies making a movement from the optimal solution to another optimal solution. Moreover, in terms of the Volume Algorithm this movement is equivalent to the choice of which subgradient $g^t$ to take on the nondifferentiable point $[\lambda, \mu]'$.

That being said, to compute the subgradient on Algorithm 4, we analyse the impact of opening arcs with zero $\bar{f}_a$ over the factor $g^t h$, with the aim of obtaining $g^t h > 0$. In that way, the chances of performing green or yellow steps may augment. To estimate such impact, we compute the product $g_a^t h$, being $g_a^t$ the partial subgradient corresponding to the arc $a$, considering that the arc is open, otherwise one has a partial vector of zeros. Therefore, for each arc with zero reduced cost, if $g_a^t h > 0$ and $\tilde{y}_a \geq \epsilon_1$, the arc is opened, and if $g_a^t h \leq 0$ and $\tilde{y}_a \leq \epsilon_o$, the arc is closed.

Finally, the solution value is computed, as well as the whole subgradient, giving place for the usual operations of the Volume Algorithm. To conclude, remark that for Bundle methods, there is the possibility of adding more than one subgradient at a time to the bundle. In order to test that feature, we also embedded the sensitivity analysis into the Bundle method, and pass to the bundle both subgradients at the same iteration: the original one, not considering the factors $g_a^t h$, and the alternative one, obtained with the procedure described.

### 4.7.3 Constraint scaling

We noticed that one of the reasons why the addition of relaxed cover and minimal cardinality inequalities was not being beneficial to the bound quality is that the magnitude of values in the dimension of dual flow variables was much bigger than the magnitude of values in the dimension of dual cover and minimal cardinality variables. That is because, considering a subgradient vector, the entries related to flow constraints, in general, may admit values in a much larger interval, while the entries corresponding to the cover and minimal cardinality inequalities, might vary between 0 and values close to 1. Note that since such dual variables are nonnegative, negative values are not interesting. Such difference of magnitude can result in a stepsize that may be sufficient in the dimension of dual flow variables, but insignificant in the dimension of the dual cover and minimal cardinality variables.

In order to verify the validity of those remarks, we implemented the Volume-based Relax-and-Cut algorithm applied to the normalized formulation of the problem. In that way, flow variables are defined in the interval $[0, 1]$, while flow balance constraints are scaled down, resulting in (4.21). That results in a more balanced

subgradient, with all entries in the same magnitude of values.

$$\sum_{j \in N_i^+} x_{ij}^k - \sum_{j \in N_i^-} x_{ji}^k = \begin{cases} 1, & \text{if} \quad i = O(k) \\ -1, & \text{if} \quad i = D(k) \\ 0, & \text{otherwise} \end{cases} \quad \forall i \in N, k \in K \qquad (4.21)$$

### 4.7.4 Pertubation function

We now stablish a relation between the approaches presented in the last two sections, and the pertubation function presented by Geoffrion in [108, 109]. Still considering the example of section 4.7.1, recall that the dual function is, with appropriate changes of signs, the conjugate function of the useful perturbation function [192]

$$w(p) = \inf_{x \in X} \{f(x) \ : \ c(x) \geq p\}$$

which is a monotone increasing step function, in general nonconvex. $w(0) = f(x^*)$ and the dual approach tries to minimize the gap between $w(0)$ and $co[w(0)]$, the greatest convex function lower than $w$. Figure 4.5 shows the shape of the epigraph of the perturbation function for the same toy example as in section 4.7.1, with the convex envelope in dashed lines. The convex envelope is indeed defined by the same supporting hyperplanes as in Figure 4.3.

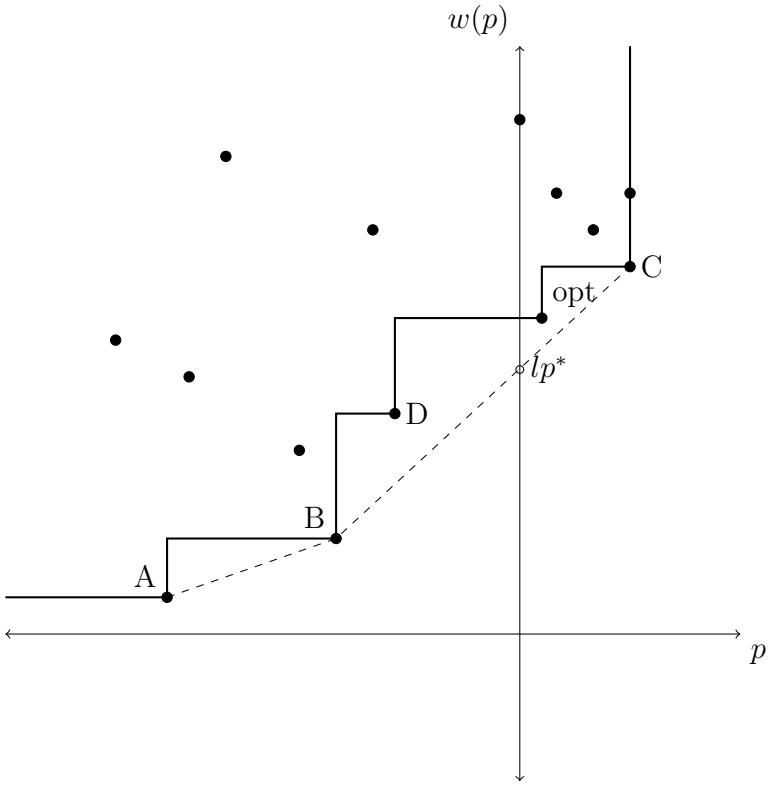
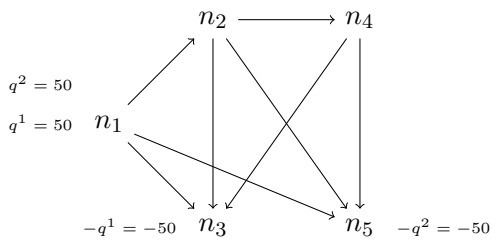



Figure 4.5: Perturbation function

To complete the present analysis, we present an example on a small FCMC instance, graphically shown in Figure 4.6. The instance is composed of 5 nodes, 8 arcs, and 2 commodities. The demands $q^k$ are given next to the origins, while their negative value is given next to their destinations. For every arc, the values of transportation cost, capacity, and fixed charge, in that order, are given on the right of the graph. Transportation costs do not vary on the same arc.




arc 1 : $(n_1,\ n_3)$, [1 50 20000]

arc 2 : $(n_1,\ n_5)$, [1 50 20000]

arc 3 : $(n_1,\ n_2)$, [1 100 15000]

arc 4 : $(n_2,\ n_3)$, [1 49 10000]

arc 5 : $(n_2,\ n_4)$, [1 98 5000]

arc 6 : $(n_2,\ n_5)$, [1 49 10000]

arc 7 : $(n_4,\ n_3)$, [1 49 5000]

arc 8 : $(n_4,\ n_5)$, [1 49 5000]

Figure 4.6: Small instance graph example

The solution value for that instance is 40100, while the linear relaxation gives a bound of value 30706, approximately. The cover inequality (4.22) is violated by

the linear relaxed solution, and can be added to the formulation, which results in a better bound of value approximately 33057.

$$y_1 + y_2 + y_4 + y_6 \geq 1 \tag{4.22}$$

We define $X$, the set of points restricted to the flow conservation constraints, and $Y$, the set of points restricted to the binary domain, and capacity and forcing constraints. Considering that, for all $i \in N$ and $k \in K$, $q_i^k = q^k$ if $i$ is the origin of commodity $k$, $q_i^k = -q^k$ if $i$ is the destination of commodity $k$, and $q_i^k = 0$ otherwise, the sets $X$ and $Y$ are formally defined as follows:

$$X = \{x \in \mathbb{R}_+^{|A||K|} : \sum\nolimits_{j \in N_i^+} x_{ij}^k - \sum\nolimits_{j \in N_i^-} x_{ji}^k = q_i^k, \forall i \in N, k \in K\}$$
$$Y = \{x \in \mathbb{R}_+^{|A||K|}, y \in \mathbb{B}^{|A|} : \sum\nolimits_{k \in K} x_a^k \leq w_a y_a;\ x_a^k \leq b_a^k y_a, \forall a \in A, k \in K\}$$

Then, for $C = \{1, 2, 4, 6\}$, the pertubation function can be defined as (4.23). Finally, for different values of $p$, Figure 4.7 illustrates the function $w(p)$, given by the black staircase line, and its convex envelope $co[w(p)]$, given by the densely dotted black line. Remind that the supporting hyperplans of $co[w(p)]$ defined by $[-\mu\ 1]^T$, give, at $p = 0$, $-w^*(\mu)$ the negative of the conjugate function, which is indeed equal to the Lagrangian function at point $\mu$. In this case, note that $\mu = 0$ provide the best value, while it becomes deteriorated for higher values of $\mu$.

$$w(p) = \inf\nolimits_{(x,y) \in X \cap Y} \{f(x, y)\ : \sum_{c \in C} y_c - 1 \geq p\} \tag{4.23}$$

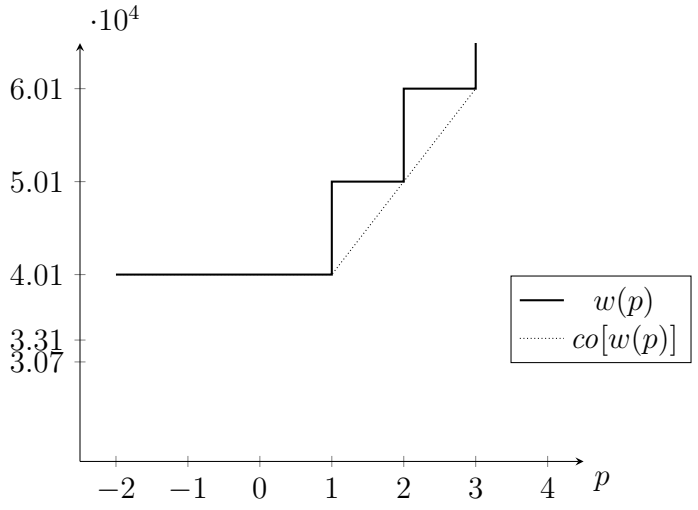



Figure 4.7: Perturbation function for the FCMC example

## 4.8 Computational experiments

For the computational experiments, the instances considered were the benchmark groups Canad-C, Canad-R, and Canad-N. Computational times are given in seconds, and Cplex 12.8 was used to solve the LPs, when necessary. To avoid adding inequalities that will rapidly become satisfied for the rest of the optimization process, no addition is performed before 100 iterations. Furthermore, added inequalities not violated after 10 consecutive iterations are removed.

The tests were run in a 60Gb RAM, Intel(R) Xeon(R) CPU E5-2670, 2.60 GHz Linux computer. The parameters of the Volume Algorithm and the Bundle Method were set accordingly to the calibration made in section 3.6.2. A time limit of 3600 seconds and a limit of 1000 iterations were set.

This section starts with the results obtained with the performances of the standard Relax-and-Cut algorithm, followed by the results with the sensitivity analysis and constraint scaling. At the end of the section, we compare the cutset generation procedure, discussed in section 4.3, to the one proposed by Chouman *et al.* [49].

Before discussing the results, the notation used in the remainder is presented: considering $z_l$ the bound given by the Lagrangian knapsack relaxation, solved by the regular Volume Algorithm, and $\bar{z}$ the bound provided by the respective Relax-and-Cut method, the deviation 'Dev' measures the benefit of the given procedure, relative to $z_l$:

$$\text{Dev} = (\bar{z} - z_l)/z_l * 100 \tag{4.24}$$

Therefore, in every Table, columns 'Max Dev %' present the maximum deviation obtained for that related group of instances, while columns 'Avg Dev %' give the average deviation for that group. Furthermore, the average computational times are given in columns 't', measured in seconds, and columns 'Num' present the average number of cuts added during the optimization processes.

### 4.8.1 Cutset-based inequalities

The first set of tests aim to verify the benefits of each type of cutting inequalities considered. The results are shown in Tables 4.1, aggregated by group of instances: Table 4.1a presents the average results for group N, Table 4.1b for group C, and Table 4.1c for group R. The first 4 lines of each table give the results for each type of cut, being added solely, while lines 'All' give the results when all types of cuts are considered, simultaneously.

(a) Benchmark Instances N

| | Short Term Violation | | | | Long Term Violation | | | |
|---|---|---|---|---|---|---|---|---|
| | Max Dev % | Avg Dev % | Num | t | Max Dev % | Avg Dev % | Num | t |
| Cover | 0.01 | 0.00 | 1141 | 16.6 | 0.01 | 0.00 | 3 | 16.8 |
| Min Card | 0.02 | 0.00 | 1178 | 18.4 | 0.01 | 0.00 | 3 | 17.0 |
| Flow Cover | 0.00 | 0.00 | 4267 | 21.3 | 0.01 | 0.00 | 0 | 19.0 |
| Flow Pack | 0.03 | 0.00 | 3995 | 21.6 | 0.04 | 0.00 | 1 | 19.2 |
| All | 0.01 | 0.00 | 8458 | 40.5 | 0.02 | 0.00 | 8 | 22.5 |

(b) Benchmark Instances C

| | Short Term Violation | | | | Long Term Violation | | | |
|---|---|---|---|---|---|---|---|---|
| | Max Dev % | Avg Dev % | Num | t | Max Dev % | Avg Dev % | Num | t |
| Cover | 8.55 | 0.48 | 567 | 6.6 | 8.99 | 0.53 | 20 | 6.6 |
| Min Card | 9.50 | 0.51 | 702 | 6.6 | 8.68 | 0.46 | 19 | 6.5 |
| Flow Cover | 1.72 | 0.10 | 1623 | 7.9 | 2.55 | 0.18 | 12 | 7.0 |
| Flow Pack | 4.44 | 0.35 | 1207 | 7.5 | 5.67 | 0.45 | 10 | 7.0 |
| All | 7.71 | 0.46 | 3247 | 12.9 | 9.46 | 0.78 | 54 | 7.6 |

(c) Benchmark Instances R

| | Short Term Violation | | | | Long Term Violation | | | |
|---|---|---|---|---|---|---|---|---|
| | Max Dev % | Avg Dev % | Num | t | Max Dev % | Avg Dev % | Num | t |
| Cover | 0.25 | 0.02 | 391 | 0.9 | 0.28 | 0.02 | 10 | 0.9 |
| Min Card | 0.25 | 0.02 | 465 | 0.9 | 0.28 | 0.02 | 11 | 0.9 |
| Flow Cover | 1.43 | 0.07 | 923 | 1.2 | 1.62 | 0.07 | 2 | 1.0 |
| Flow Pack | 4.13 | 0.16 | 770 | 1.2 | 4.44 | 0.18 | 3 | 1.0 |
| All | 3.35 | 0.14 | 1853 | 3.2 | 4.91 | 0.19 | 27 | 1.3 |

Table 4.1: Efficiency of cuts in the Relax-and-Cut algorithm

Columns under 'Short Term Violation' label present the results considering that, to be added, inequalities have to violate only the current Lagrangian subproblem solution, while 'Long Term Violation' label makes reference to the fact that inequalities may also violate the historical linear solution (vector $\tilde{x}$ in Algorithm 4) provided by Volume. As one can notice in the tables, the number of added inequalities is drastically reduced, without deteriorating performances in terms of quality of bound, with 'Long Term Violation'. Especially when all cut types are allowed, the gains in computational times, with a reduced number of added cuts

is clearly perceived. Nevertheless, despite the significant quantity of cuts inserted with 'Short Term Violation', the computational times remain competitive, thanks to the removing procedure of non-violated cuts.

Still regarding 'Short Term Violation', one can deduce that most of the identified violated cuts were not improving with respect to the bound value, since the results obtained with that approach were not better than the ones obtained with 'Long Term Violation', sometimes even worse. Indeed, the addition of irrelevant cuts may disturb the search direction and stepsize, resulting in bounds a bit worse than the one expected. In that manner, only the 'Long Term Violation' approach is considered for the rest of the computational experiments.

With respect to each type of cut independently, the flow pack inequalities provided the best results for instances in groups R and N, while for group C, cover and minimal cardinality inequalities were more efficient. At first glance, the best overall performances were obtained with the 'All' configuration. However, the same cannot be affirmed with the consideration of sensitivity analysis and constraint scaling. Unfortunately, even though better results were obtained when those two additional features are considered, the average bound improvements were not significant for the majority of the instances, presenting small increments on average.

### 4.8.2 Sensitivity analysis

We now present the results obtained when sensitivity analysis is applied in Algorithm 4, i.e. considering *flag=true*. The results, also aggregated by group of instances, as in precedent tables, are presented in Table 4.2. Since the 'All' configuration provided the best overall results in the previous section, its performances are compared to the cover and minimal cardinality configurations. A third case is considered, allowing the addition of both cover and minimal cardinality cuts (lines 'Cover & Min Card').

As one can observe, the sensitivity analysis improved the performances of the Volume-based Relax-and-Cut algorithm for all groups of benchmark instances, providing better of equal quality solutions on average. Especially for group R, that gain was greater than 6% considering the maximum deviation in that group. For group C, average results reached 1% of bound improvement, while it could reach no better than 0.8%, before.

As mentioned, the benefits of considering all types of cuts are no longer evident when sensitivity analysis is applied. In this case, it may be preferable to consider only cover and minimal cardinality cuts, noting that for group C, the 'Cover & Min Card' configuration performed better, while for group R the 'Cover' configuration

presented better results on average.

(a) Benchmark Instances N

| | flag = false | | | flag = true | | |
|---|---|---|---|---|---|---|
| | Max Dev % | Avg Dev % | t | Max Dev % | Avg Dev % | t |
| Cover | 0.01 | 0.00 | 16.8 | 0.42 | 0.05 | 21.1 |
| Min Card | 0.01 | 0.00 | 17.0 | 0.42 | 0.05 | 21.6 |
| All | 0.02 | 0.00 | 22.5 | 0.42 | 0.05 | 28.9 |
| Cover & Min Card | 0.01 | 0.00 | 21.7 | 0.42 | 0.05 | 22.3 |

(b) Benchmark Instances C

| | flag = false | | | flag = true | | |
|---|---|---|---|---|---|---|
| | Max Dev % | Avg Dev % | t | Max Dev % | Avg Dev % | t |
| Cover | 8.99 | 0.53 | 6.6 | 10.82 | 0.99 | 7.5 |
| Min Card | 8.68 | 0.46 | 6.5 | 10.68 | 0.93 | 7.4 |
| All | 9.46 | 0.78 | 7.6 | 10.38 | 0.97 | 8.9 |
| Cover & Min Card | 10.31 | 0.63 | 7.6 | 11.18 | 1.01 | 7.7 |

(c) Benchmark Instances R

| | flag = false | | | flag = true | | |
|---|---|---|---|---|---|---|
| | Max Dev % | Avg Dev % | t | Max Dev % | Avg Dev % | t |
| Cover | 0.28 | 0.02 | 0.9 | 6.85 | 0.66 | 0.7 |
| Min Card | 0.28 | 0.02 | 0.9 | 6.34 | 0.58 | 0.7 |
| All | 4.91 | 0.19 | 1.3 | 6.89 | 0.65 | 0.8 |
| Cover & Min Card | 0.59 | 0.03 | 1.1 | 6.60 | 0.65 | 0.7 |

Table 4.2: Sensitivity Analysis Results

In a second moment, the performances of sensitivity analysis are measured considering a predefined set of cover inequalities to be added in a Lagrangian fashion to the strong formulation, before start solving the Lagrangian dual. Here, the goal is to observe if the good results obtained previously are related only to the strength of the cuts inserted, or if the sensitivity analysis may really boost the Lagrangian optimization process, given that the set of *on-the-fly* added cuts is probably different, when running the Relax-and-Cut with and without the additional feature.

We also test the effect of solving the problem (4.10)-(4.20) to optimality, re-

stricted to the space of added dual variables, leaving the flow dual variables fixed to their current values. Moreover, the optimal solution value for the strong linear relaxation, including the set of additional cuts, is also computed, with results shown in Table 4.3b, serving as reference, since the Lagrangian subproblem presents the integrality property. In that sense, to avoid having to deal with too big linear programs, only cover inequalities are considered in this second test set.

The results are presented in Table 4.3, aggregated by groups of instances arranged in lines. In 4.3a, the last three columns under 'flag=true, exact' present the results obtained with the exact approach of sensitivity analysis, while the three columns in the middle correspond to the heuristic approach, and the first three ones to the Relax-and-Cut algorithm without such a feature.

(a) Relax-and-Cut

| | flag=false | | | flag=true | | | flag=true, exact | | |
|---|---|---|---|---|---|---|---|---|---|
| | Max Dev % | Avg Dev % | t | Max Dev % | Avg Dev % | t | Max Dev % | Avg Dev % | t |
| N | 0.02 | 0.01 | 7.9 | 0.39 | 0.05 | 7.9 | 0.41 | 0.05 | 9.3 |
| C | 7.85 | 0.28 | 1.6 | 11.24 | 1.02 | 1.6 | 6.12 | 0.45 | 2.8 |
| R | 0.05 | 0.01 | 0.9 | 5.20 | 0.01 | 0.9 | 5.12 | 0.01 | 1.7 |

(b) LP Bound

| | Max Dev % | Avg Dev % |
|---|---|---|
| N | 0.51 | 0.13 |
| C | 12.17 | 1.20 |
| R | 5.23 | 0.78 |

Table 4.3: Sensitivity Analysis with Exact Approach

As one can notice, the results show that, indeed, the procedure of sensitivity analysis significantly improved the performances of the regular Relax-and-Cut algorithm, also providing better bounds for all groups of instances in this second set of tests. Remark that the heuristic procedure provided bounds less than 1% lower than the bound obtained solving the strengthened linear relaxation (Table 4.3b). In its turn, for the standard procedure, a margin of more than 4%, when compared the LP value, is observed.

Furthermore, the exact approach did not present an expressive improvement with respect to the heuristic one, performing better only for group N. In fact, the proposed heuristic tries to get a modified trial point that is closed to the

current search direction, while the exact approach, do not take such an aspect into account, which may cause a convergence issue.

In terms of computational times, the heuristic sensitivity analysis did not cause an increase in computational costs, while solving an LP at each iteration in the exact approach, may be costly in terms of time consumption, according to the results in Table 4.3a. Moreover, note that these second computational tests correspond to one iteration of a Delayed Relax-and-Cut procedure [159]. That is to say, cuts were identified after a run of the regular Volume Algorithm, and a second run took place with the set of identified cuts added. The dual solution of the first run was used as hotstart to the second one. Therefore, it can be concluded that the proposed feature of sensitivity analysis may improve Relax-and-Cut scheme performances, both in Delayed and Non-Delayed versions, without increasing computational times too much.

### Bundle Method

Finally, remind that more than one subgradient can be included at a time into the bundle of Bundle Methods. Therefore, two versions of the Bundle-based Relax-and-Cut procedure using heuristic sensitivity analysis were implemented. The first one computes only one subgradient, following the same procedure described in section 4.7.2. The second one computes two subgradients: one corresponding to the trial dual point before sensitivity analysis, and another one after sensitivity analysis, computed like in the first version.

The results are presented in Table 4.4, aggregated by groups of instances arranged in lines, considering only cover inequalities. The last three columns under 'flag=true, 2 items' present the results obtained with the addition of 2 items per iteration, while the three columns in the middle correspond to the unique item approach, and the first three ones to the version without sensitivity analysis.

| | flag=false | | | flag=true | | | flag=true, 2 items | | |
|---|---|---|---|---|---|---|---|---|---|
| | Max Dev % | Avg Dev % | t | Max Dev % | Avg Dev % | t | Max Dev % | Avg Dev % | t |
| N | 0.05 | 0.00 | 33.0 | 0.39 | 0.06 | 29.4 | 0.20 | 0.03 | 32.2 |
| C | 6.15 | 0.30 | 8.9 | 10.85 | 0.97 | 8.2 | 9.93 | 0.82 | 8.9 |
| R | 3.19 | 0.10 | 1.0 | 5.96 | 0.53 | 1.1 | 6.59 | 0.41 | 1.2 |

Table 4.4: The Bundle Method and Sensitivity Analysis

According to Table 4.4, the use of sensitivity analysis is also beneficial for the

Bundle-based Relax-and-Cut scheme, providing better solutions for every instance group, when compared to the regular version. Moreover, it can be seen that the possibility of adding more than one subgradient at a time is not necessarily an advantage for the optimization process. Except for some instances of group R, the single-item approach provided better bounds within less time, also giving the best average results overall.

Comparing the Volume and Bundle versions ('flag=true' columns of Table 4.4 and 'Cover' lines of Table 4.2), Volume provided the best performances in general, with respect to time consumption. In terms of quality of bound, the Bundle version managed to be slightly better for group N, on average. However, for groups C and R the Volume version presented the best improvements on average, especially for group R, where the difference was bigger.

### 4.8.3 Constraint scaling

We now present the results obtained with the constraint scaling, which considers the 'normalized' formulation of the FCMC problem. The sensitivity analysis was also tested in this case. The results are shown in Tables 4.5 and 4.6. Table 4.5 shows results aggregated by group of instances, as in precedent tables. Like in Table 4.1, the first 4 lines of each subtable give the results for each type of cut, being added solely, while lines 'All' give the values when all types of cuts are considered.

(a) Benchmark Instances N

| | Max Dev % | Avg Dev % | t |
|---|---|---|---|
| Cover | 0.43 | 0.06 | 21.1 |
| Min Card | 0.44 | 0.07 | 21.4 |
| Flow Cover | 0.05 | 0.01 | 24.0 |
| Flow Pack | 0.05 | 0.01 | 23.7 |
| All | 0.43 | 0.06 | 28.1 |

(b) Benchmark Instances C

| | Max Dev % | Avg Dev % | t |
|---|---|---|---|
| Cover | 11.75 | 1.13 | 7.4 |
| Min Card | 11.69 | 1.10 | 7.6 |
| Flow Cover | 1.19 | 0.06 | 8.0 |
| Flow Pack | 2.06 | 0.07 | 8.1 |
| All | 7.35 | 0.44 | 8.7 |

(c) Benchmark Instances R

| | Max Dev % | Avg Dev % | t |
|---|---|---|---|
| Cover | 7.03 | 0.73 | 1.0 |
| Min Card | 7.02 | 0.69 | 1.0 |
| Flow Cover | 0.66 | 0.03 | 1.2 |
| Flow Pack | 0.66 | 0.02 | 1.2 |
| All | 4.30 | 0.37 | 1.5 |

Table 4.5: Constraint Scaling Results

As can be observed in the first table, the 'Cover' and 'Min Card' configurations provided the best performances in both aspects: time consumption and quality of bound. Like in section 4.8.2, no benefits in including flow pack and flow cover inequalities were perceived. Moreover, comparing Tables 4.2 and 4.5, constraint scaling turned out to be the best alternative to the Relax-and-Cut scheme, providing notably better results than the sensitivity analysis alone.

Still, it can be observed in Tables 4.5 and 4.1 ('Long Term Violation' columns) that the 'Flow Cover' and 'Flow Pack' configurations were sometimes worse, when constraint scaling is considered. Mainly for groups C and R, the losses in bound improvement attained almost 3.8%, indicating that constraint scaling may be useful for the flow pack and flow cover inequalities too, although such a procedure seems not to be as simple as for flow balance constraints. In addition, a sensitivity analysis can be attempted in the dimension of the flow cuts dual variables, for

which case a more complex procedure is needed too. We leave these considerations for further research.

(a) Benchmark Instances N

| | flag = false | | | flag = true | | |
|---|---|---|---|---|---|---|
| | Max Dev % | Avg Dev % | t | Max Dev % | Avg Dev % | t |
| Cover | 0.43 | 0.06 | 21.1 | 0.43 | 0.07 | 21.1 |
| Min Card | 0.44 | 0.07 | 21.4 | 0.44 | 0.07 | 21.6 |
| Cover & Min Card | 0.43 | 0.07 | 21.9 | 0.44 | 0.07 | 21.9 |

(b) Benchmark Instances C

| | flag = false | | | flag = true | | |
|---|---|---|---|---|---|---|
| | Max Dev % | Avg Dev % | t | Max Dev % | Avg Dev % | t |
| Cover | 11.75 | 1.13 | 7.4 | 11.79 | 1.15 | 7.6 |
| Min Card | 11.69 | 1.10 | 7.6 | 11.93 | 1.15 | 7.5 |
| Cover & Min Card | 11.39 | 0.83 | 7.6 | 11.65 | 1.09 | 7.5 |

(c) Benchmark Instances R

| | flag = false | | | flag = true | | |
|---|---|---|---|---|---|---|
| | Max Dev % | Avg Dev % | t | Max Dev % | Avg Dev % | t |
| Cover | 7.03 | 0.73 | 1.0 | 7.04 | 0.74 | 1.0 |
| Min Card | 7.02 | 0.69 | 1.0 | 7.04 | 0.72 | 1.1 |
| Cover & MinCard | 6.05 | 0.68 | 1.1 | 6.76 | 0.70 | 1.1 |

Table 4.6: Constraint Scaling with Sensitivity Analysis Results

Finally, Table 4.6 presents the results obtained combining constraint scaling and sensitivity analysis, with values arranged such as in Table 4.2. Only cover and minimal cardinality inequalities are considered, since they provided the best results in the previous table.

According to Table 4.6, the use of sensitivity analysis is always beneficial in general. However, the gains in bound quality are not as high as it was observed in Table 4.2. Furthermore, for this combined approach, observe that the 'Cover' configuration performed generally better, except for the group N, for which 'Min Card' configuration presented slightly better results.

### 4.8.4 Cutset generation

In order to test the efficiency of the heuristic proposed to define cutsets, we solve, for each instance, the weak relaxation of the present problem, including cover inequalities obtained with the separated cutsets. We measure the proportional increase $\Delta z_w$ to the weak relaxation bound $z_w$, due to the addition of the generated cover inequalities. Therefore, the value of $\bar{z}$ in equation (4.25) is given by the weak linear relaxation with the cover inequalities added. We take the cover inequalities as example, to measure performances.

$$\Delta z_w = (\bar{z} - z_w)/z_w * 100 \tag{4.25}$$

The Table 4.7 show the results. The cutsets and covers were generated during two runs of the Relax-and-Cut procedure, with 1000 iterations each. The first run used the standard algorithm, and the second one embedded sensitivity analysis. Then, the collection of generated covers inequalities was added to the weak relaxation, and the LP was solved with Cplex.

| | $\Delta z_w$ % | Metaheuristic % |
|---|---|---|
| N | 0,7 | - |
| C | 11,8 | - |
| R | 10,2 | - |
| (C, R) | 11,0 | 8,54 |

Table 4.7: Cutset generation for benchmark Instances C, N and R

We compare our results to the ones obtained by Chouman *et al.* [49], on the same set of instances (C and R). The authors proposed a metaheuristic-based procedure for cutset generation. The column 'Metaheuristic' shows the average value of $\Delta z_w$ provided in [49], stressing that only the overall average result was provided. As one can notice, the procedure proposed here produced interesting cutsets, that enabled the identification of better cover inequalities than the metaheuristic-based one.

## 4.9 Conclusion

In this chapter we presented a Volume-based Relax-and-Cut scheme, using cutset-based inequalities as cuts to be added in a 'Non-Delayed' fashion to the

Lagrangian relaxation of the FCMC problem. Unfortunately, the improvements on the lower bound maximal values were not much bigger than 1% on average, but for some instances a gain of 11% could be achieved using additional features to enhance performances: Constraint Scaling and Sensitivity Analysis. In addition, the best results were obtained with cover and minimal cardinality inequalities.

Considering the best Relax-and-Cut configuration, the average results for very large scale instances (groups A to H of section 3.6.1) are shown in the appendix C. The same conclusions can be made for those instances, such that lower bounds increases remained under 1%, with the exception of instances in group B, for which improvements attained 1.25%.

Moreover, a new procedure to separate cutsets was proposed, proving to be competitive to another successful approach in the literature. Remark that the presented procedure can be used in the LP context too. Since, with available dual values, it suffices to solve the Lagrangian subproblem to get an integer vector used in Algorithm 5.

Future research may aim at proposing different cuts that are able to further improve bounds, and the implementation of sensitivity analysis in other contexts, for which constraint scaling is not trivial. In the next chapter, the present Relax-and-Cut algorithm is embedded in a Branch-and-Cut scheme. It is used as a solver to each node LP of the search tree.

# Chapter 5

# Volume-Based Branch-and-Cut

As a final contribution of the present thesis, in this chapter, we present the Branch-and-Cut algorithm, which embeds the Relax-and-Cut procedure discussed in the previous chapter. Moreover, a Lagrangian Feasibility Pump heuristic is used to accelerate finding better upper bounds. A Branch-and-Cut heuristic version is also tested. The results were promising for both exact and heuristic approaches.

## 5.1 Introduction

As mentioned in section 2.3.1, the Branch-and-Cut algorithm is based on Branch-and-Bound schemes, where cutting planes are added to the formulation, at each search tree node, in order to strengthen the linear relaxation of the problem. In that sense, the Relax-and-Cut method discussed in the previous chapter is now embedded in the search tree to help solving the linear restriction of FCMC at each node. Moreover, we consider that a good initial feasible solution is available, which is a completely plausible statement, given the good efficiency of the heuristics present in the literature (see 2.3.3). The same is considered by [50], while developing an LP-based Branch-and-Cut algorithm for the FCMC. In section 5.8, we compare our results to their algorithm.

The best Relax-and-Cut implementation, according to the results presented in subsection 4.8, was considered, inclusive the parameters setting, the stopping criteria and the constraint scaling features (see 4.7.3), as well as the cover inequalities being included in a non-delayed fashion. Minimal cardinality inequalities and the sensitivity analysis are not considered since their inclusion in the Relax-and-Cut algorithm did not improve performances significantly, which may lead to losses in time-consuming aspects.

For the remainder of this section, let consider $(A_1, A_0, A_*)$ an arbitrary node, where $A_1$ is the set of 1-fixed arcs $\{a \in A : y_a = 1\}$, $A_0$ is the set of 0-fixed arcs

$\{a \in A : y_a = 0\}$, and $A_*$ is the set of nonfixed arcs $A \setminus \{A_1 \cup A_0\}$. Note that $A_1 \cap A_0 \cap A_* = \emptyset$ and $A_1 \cup A_0 \cup A_* = A$. Moreover, let consider $\bar{Z}$ the value of the best known feasible solution, $z_L$ the Lagrangian final objective value of a node, and $\bar{f}_a, a \in A_*$, the reduced costs of the arcs, given the final dual solution on that node. As in the previous chapter, here we keep notating $(\tilde{x}, \tilde{y})$: the linear relaxation approximate vector provided by the Volume Algorithm, and $(\hat{x}, \hat{y})$: the solution for the current Lagrangian subproblem.

The next sections discuss implementation aspects of the Branch-and-Cut algorithm, such as local and global cuts (section 5.2), variable fixing (section 5.3), branching, and pruning (sections 5.4 and 5.5 respectively). Most of the implemented features are based on strategies successfully implemented in previous works in the literature, mainly in [197, 27, 150, 50, 128]. However, a specially tailored Volume-based Branch-and-Cut algorithm for the FCMC is a new contribution. Moreover, a Feasibility Pump heuristic [80] for the FCMC is presented, using Volume as a solver for LPs. In fact, a Branch-and-Cut algorithm with heuristic variable fixing, based on Feasibility Pump is also tested. Upper bounds and the heuristic procedure are presented in section 5.6. At the end of this chapter, the computational experiments are reported in section 5.8, followed by a conclusion in section 5.9.

## 5.2 Local and global cuts

The terms *local* and *global*, in the present context, make reference to the validity of an added cut w.r.t the nodes of the search tree. Local cuts are only valid for some node and its descendants (the subtree), while the global ones are valid in the whole tree.

In that sense, cover inequalities generated in the root node may be considered as global cuts, while for a given node $(A_1, A_0, A_*)$, only variables in $A_*$ will be used to generate covers inequalities, thus they must be considered as local cuts. Remark that a local cut can be made globally valid by a lifting procedure (see 4.4.3). According to Mitchell [173], that may represent a gain in terms of memory storage requirements, since one can avoid storing the same inequality in different subtrees. However, remind that a binary knapsack problem must be solved for each lifting variable, which may represent a loss in time performances. Given the large size of instances considered, we choose not to apply such a procedure, giving preference to a faster algorithm.

In addition to cover inequalities, in the next subsections, we present other cuts considered to be added, taking advantage of the problem structure and the

information provided by the search tree. That is a very advantageous strategy for accelerating the convergence of the Branch-and-Cut algorithm, being widely applied in the literature as one can see in surveys [170, 173, 73].

### 5.2.1 Local Lagrangian cardinality cuts

The second type of local cut considered is related to inequalities presented by Sellman *et al.* [197], and the reduced cost variable fixing procedure discussed in subsection 5.3.1. It comes from the fact that, given a feasible solution, one can set bounds on the number of arcs in a defined set, that has to be present in any improving solution.

According to Kliewer and Timajev [150], at a given node, inequality (5.1) is locally valid, for $T_+ \subseteq \{a \in A_* : \bar{f}_a > 0\}$, such that $z_L + \bar{f}_a + \bar{f}_b \geq \bar{Z},\ \forall\ a, b \in T_+$, i.e. the set of nonfixed arcs, such that every pair of arcs in the set satisfies that inequality. That means that feasible solutions in the subtree of the current node must present at most one opened arc from $T_+$, so it can be an improving solution over $\bar{Z}$. Similarly, inequality (5.2) is locally valid, for $T_- \subseteq \{a \in A_* : \bar{f}_a < 0\}$, such that $z_L - \bar{f}_a - \bar{f}_b \geq \bar{Z},\ \forall\ a, b \in T_-$, which means that any improving solution existing in that subtree must have at most one arc in $T_-$ closed.

$$\sum_{a \in T_+} y_a \leq 1 \tag{5.1}$$

$$\sum_{a \in T_-} y_a \geq |T_-| - 1 \tag{5.2}$$

### 5.2.2 Global feasibility cuts

Remark that any feasible solution in a certain subtree $(A_1, A_0, A_*)$ must satisfy the multicommodity flow system (5.3)-(5.5), where node sets $N_i^+$ and $N_i^-$, $i \in N$, are defined for the arc set $A \setminus A_0$ as in section 2.1.

$$\sum_{j \in N_i^+} x_{ij}^k - \sum_{j \in N_i^-} x_{ji}^k = \begin{cases} q^k, \text{ if } i = O(k) \\ -q^k, \text{ if } i = D(k) \\ 0, \text{ otherwise} \end{cases} \quad \forall i \in N, k \in K \tag{5.3}$$

$$\sum_{k \in K} x_a^k \leq w_a \qquad \forall a \in A \setminus A_0 \tag{5.4}$$

$$x_a^k \geq 0 \qquad \forall a \in A \setminus A_0, k \in K \tag{5.5}$$

Hence, the global feasibility cuts come from the fact that, if the multicommodity flow system is infeasible for the arc set $A \setminus A_0$, then inequality (5.6) is valid not only in that subtree but for the whole problem.

$$\sum_{a \in A_0} y_a \geq 1 \tag{5.6}$$

Since only closed arcs can cause infeasibility, the global feasibility cut prevents an infeasible topology to be generated again. Moreover, such inequality was implemented by Chouman *et al.* [50] as a filtering strategy for the same problem, obtaining good results.

In the Branch-and-Cut algorithm, the cuts (5.6) are generated during the heuristic procedure, or when node infeasibility is detected either by the Volume Algorithm, or by solving the multicommodity flow system exactly. Indeed, the Lagrangian dual bound tends to infinity when the node is infeasible, but in practice, this is not always detectable, due to stopping criteria such as time and iterations limits. In that sense, the multicommodity flow system has to be solved to ensure the feasibility of each node. Note that this corresponds to find any feasible solution, which is less costly than finding the minimum-cost multicommodity flow (problem (5.7)).

### 5.2.3 Local optimality cuts

Similar to the feasibility cuts, optimality cuts also aim to prevent undesired topologies to be repeatedly generated. Indeed, considering the subproblem (5.7), given a topology $(\bar{A}_0, \bar{A}_1)$, $A = \bar{A}_0 \cup \bar{A}_1$ and $\bar{A}_0 \cap \bar{A}_1 = \emptyset$, where $\bar{A}_1$ is the set of opened arc, and $\bar{A}_0$ is the set of closed arcs, if $Z_{01} = \sum_{a \in \bar{A}_1} f_a + \bar{c}(\bar{A}_0, \bar{A}_1) \geq \bar{Z}$, then that topology is not desired, and inequality (5.8) must be included.

$$\bar{c}(A_0, A_1) = min\{\sum_{k \in K} \sum_{a \in A_1} c_a^k x_a^k : (5.3) - (5.5)\} \tag{5.7}$$

In fact, such inequality has been proposed as combinatorial Benders cut, by Codato and Fischetti [53]. However, in the context of Benders decomposition, cuts

of type (5.8) are generated to break infeasibility. Note that the global feasibility cut (5.6) is indeed a stronger combinatorial Benders cut.

$$\sum_{a \in \bar{A}_1} (1 - y_a) + \sum_{a \in \bar{A}_0} y_a \geq 1 \tag{5.8}$$

Given that they exclude feasible solutions, thus treated as pseudo-cuts for the problem, inequalities (5.8) are not valid for the entire set of integer solutions of the problem, but one can be sure that the optimal point is not cut off. Note that the same statement can be done for (5.1) and (5.2). In contrast to combinatorial Benders cut, local optimality pseudo-cuts intend to cut off solution areas that have already been explored and are known not to contain better solutions. In that sense, the proposed pseudo-cuts are similar to the local branching constraints [81], for $k = 0$ neighborhoods. In the present work, inequalities (5.8) are separated only when a feasible solution is already available, provided by the upper bounding procedures of section 5.6.

## 5.3 Variable fixing

Another important feature to improve performances of the Branch-and-Cut algorithm is the variable fixing strategy, which includes reductions in the bounds of variables. Indeed, much information can be extracted from feasibility, added cuts and reduced costs, to make inferences about the actual possible variable values.

### 5.3.1 Reduced cost fixing

A very classical type of variable fixing regards the reduced costs of binary variables (see [182]). We use the Lagrangian reduced costs to infer if a variable can be fixed to either 0 or 1. Considering the reduced costs $\bar{f}$ at the final solution given by the Volume at a certain node of the tree, one can say that:

If $\bar{f}_a < 0$ and $\bar{z}_L - \bar{f}_a \geq \bar{Z}$ then one can fix $y_a = 1$
If $\bar{f}_a > 0$ and $\bar{z}_L + \bar{f}_a \geq \bar{Z}$ then one can fix $y_a = 0$

### 5.3.2 Flow upper bound fixing

As reduced cost were previously used to determine binary variable fixing, one can consider $\bar{c}^k_a, a \in A \setminus A_0, k \in K$, the reduced costs of flow variables, to set

reductions in the upper bound of these variables. According to Chouman *et al.* [50], given $(\hat{x}, \hat{y})$ the optimal solution to a Lagrangian subproblem, if for an arc $a \in A \setminus A_0$ and commodity $k \in K$, $\hat{x}_a^k = 0, \bar{c}_a^k > 0$ and $z_L + \bar{f}_a(1 - \hat{y}_a) + \bar{c}_a^k b_a^k > \bar{Z}$, then $x_a^k \leq d_a^k < b_a^k$, where:

$$d_a^k = \frac{\bar{Z} - z_L - \bar{f}_a(1 - \hat{y}_a)}{\bar{c}_a^k} \tag{5.9}$$

The reasoning behind this domain reduction is based on the consequences, to the current lower bound, of the addition of constraint $x_a^k > d_a^k$. If the new lower bound exceeds the current best integer solution value $\bar{Z}$, then the constraint $x_a^k \leq d_a^k$ must be added.

Note that if no lower bounds are imposed on the flows, $\bar{c}_a^k > 0$ implies $\hat{x}_a^k = 0$, however if lower bounds $x_a^k \geq l_a^k$, $l_a^k > 0$ exist for some $a \in A \setminus A_0$ and $k \in K$, then it is possible to encounter $\bar{c}_a^k > 0$, while $\hat{x}_a^k > 0$. Such a situation may occur due to the flow lower bound fixing strategy proposed in section 5.3.3. Hence, we extend the proposition provided in [50], to the case where $\hat{x}_a^k \geq 0$ and $\bar{c}_a^k > 0$.

**Corollary 5.3.1.** *given $(\hat{x}, \hat{y})$ the optimal solution to the Lagrangian subproblem, if for an arc $a \in A \setminus A_0$ and commodity $k \in A \setminus A$, $\hat{x}_a^k \geq 0, \bar{c}_a^k > 0$ and $z_L + \bar{f}_a(1 - \hat{y}_a) + \bar{c}_a^k(b_a^k - \hat{x}_a^k) > \bar{Z}$, then $x_a^k \leq \hat{x}_a^k + d_a^k < b_a^k$.*

*Proof.* If $\hat{x}_a^k = 0$ then the proof resumes to what is presented in Proposition 1 of [50]. For $\hat{x}_a^k > 0$, let consider that the constraint $x_a^k > d_a^k$ is added to the problem. The new solution value for the Lagrangian subproblem must be greater than $z'_L = z_L + \bar{c}_a^k(d_a^k)$, thus if $z'_L \geq \bar{Z}$, (which is true, given the definition of $d_a^k$) then $x_a^k \leq d_a^k$.

□

### 5.3.3 Flow lower bound fixing

From another perspective, one may determine lower bounds on the flow of opened arcs considering the optimal solution $(\hat{x}, \hat{y})$ of value $z_L$ to the current Lagrangian subproblem, and the reduced costs $\bar{c}_a^k, a \in A \setminus A_0, k \in K$. Remind that, for all $a \in A_1$, $\hat{x}_a^k \geq 0$ if $\bar{c}_a^k < 0$, and $\hat{x}_a^k = l_a^k$ if $\bar{c}_a^k > 0$, being $l_a^k \geq 0$ the current lower bound on the corresponding flow.

**Proposition 3.** *For all arc $a \in A_1$, let consider the reduced costs in a non-decreasing order: $\bar{c}_a^{k_1} \leq \bar{c}_a^{k_2} \leq ... \leq \bar{c}_a^{k_n} < 0$, such that $n = |\bar{C}|$, $\bar{C}$ the set of indices of commodities with negative reduced costs in that arc. If for a certain $a \in A_1, k_i, i \in \bar{C}$, one have $z_L - \bar{c}_a^{k_i}(\hat{x}_a^{k_i} - l_a^{k_i}) > \bar{Z}$, and $\sum_{j=(i+1)}^{n} \bar{c}_a^{k_j}(b_a^{k_j} - \hat{x}_a^{k_j}) = 0$, then $x_a^{k_i} \geq \hat{x}_a^{k_i} - m_a^{k_i}$ can be added, and $\hat{x}_a^{k_i} - m_a^{k_i} \leq b_a^k$ .*

$$m_a^{k_i} = -(\bar{Z} - z_L)/\bar{c}_a^{k_i} \tag{5.10}$$

*Proof.* Since we are dealing with a minimization function, only negative reduced costs need to be taken in consideration. Moreover, if the restriction $x_a^{k_i} < \hat{x}_a^{k_i} - m_a^{k_i}$ is added to the problem, the new solution value to the subproblem must be greater than:

$$z'_L = z_L - \bar{c}_a^{k_i} m_a^{k_i} + \sum_{j=(i+1)}^{n} \bar{c}_a^{k_j}(b_a^{k_j} - \hat{x}_a^{k_j})$$

Since $z_L - \bar{c}_a^{k_i} m_a^{k_i} = \bar{Z}$, thus $z'_L \geq \bar{Z}$ if $\sum_{j=(i+1)}^{n} \bar{c}_a^{k_j}(b_a^{k_j} - \hat{x}_a^{k_j}) = 0$. Hence, if $z'_L \geq \bar{Z}$, so we can state that $x_a^{k_i} \geq \hat{x}_a^{k_i} - m_a^{k_i}$, since smaller values of $x_a^{k_i}$ would only increase the value of $z'_L$.

□

### 5.3.4 Constraint propagation

In the present case, remark that a reduction on the domain of a binary variable, due to branching or variable fixing, may result in more variables being fixed to satisfy local and global constraints. A concept proper to the *constraint programming* area, constraint propagation is a process in which a reduction in the domain of a decision variable is propagated to all of the other constraints being related to that variable, maybe resulting in more domain reductions [34].

For a constraint generally formulated as $\sum_{a \in B} \gamma_a y_a \geq b$, where $B \subseteq A$ and $\gamma_a > 0 \ \forall \ a \in B$, given the configuration $A'_1$, $A'_0$ and $A'_*$:

- If $\sum_{a \in \{B \cap A'_*\}} \gamma_a = b - \sum_{a \in \{B \cap A'_1\}} \gamma_a$, then $y_a = 1, \ \forall \ a \in A_*$
- If $\sum_{a \in \{B \cap A'_*\}} \gamma_a < b - \sum_{a \in \{B \cap A'_1\}} \gamma_a$, then the node can be pruned

For a constraint generally formulated as $\sum_{a \in B} \gamma_a y_a \leq b$, where $B \subseteq A$ and $\gamma_a > 0 \ \forall \ a \in B$:

- If $\sum_{a \in \{B \cap A'_1\}} \gamma_a = b$, then $y_a = 0, \ \forall \ a \in A_*$

- If $\sum_{a \in \{B \cap A'_1\}} \gamma_a > b$, then the node can be pruned

In the case of constraints (5.8), if $A'_1 = \bar{A}_1$ and $A'_0 = \bar{A}_0$, then the node can be pruned. However, if $A'_1 \subseteq \bar{A}_1$, $A'_0 \subseteq \bar{A}_0$ and $|A'_*| = 1$, then for $a \in A'_*$, $y_a = 1$ if $a \in \bar{A}_0$, otherwise $y_a = 0$.

### 5.3.5 Connectivity-based fixing

Another type of constraint propagation may concern the flow balance constraints, especially when arcs have been closed. In [50], Chouman *et al.* proposed an algorithm to test connectivity, based on graph traversal procedures, that for every node, performs complete forward and backward traversals of the graph, while labeling arcs. At the end of the labeling procedure, one is able to tell if an arc makes part of some path for a given commodity, or not at all. Moreover, for an outgoing arc starting from a commodity origin, one can infer if all possible paths for that commodity include that arc. The same can be inferred for an incoming arc ending into a commodity destination.

With the connectivity test, one can conclude that $x^k_a = 0$, if arc $a \in A \setminus A_0$ does not belong to any path between the origin and destination of commodity $k \in K$. In contrast, if an arc $a$ belongs to all possible paths for $k$, then $x^k_a = q^k$, and if $a \in A_*$, $y_a = 1$. Moreover, if an arc $a \in A_*$ do not belong to any possible path for any commodity, then $y_a = 0$. Note that for a closed arc, the corresponding flow upper bounds are consequently equal to zero.

In addition, considering the flow lower bounds $l$, and the flow upper bounds $b$, at every node $i \in N$, and for commodities $k \in K$, such that $i \neq O(k)$ and $i \neq D(k)$:

- If $\sum_{j \in N_i^-} b^k_{ji} < b^k_{ih}$, then $b^k_{ih} = \sum_{j \in N_i^-} b^k_{ji}$, for $h \in N_i^+$;
- If $\sum_{j \in N_i^+} b^k_{ij} < b^k_{hi}$, then $b^k_{hi} = \sum_{j \in N_i^+} b^k_{ij}$, for $h \in N_i^-$;
- If $\sum_{j \in N_i^-} l^k_{ji} > \sum_{h \in N_i^+} b^k_{ih}$, then the node must be pruned;
- If $\sum_{j \in N_i^+} l^k_{ij} > \sum_{h \in N_i^-} b^k_{hi}$, then the node must be pruned;
- If $\sum_{j \in N_i^-} l^k_{ji} > \sum_{h \in N_i^+ \setminus \{r\}} b^k_{ih}$, then
  $l^k_{ir} = max\{l^k_{ir},\ \sum_{j \in N_i^-} l^k_{ji} - \sum_{h \in N_i^+ \setminus \{r\}} b^k_{ih}\}$ and $y_{ir} = 1$, for $r \in N_i^+$;
- If $\sum_{j \in N_i^+} l^k_{ij} > \sum_{h \in N_i^- \setminus \{r\}} b^k_{hi}$, then
  $l^k_{ri} = max\{l^k_{ri},\ \sum_{j \in N_i^+} l^k_{ij} - \sum_{h \in N_i^- \setminus \{r\}} b^k_{hi}\}$ and $y_{ri} = 1$, for $r \in N_i^-$;

The first two points express the fact that a commodity flow in a outgoing (incoming) arc is at most the sum of maximum flow arriving into (leaving) the transshipment node $i$. The following two points are related to flow conservation, and the two final ones are derived from the fact that if the minimum flow entering (leaving) a node is greater than the maximum possible flow that can leave (enter) that node, after artificially closing an outgoing (incoming) arc, then that arc must be open, and its flow will be at least the exceeding amount of flow that cannot be satisfied by the other arcs.

### 5.3.6 LP-based fixing

A final type of variable fixing was considered, based on how an artificially fixed arc may impact the current node linear program. Indeed, as mentioned before, if a configuration $(A'_0, A'_1, A'_*)$ is not feasible, or a computed lower bound on its optimal value is greater than $\bar{Z}$, then such an arc configuration is not interesting and may be discarded.

Hence, the variable fixing procedure considers an arc whose value $\tilde{y}$ is close to either its lower or upper bounds, and fix it to the opposite bound value, e.g. an arc $a \in A_*$ presents $\tilde{y}_a \approx 1$, so we fix $y_a = 0$, then the modified LP is solved with the Volume Algorithm (with a reduced limit of iterations), using the current solution as warmstart. If the obtained lower bound is greater than $\bar{Z}$, then the arc must be fixed to 1.

In fact, this variable fixing is most likely to occur when the gap is already sharp, so the procedure may contribute to close it. Therefore, the LP-based fixing procedure is called when the current relative gap is less than or equal to 1.0%. In that case, we compute, for every arc $a \in A_*$ presenting $\tilde{y}_a \leq 0.1$ ($\tilde{y}_a \geq 0.9$), the Lagrangian lower bound $z_L$ for the LP considering $y_a = 1$ ($y_a = 0$). If $z_L \geq \bar{Z}$, then arc $a$ is fixed to $y_a = 0$ ($y_a = 1$).

## 5.4 Branching

For the present problem, branching is performed on design variables, resulting in two branches (0-child and 1-child nodes) each time. Here, an adaptive rule is implemented, where the *strong branching* strategy (see [2] and the references therein) is turned on and off accordingly to the lower bound evolution.

The strong branching strategy implemented selects five arcs as branching candidates. Namely, such a branching strategy consists in solving, with a limit of a few iterations, every candidate child, choosing the best variable to branch on, provided a certain score function. Being $C$ the set of branching candidates, the

chosen one will be the best, w.r.t $max_{a \in C}\{min(z_1(a), z_0(a))\}$, where $z_1(a)$ and $z_0(a)$ are the computed objective values for the 1-child and 0-child of candidate $a \in C$, respectively. To select set $C$, we use the linear relaxation approximate vector $\tilde{y}$, provided by the Volume Algorithm, to take the five variables with the greatest values of $w_a * min\{\tilde{y}_a, 1 - \tilde{y}_a\}$ as candidates, $a \in A_*$. In that manner, we intend to evaluate arcs that are close to 0.5 and that have great capacities.

Even if such a branching strategy has provided good results in the present work and in other contexts, its drawback may be to spend a considerable amount of computational time solving the candidate children (10 in this case). In that sense, we switch to a faster strategy every time the gap reduction starts to tail off. Namely, if the current nominal gap ($\bar{Z}$ minus the current lower bound) has not improved at least 0.1% for more than 5 iterations, then the list of candidates is set to 1, and a different rule to select the candidate is applied. The algorithm returns to strong branching after a 1% improvement is perceived in the nominal gap, compared to the last iteration where strong branching was performed.

When a single candidate must be selected for branching, the *reliability branching* proposed in [2] is applied. In this case, we compute an average gain per unit, obtained selecting a certain branching variable, and use a score function based on those averages to choose the next candidate. In fact, that is called *pseudo-cost branching* [32], but a reliability parameter $\sigma$ is introduced to exclude those variables that have not been selected for branching at least $\sigma$ times.

As described in [2], for a variable $a \in A_*$, let $\delta_a^0$ and $\delta_a^1$ be the increase in the objective values from the parent node to the 0-child and the 1-child, respectively; their gain per unit is then: $\rho_a^0 = \delta_a^0/\tilde{y}_a$ and $\rho_a^1 = \delta_a^0/(1 - \tilde{y}_a)$. Furthermore, let $\eta_a^0$ and $\eta_a^1$ be the number of times that $\delta_a^0$ and $\delta_a^1$ were computed and their respective child was feasible. Hence, the average gain per unit $\bar{\rho}_a^h$ is given by the sum of the $\eta_a^h$ values of $\delta_a^h$ divided by $\eta_a^h$, $h = 0, 1$. Then, the candidate variable $\bar{a}$ is chosen accordingly to the score function:

$$\bar{a} = argmax_{a \in A_*}\{min[\tilde{y}_a\rho_a^0,\ (1 - \tilde{y}_a)\rho_a^1] : min(\eta_a^1,\ \eta_a^0) \geq \sigma\}$$

If no reliable variable exists, then we select the variable whose $\tilde{y}$ is the closest to 0.5. The reliability parameter $\sigma$ was set to 10.

## 5.5 Pruning

In its turn, the pruning strategy is extremely important to reduce the size of the search tree. Here, we discuss some rules that enable us to prune subtrees that are known to not contain an optimal solution. As one can predict, a first intuitive

common rule is to fathom infeasible nodes and the ones that provide a lower bound higher than the current upper bound. Furthermore, even though it rarely occurs, if a solution provided by the Volume-based Relax-and-Cut is feasible, then it is also optimal to that subtree, and no further exploration is needed, thus that branch is pruned.

Remark that while performing strong branching, one can extract information leading to pruning or variable fixing in the current node. Indeed, if both 0-child and 1-child are infeasible or have their solution value greater than the current upper bound, the current node can be pruned, since no optimal solution can be obtained with the present $(A_1, A_0, A*)$-configuration. Furthermore, if only the 0-child a candidate $a \in C$ is infeasible or provided a too high solution value, the arc candidate can be opened, i.e $y_a = 1$. The same can be stated for the 1-child. Obviously, if any child node fits in some pruning situation, the corresponding branch is not included in the tree.

Moreover, note that more pruning occasions may happen during the constraint propagation of section 5.3.4. Regarding the connectivity-based fixing of section 5.3.5, if no connected path exists for a certain commodity, the node can be fathomed. In addition, pruning must occur if a flow variable $x_a^k$ must be set to the corresponding demand, and the current upper bound $b_a^k$ is inferior to that quantity, i.e. $b_a^k < q^k$. The same must be done if it is detected that an arc can be closed, while it is currently fixed opened.

Finally, proposed by Holmberg and Yuan [128], if for a node $(A_0, A_1, A_*)$, $\bar{c}(A_0,\ A_1 \cup A_*) + \sum_{a \in A_1} f_a \geq \bar{Z}$ (see equation (5.7)) then the node is fathomed, since it cannot lead to a better solution than the current one. Additionally, according to [128], if $\bar{A}_0 \subseteq \{A_1 \cup A_*\}$, the set of arcs not used in the optimal solution of $\bar{c}(A_0,\ A_1 \cup A_*)$, is such that $\bar{A}_0 \supseteq A_*$, then no better primal solutions can be found by further branching, thus the node can be fathomed. Such a pruning occasion may occur while computing upper bounds, as in the next section.

## 5.6 Integer feasible solutions

A first way of obtaining a feasible solution is to solve the multicommodity flow system (5.3)-(5.5) every time all arcs are fixed, more precisely the problem (5.7), for a better quality solution. The system once solved, we add to the solution value the fixed cost of arcs containing some amount of flow, closing the others to construct the entire solution. As mentioned before, the Volume Algorithm rarely provides a feasible solution, thus a linear programming solver is needed to solve the multicommodity flow system.

Actually, in the implemented algorithm, every time $|A_*| \leq 0.3|A|$, we solve problem $\bar{c}(A_0,\ A_1 \cup A_*)$, and an integer feasible solution is constructed as discussed above. In this case, observe that the pruning rules defined at the end of section 5.5 may be applied. Moreover, if the obtained integer solution is not better than the current best one, an optimality cut can be added. Observe that an optimality cut can be globally considered, but given the structure of a binary search tree, it can only be of some interest in the local subtree, since it will be satisfied everywhere else. If infeasibility is detected, global feasibility cuts are added.

In case $A_* \neq \emptyset$, and the solution $\bar{x}$ of $\bar{c}(A_0, A_1 \cup A_*)$ is such that $\bar{x}_a^k > b_a^k$, for some arc $a$ and commodity $k$, the current flow bounds are added to (5.7) i.e. $\bar{x}_a^k \leq b_a^k,\ \forall\ a \in A \setminus A_0, k \in K$, and the problem is reoptimized (the same is done with the flow lower bounds). That surely will not produce a better solution, but may increase the chances of pruning.

### 5.6.1 Feasibility pump

Additionally, a heuristic based on the Feasibility Pump algorithm proposed by Fischetti *et al.* [80] was implemented. Originally, the method was proposed as a solution approach for the problem of finding a feasible solution for a generic MIP. Later, several other works have proposed improvements to the algorithm [82, 1, 33, 71, 99]. The procedure keeps $\tilde{y}$ a feasible solution to the linear relaxation of the problem, and an integer point $y^*$. At each iteration, those vectors are updated, trying to reduce their distance as much as possible.

The Feasibility Pump algorithm is formally presented in Algorithm 10. Considering $\bar{P}$ the continuous space defined by the constraints of the problem, the heuristic initializes with the linear relaxation vector $\tilde{y}$, and a rounding procedure is applied to get the initial integer vector $y^*$. Then, at each iteration the integer (infeasible) point is projected onto $\bar{P}$, with the computation of $argmin_y\{\Delta(y, y^*) : x, y \in \bar{P}\}$. That generates the next $\tilde{y}$ point, which is rounded to provide, in its turn, an updated $y^*$. The function $\Delta(y, y^*)$ measures the distance of $y^*$ from $\bar{P}$, to be minimized.

According to Fischetti *et al.*, the algorithm has the tendency to stall, cycling over the same integer points. In that case, a perturbation step is performed in the attempt of exploring other regions of the solution space. The procedure stops either when an integer feasible solution has been returned, or when some stopping criteria have been reached (time or iterations limit).

**Algorithm 10** The Feasibility Pump

Initialize $\tilde{y} := argmin_y\{cx + fy : x, y \in \bar{P}\}$
If $\tilde{y}$ is integer, return$(\tilde{y}, \tilde{x})$;
$y^* \leftarrow$ Round $(\tilde{y})$
**while** Stopping-criteria not reached **do**
    Compute $\tilde{y} := argmin_y\{\Delta(y, y^*) : x, y \in \bar{P}\}$
    If $\tilde{y}$ is integer, return$(\tilde{y}, \tilde{x})$;
    $y^* \leftarrow$ Round $(\tilde{y})$
    Perturb $y^*$ if *cycling* is detected.

In the implemented Branch-and-Cut scheme, the Feasibility Pump algorithm runs in every node whose depth level *d_level* in the search tree is divisible by 20 (*d_level* mod 20 = 0), since running it more often may be too time consuming. Therefore, for a node with proper depth, the heuristic considers only the arcs in $A'_* = \{a \in A_* : 0.01 \leq \tilde{y}_a \leq 0.99\}$, artificially setting to 1, arcs $\{a \in A_* : \tilde{y}_a > 0.99\}$ and to 0, the arcs $\{a \in A_* : 0.01 < \tilde{y}_a\}$. This may prevent having to deal with too large projection problems. The initial vector $\tilde{y}$ is then the approximate solution provided by the Volume, while solving the current node LP.

The rounding procedure consists in setting $y^*_a = 1$ for arcs in $A^1_* = \{a \in A'_* : \tilde{y}_a \geq \epsilon\}$, and setting $y^*_a = 0$ for arcs in $A^0_* = \{a \in A'_* : \tilde{y}_a < \epsilon\}$. Considering $A'_1 = A_1 \cup \{a \in A_* : \tilde{y}_a > 0.99\}$, and $A'_0 = A_0 \cup \{a \in A_* : \tilde{y}_a < 0.01\}$ the function $\Delta(y, y^*)$ was set to:

$$\Delta(y, y^*) := \sum_{a \in A^0_* \cup A^1_* \cup A'_1} \sum_{k \in k} c^k_a x^k_a + \sum_{a \in A^0_*} f_a y_a$$

In that manner, note that the new continuous solution $\tilde{y}$ tends to present values close to 1 for arcs in $A^*_1$, since no fixed cost is imposed on those arcs. In contrast, for arcs in $A^*_0$, the values tend to be close to 0, due to the presence of fixed costs. The projection problem is solved by the Volume Algorithm, and the new $\tilde{y}$ is set to the linear approximate solution provided by the solver.

Besides returning when an integer $\tilde{y}$ is obtained, the implemented heuristic stops when a limit of 5 iterations has been reached or when cycling is detected. Indeed, no perturbation is performed in the present feasibility pump algorithm, however since it is executed in different nodes, we may consider each different node configuration as a perturbed integer point. As a matter of fact, for every heuristic run, the initial integer vector $y^*$ is constructed as follows: in the presence of random values $0 \leq r_a \leq 1$, $\forall a \in A'_*$, $y^*_a = 1$ if $r_a \leq \tilde{y}_a$, otherwise $y^*_a = 0$ (the reasoning behind this procedure is commented in section 3.5.2). Moreover, if a precomputed integer feasible solution is available, up to $0.01|A|$ arcs are randomly

chosen, so $y_a^* = \bar{y}_a$, the value of those arcs in that feasible solution.

As one is not certain to obtain a feasible solution with the Volume Algorithm, problem $\bar{c}(A'_0 \cup A^0_*,\ A'_1 \cup A^1_*)$ is solved, and a feasible integer solution is constructed as described at the beginning of this section. The rounding parameter $\epsilon$ is initially set to 0.3 and then it is multiplied by 0.8 every time a heuristic run finishes with an infeasible pair $(A'_0 \cup A^0_*,\ A'_1 \cup A^1_*)$, and multiplied by 1.8 every time it returns a feasible integer solution worse than the current one if any. However, $\epsilon$ is forced to stay in the interval [0.3, 0.7].

## 5.7 The algorithm

The Branch-and-Cut algorithm is formally presented below. In *Step 1*, the upper bound is initialized by a heuristic procedure *ad hoc*, being the node pool $\Gamma$ initialized with the root node, which is set as the current one. In its turn, *Step 2* regards the tree search strategy. In fact, performances may depend on the order in which the nodes stored in the pool are evaluated. Some strategies are the breadth-first and the depth-first approaches, where the ordering is based on the depth levels of the search tree. The former prioritizes nodes in a horizontal way, while the latter prioritizes deeper nodes in the tree.

Another option is the best-first approach, which provided a score function, the ordering is established according to the node score given by that function. Moreover, hybrid strategies are possible, switching between approaches, as the search goes through. In the implemented algorithm, the best-first approach is implemented, prioritizing nodes with the smallest preprocessed LP solution values. Remind that a number of Volume iterations are performed for every child node before being inserted into the pool, thus a preprocessed bound value exists for every child.

In fact, the best-first approach is combined with a *diving* strategy, that becomes active when the value computed for a child node (provided a selected branching variable) did not increase at least 0.1% w.r.t the bound value $z_L$ of the parent node. In this case, the child with the least solution value is automatically set as the next node to be evaluated. That same strategy occurs, if one of the selected children is fathomable, in which case the algorithm dive in the search tree, setting the other child as the next node to be evaluated.

The whole rationale behind the present tree search strategy is to focus on raising lower bounds as fast as possible. Since the current lower bound corresponds to the smallest solution value in $\Gamma$, the best-first strategy seems to be the most appropriate to increase the lower bound quickly. In its turn, diving is performed

in the attempt to prevent the algorithm from tailing off, which happens when $\Gamma$ has too many items with values very close to the current lower bound. Moreover, to accelerate the algorithm, when diving is performed or when it starts to tail off, a *reduced iteration* is considered: reliability branching is performed, rather than strong branching (only one candidate variable is evaluated); and time-consuming features, such as LP-based fixing and heuristics, are not allowed to run.

We take into consideration the fact that the solution value provided by the Volume Algorithm may be up to 1% smaller than the linear relaxation value. Hence, If the algorithm has dived 3 times or more, the current node is solved exactly, by the simplex algorithm. The same happens at a not reduced iteration, when the relative gap between the current node value $z_L$ and the upper bound $\bar{Z}$ is less than or equal to 1%. A similar strategy is considered in [27].

The main operations performed on each node are listed in *Step 3*. Before being solved, in *Step 3(a)* the node is preprocessed in order to verify if some variable can be fixed by constraint propagation and connectivity, provided that branching has already fixed one. We stress that connectivity-based fixing is performed only if arcs have been closed previously. Afterwards, the node LP is solved by the Relax-and-Cut procedure described in section 4.1 (*Step 3(b)*). More variable fixing is attempted in *Step 3(c)*, followed by the separation of Lagrangian cardinality cuts for the current subtree, in *Step 3(d)*.

Even though cuts may be removed from the Relax-and-Cut procedure (according to section 4.1), the removed cuts are not completely deleted from that node, remaining available for the corresponding subtree. Nevertheless, if a local cut becomes satisfied in some node, due to the changes in variable bounds performed by branching or variable fixing, the cut is deleted for the rest of the subtree of that node.

In order to enhance the algorithm performances, the Volume iterations are stopped if the current dual solution value reaches the current upper bound. Another implemented feature aiming better performances is the utilization of previous dual points as warmstarts for the Volume Algorithm. In that manner, the dual solution of a certain node is taken as warmstart in strong branching preprocessing iterations. In their turn, the preprocessed dual points work as warmstart in the node evaluation.

In *Step 4* we look for feasible solutions as discussed in section 5.6. Note that if a feasible integer solution is obtained, an optimality cut is always available, since either the current or the new solution will represent an undesired topology.

Finally, strong branching is performed in *Step 5*, as described in section 5.4. Once the branching variable is selected, the feasibility of the children nodes is

checked by solving the multicommodity flow system. Remark that feasibility cuts (5.6) are only globally valid if the detected infeasibility is due to closed arcs, regardless of updated flow bounds. In case the current design is infeasible only if flow bounds are imposed, then those cuts are just locally valid, and there is no need to add them since the subtree is pruned anyways. For that reason, we first solve $(5.3) - (5.5)$, adding flow bounds afterwards, if violated, and the multicommodity flow system is resolved. The algorithm is stated as follows:

*Step 1.* Initialize the upper bound $\bar{Z}$, and the root node $(A_1, A_0, A_*)$, $A_1 = A_0 = \emptyset$, $A_* = A$. $\Gamma \leftarrow (A_1, A_0, A_*)$.

*Step 2.* Tree strategy: If $\Gamma = \emptyset$ *Stop.* Otherwise, select a node node from $\Gamma$.

*Step 3.* Node evaluation:

(a) Perform variable fixing: *constraint propagation* and *connectivity-based fixing*, until no more arcs can be fixed. If infeasibility is identified, then *fathom*, and add *feasibility cut.* Go to *Step 2.*

(b) Solve the LP node with the Relax-and-Cut algorithm. Get $z_L$, $(\hat{x}, \hat{y})$ and $(\tilde{x}, \tilde{y})$. If $z_L \geq \bar{Z}$, *fathom* and go to *Step 2*

(c) Perform variable fixing: first *reduced cost fixing.* Then, if some arc was fixed, perform *constraint propagation* and *connectivity-based fixing* until no more arcs can be fixed. Afterwards, check for *flow bounds fixing.* If infeasibility is identified, then *fathom*, add *feasibility cut*, and go to *Step 2.* Otherwise, perform *LP-based fixing* if allowed.

(d) Perform cut separation for *Lagrangian cardinality cuts*

*Step 4.* Upper bound:

(a) If $|A_*| \leq 0.3|A|$ solve $\bar{c}(A_0, \{A_1 \cup A_*\})$. If a new feasible solution $(\bar{x}, \bar{y})$ of value $Z$ can be obtained, then for $Z < \bar{Z}$ update the current best solution. Add *optimality cut.* If the problem is infeasible, add *feasibility cut*, fathom and go to *Step 2.*

(b) If $\bar{c}(A_0, \{A_1 \cup A_*\}) + \sum_{a \in A_1} f_a \geq \bar{Z}$, or $\bar{A}_0 \supseteq A_*$ such that $\bar{A}_0 = \{a : \sum_{k \in K} \bar{x}_a^k = 0\}$, then *fathom* and go to *Step 2.*

(c) Perform Feasilibity Pump heuristic if allowed. Add *optimality cut* if a feasible solution is obtained.

*Step 5.* Branching:

(a) If $A_* = \emptyset$ go to *Step 2.* Otherwise, select candidates set $C \subseteq A_*$

(b) Perform Volume iterations for the 0-child and 1-child, for each candidate in $a \in C$, to get $z_0(a)$ and $z_1(a)$ the respective preprocessed values. Fix variables according to feasibility and lower bounds. If pruning can be applied: *fathom* and go to *Step 2*.

(c) Select the best candidate and verify feasiblity, computing:

$$\{(x_a^k)_{a \in A \setminus A_0, k \in K} : l_a^k \leq x_a^k \leq b_a^k, (5.3) - (5.5)\}$$

for each selected children. If infeasibility is detected add *feasibility cut* if applicable. If both children are infeasible, *fathom* and go to *Step 2*. If only one of the children is infeasible, then set the other child node as the current node and (*dive*) go to *Step 3*.

(d) If $min\{z_0(a), z_1(a)\}$ is close to the current lower bound, then set the corresponding child as the next node, and put the other one into $\Gamma$. *Dive*: go to *Step 3*. Otherwise, insert both selected children nodes into $\Gamma$. Go to *Step 2*.

## 5.8 Computational experiments

We now report the computational experiments carried out with the Volume-based Branch-and-Cut. At the first moment, we evaluate the effectiveness of the cuts implemented, as well as the efficiency of variable fixing features. The small to medium-sized benchmark instances Canad-C and Canad-R were used in this first testbed. The results for the best configuration are presented right afterwards. A comparison to the results presented by Chouman *et al.* in [50] is also provided (their work only considered Canad-C and Canad-R instances). The final computational experiments involved a Branch-and-Cut heuristic version, based on the Feasibility Pump algorithm (see section 5.8.3 for mode details).

The tests were all run in single thread. The largest instances Canad-N and groups A to H were tested on a 60Gb RAM, Intel(R) Xeon(R) CPU E7-8890 v3 at 2.50 GHz, Linux machine. Especially for the comparison with literature results, the instances Canad-C and Canad-R were tested on a 30Gb RAM, Intel(R) Xeon(R) CPU E5-2687W v3 at 3.10GHz Linux machine, since the former is not present in the PassMark CPU Database used for time comparisons. Indeed, to compare performances in terms of computational times considering different CPUs, we apply the approach of [74], taking into account the CPU scores from `https://www.cpubenchmark.net`. The normalized time is obtained by $\bar{t} = t(P_l/P_0)$, where $P_l$ is the PassMark CPU score of the computer used in the literature work, and $P_0$ is the score of the computer used in the present work.

The Relax-and-Cut algorithm was allowed to run for a time limit of 3 hours and a limit of 1000 iterations for the root node, and 250 iterations in the rest of the Branch-and-Cut algorithm. Cutsets are separated in the root node only. Furthermore, an additional stopping criterion stops the solver if the solution value has not increased for 100 Volume iterations. In the case where the current Branch-and-Cut iteration is a reduced one that limit is set to 50, however that criterion is checked only after 100 Volume iterations. The initial upper bounds were set to the best heuristic solution existing in the literature. Moreover, the COIN-OR BCP framework (available at `https://projects.coin-or.org/Bcp`) was used for the implementation of the present Branch-and-Cut algorithm.

The software Cplex version 12.8 was used to solve multicommodity flows and LPs, when required. Many other efficient software for multicommodity flows exist in the literature (and surveyed in [88]), but Cplex remains competitive, being the chosen one to simplify the implementation.

### 5.8.1 Cuts and variable fixing efficiency

In order to check the efficiency of cuts and variable fixing procedures, here it is presented the results obtained while disabling each type of cut, and in how many nodes the variable fixing procedures were successful in fixing variables indeed. Actually, it is shown in [50] that this latter feature can be worthful to enhance the performance of Branch-and-Cut schemes for the present problem, so we just focus on the efficiency of the mentioned procedures themselves in the implemented algorithm.

The impact of solving the minimum cost multicommodity flows for pruning purposes like described at the end of section 5.5 is also tested. We call it the 'pruning heuristic' (Step 4a and 4b of the algorithm in section 5.7). It is provided the number of times an improving solution is obtained or pruning has happened, w.r.t the number of times those subproblems were solved.

The Feasibility Pump heuristic is not considered for instances Canad-R and Canad-C, since the provided initial upper bounds are already near-optimal for those instances. In that sense, performing Feasibility Pump would likely represent a loss in speed performance. However, such a heuristic is tested for those instances in section 5.8.3, where no initial upper bound is provided.

The Canad-R instances were divided into two groups: RI, the smallest ones, and RII, the greatest ones. In that manner, Table 5.1 reports the average optimality gaps 'Avg Gap (%)', the minimum optimality gap 'Min Gap (%)', and the maximum optimality gap 'Max Gap (%)' observed in each group of instances. All those values are given in percentages. Moreover, the average computational times,

in seconds, and the average number of nodes are reported in columns 'Avg Time (s)' and 'Avg Nodes', respectively. The ratio giving the average time per node is given in column 'Avg Time/Node'. A time limit of 10 hours was set for this current testbed.

Then, each subtable presents the performance for each different cut configuration. The first, 'All cuts', is related to the case where all mentioned cuts are considered, and the following three subtables refer to configurations where the respective type of cut is uniquely disabled. Namely, '(-) Opt cuts' refers to the configuration where only optimality cuts are disabled, '(-) Lag cuts' refers to the case where only Lagrangian cardinality cuts are disabled, and '(-) Feas cuts' to when solely feasibility cuts are disabled. We also tested the case where no cuts, apart from cover inequalities, are allowed to be separated (subtable 'No cuts'), and the case where they are all considered, but only for constraint propagation, not being included in dual programs (subtable 'All cuts not in Dual').

As can be seen in Table 5.1, the average time per node does not vary too much from a configuration to another. Even when no cuts are separated, the times did not decrease significantly, instead it was sometimes greater. That must indicate an advantage of constraint propagation in terms of computational times since, when no cuts are available, such variable fixing procedure can not be applied (only for cover inequalities). Indeed, fixing binary variables may significantly reduce the size of node LPs, especially when they are fixed to zero.

Table 5.2 shows the average number of cuts separated throughout the whole search tree, considering all different Branch-and-Cut runs. Columns 'Lag cuts', 'Feas cuts', and 'Opt cuts' present, in that order, the average numbers for Lagrangian cardinality cuts, feasibility cuts, and optimality cuts. As shown in Table 5.2, the largest numbers of added cuts correspond to instances in RII, which explains the gains in computational times per node for that group, particularly when 'Lag cuts' are disabled.

In terms of optimality gaps, again a quite small variation is perceived. However, one can notice that the absence of extra cuts in the dual program can be beneficial for the algorithm. As shown in subtables 'No cuts' and 'All cuts not in Dual', the optimality gaps of the hardest instances in C and RII reduced when only cover inequalities were added to the dual program. Actually, the addition of too many cuts to the dual program may excessively penalize the arcs while solving the Lagrangian subproblem (reduced arc costs more negative), which results in the deterioration of bound values (see section 3.4.1 for details on the Lagrangian subproblem).

| Canad instances | Avg Gap (%) | Min Gap (%) | Max Gap (%) | Avg Time (s) | Avg Nodes | Avg Time/Node |
|---|---|---|---|---|---|---|
| **All cuts** | | | | | | |
| C | 0.57 | 0.00 | 6.43 | 19708.7 | 12357 | 1.59 |
| RI | 0.00 | 0.00 | 0.00 | 10.2 | 74 | 0.14 |
| RII | 0.18 | 0.00 | 1.50 | 12633.4 | 14737 | 0.86 |
| **(-) Opt cuts** | | | | | | |
| C | 0.57 | 0.00 | 6.43 | 19670.1 | 12382 | 1.59 |
| RI | 0.00 | 0.00 | 0.00 | 10.1 | 75 | 0.14 |
| RII | 0.18 | 0.00 | 1.49 | 12627.7 | 14755 | 0.86 |
| **(-) Lag cuts** | | | | | | |
| C | 0.57 | 0.00 | 6.43 | 19573.6 | 12256 | 1.60 |
| RI | 0.00 | 0.00 | 0.00 | 10.3 | 73 | 0.14 |
| RII | 0.18 | 0.00 | 1.49 | 12635.6 | 15002 | 0.84 |
| **(-) Feas cuts** | | | | | | |
| C | 0.57 | 0.00 | 6.43 | 19660.1 | 12442 | 1.58 |
| RI | 0.00 | 0.00 | 0.00 | 10.2 | 75 | 0.14 |
| RII | 0.18 | 0.00 | 1.49 | 12611.2 | 14609 | 0.86 |
| **No cuts** | | | | | | |
| C | 0.57 | 0.00 | 6.59 | 19627.6 | 12372 | 1.59 |
| RI | 0.00 | 0.00 | 0.00 | 10.5 | 77 | 0.14 |
| RII | 0.17 | 0.00 | 1.45 | 12823.8 | 16884 | 0.76 |
| **All cuts not in Dual** | | | | | | |
| C | 0.57 | 0.00 | 6.42 | 19793.3 | 12506 | 1.58 |
| RI | 0.00 | 0.00 | 0.00 | 10.1 | 76 | 0.13 |
| RII | 0.18 | 0.00 | 1.46 | 12528.2 | 14682 | 0.85 |

Table 5.1: Average results for different cut configurations

| Canad instances | Lag cuts | Feas cuts | Opt cuts |
|---|---|---|---|
| C | 8021 | 6 | 429 |
| R-I | 38 | 5 | 7 |
| R-II | 18413 | 16 | 477 |

Table 5.2: Average total number of separated cuts

Comparing the 'No cuts', and 'All cuts not in Dual' configurations, one can see that, although it might be better not to dualize extra cuts other than covers, their separation for constraint propagation purposes can be fruitful for hard instances. For example, in the Canad-C group, the 'Max Gap' was 6.59, while with constraint propagation, it reduced to 6.42 for practically the same computational time.

In fact, the gains with constraint propagation were quite good considering the number of times it managed to fix variables. Figure 5.1 provides a chart showing the efficiency of each variable fixing procedure. Given by percentages, the bars correspond to the average number of nodes where the respective fixing proce-

dure managed to fix variables, w.r.t the total average number of nodes processed. Specifically for the LP-based fixing and the 'pruning heuristic' (Step 4a and 4b of the algorithm in section 5.7), the percentages were computed w.r.t the number of times they were performed.

Even though the constraint propagation was effective in less than 10% of the nodes in all groups of instances, it proved to be worthful by the results in Table 5.1. However, a tighter limit on the number of separated cuts must be imposed, taking profit of variable fixing without increasing too much the times spent on each node.

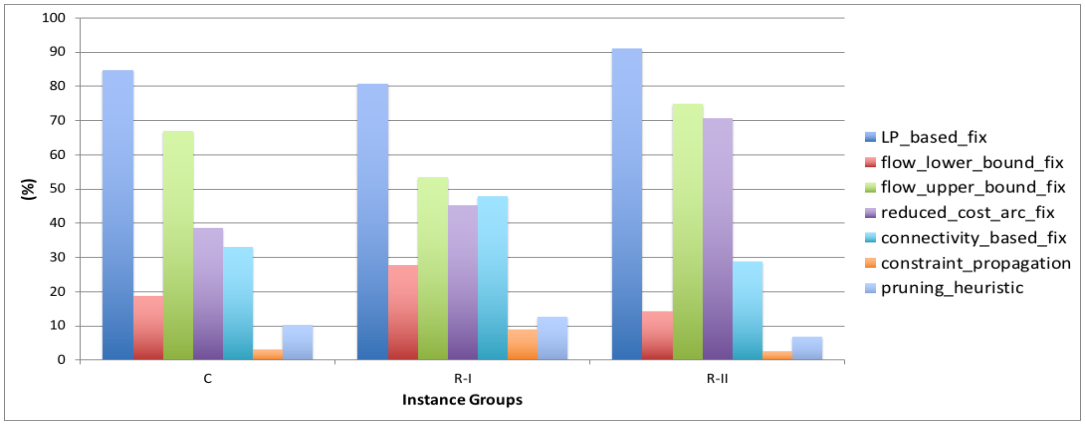



Figure 5.1: Variable fixing and pruning efficiency

As one can conclude, most of the fixed arcs were obtained by reduced cost fixing and by LP-based fixing. In fact, the LP-based fixing was performed in 2.79%, 27,05%, and 2,70% of the total number of processed nodes, respectively for instances in Canad-C, Canad-RI, and Canad-RII. However, it was very effective in fixing variables each time it was run. We stress that we do not count the number of variables fixed per time, so the impact of LP-based fixing and reduced cost fixing might be greater. The same can be told for constraint propagation.

Finally, as shown by the chart in Figure 5.1, the flow upper bound fixing procedure was the most effective in sharpening flow bounds, followed by connectivity-based fixing. Nevertheless, the flow lower bound fixing was quite effective considering that it depends on the number of opened arcs. In its turn, the 'pruning heuristic' was worthless given the amount of computational effort demanded.

The next subsection presents the results obtained with the best configuration according to the previous discussion. That means Step 4b is not performed, and the condition in Step 4a was restricted to $A_* = \emptyset$. Moreover, all cuts are separated but only cover inequalities are added to the dual programs. The other steps of the algorithm are performed as stated in section 5.7.

### 5.8.2 The algorithm performances

In this subsection, we report the optimality gaps obtained for each instance, with the Volume-based Branch-and-Cut algorithm, identified as VA-B&C. The first results are shown for benchmark instances Canad-N. A comparison with Cplex performance is provided. Both algorithms were let to run for 24 hours, and the initial upper bound was set to the best-known solution value.

Table 5.3 presents those results. The column '$\bar{Z}_0$' indicates the initial upper bound for each instance, while columns 'Gap (%)' show the final optimality gap provided by each algorithm, in percentages. We follow the measure adopted by Cplex, which is

$$\text{Gap} = (\bar{Z} - z_l)/\bar{Z} * 100$$

where $\bar{Z}$ is the final upper bound, and $z_l$ the final lower bound. The total processed nodes are given in columns 'Nodes', followed by the computational times in seconds. The columns 'UB' report the relative differences between the initial and the final upper bounds: $(\bar{Z} - \bar{Z}_0)/\bar{Z} * 100$. In the final row, the averages are shown.

| | | Cplex | | | | VA-B&C | | | |
|---|---|---|---|---|---|---|---|---|---|
| Instance | $\bar{Z}_0$ | Gap (%) | UB | Nodes | Time (s) | Gap (%) | UB | Nodes | Time (s) |
| pN01 | 2122239 | 0.00 | 0.00 | 447 | 881 | 0.00 | 0.00 | 1311 | 3131 |
| pN02 | 1664693 | 0.00 | 0.00 | 151 | 80 | 0.00 | 0.00 | 182 | 263 |
| pN03 | 2139957 | 0.00 | 0.00 | 196 | 519 | 0.00 | 0.00 | 135 | 275 |
| pN04 | 2840339 | 0.00 | 0.00 | 9092 | 33823 | 0.00 | 0.00 | 6068 | 14635 |
| pN05 | 125255 | 3.27 | -0.08 | 2901 | 86400 | **1.49** | 0.00 | 21152 | 86602 |
| pN06 | 81932 | 3.48 | 0.00 | 5339 | 86400 | **2.15** | 0.00 | 31896 | 86836 |
| pN07 | 120638 | 4.33 | -0.24 | 3101 | 86400 | **2.26** | 0.00 | 24141 | 86831 |
| pN08 | 83565 | 3.07 | -0.11 | 6138 | 86400 | **1.68** | 0.00 | 29576 | 86886 |
| pN09 | 374989 | 4.92 | 0.00 | 458 | 86400 | **2.77** | 0.00 | 7981 | 86581 |
| pN10 | 603615 | 4.58 | 0.00 | 423 | 86400 | **2.68** | 0.00 | 6525 | 86478 |
| pN11 | 473897 | 2.38 | -0.18 | 651 | 86400 | **1.69** | 0.00 | 6755 | 86464 |
| pN12 | 828747 | 3.57 | 0.00 | 530 | 86400 | **2.65** | 0.00 | 6019 | 86522 |
| pN13 | 354007 | 3.22 | 0.00 | 29 | 86401 | **2.07** | 0.00 | 4754 | 86542 |
| pN14 | 416131 | 2.93 | -0.01 | 11 | 86401 | **1.91** | 0.00 | 4033 | 86511 |
| pN15 | 378156 | 2.44 | 0.00 | 84 | 86401 | **1.71** | 0.00 | 5033 | 86514 |
| pN16 | 598857 | 3.70 | -0.15 | 22 | 86401 | **2.87** | 0.00 | 4113 | 86508 |
| pN17 | 407057 | 5.27 | 0.00 | 3321 | 86400 | **4.32** | 0.00 | 31938 | 86649 |
| pN18 | 159099 | 2.59 | -0.35 | 5410 | 86400 | 2.79 | 0.00 | 48881 | 86823 |
| pN19 | 358984 | 6.43 | 0.00 | 2192 | 86400 | **4.52** | 0.00 | 33629 | 86670 |
| pN20 | 149918 | 2.19 | 0.00 | 5007 | 86400 | **1.40** | 0.00 | 40050 | 86646 |
| pN21 | 314877 | 8.11 | -0.68 | 346 | 86400 | **7.19** | 0.00 | 10719 | 86552 |
| pN22 | 555984 | 8.54 | -0.63 | 203 | 86400 | **7.60** | 0.00 | 9767 | 86550 |
| pN23 | 435738 | 7.26 | 0.00 | 586 | 86400 | **6.51** | 0.00 | 10334 | 86550 |
| pN24 | 804561 | 7.05 | -0.05 | 549 | 86400 | **6.39** | 0.00 | 9536 | 86540 |
| pN25 | 1685350 | 3.40 | 0.00 | 117 | 86401 | **3.03** | 0.00 | 6572 | 86537 |
| pN26 | 2509613 | 3.19 | 0.00 | 51 | 86401 | **2.90** | 0.00 | 5172 | 86495 |
| pN27 | 1545679 | 2.82 | 0.00 | 54 | 86401 | **2.30** | 0.00 | 6463 | 86502 |
| pN28 | 2246542 | 2.92 | 0.00 | 45 | 86401 | **2.48** | 0.00 | 5445 | 86490 |
| pN29 | 972311 | 4.92 | -0.01 | 0 | 86403 | **3.95** | 0.00 | 2438 | 86507 |
| pN30 | 857118 | 3.55 | 0.00 | 0 | 86402 | **2.94** | 0.00 | 2344 | 86489 |
| pN31 | 572552 | 4.07 | 0.00 | 0 | 86403 | **3.39** | 0.00 | 2981 | 86544 |
| pN32 | 926792 | 3.71 | -0.38 | 0 | 86402 | **3.33** | 0.00 | 2407 | 86515 |
| pN33 | 2129079 | 6.57 | -0.74 | 1311 | 86400 | 8.63 | 0.00 | 9640 | 86482 |
| pN34 | 1061480 | 5.77 | -0.63 | 1468 | 86400 | 8.05 | 0.00 | 11300 | 86462 |
| pN35 | 2519584 | 5.96 | -0.56 | 2016 | 86400 | 7.80 | 0.00 | 9737 | 86481 |
| pN36 | 1223827 | 5.44 | -0.80 | 2189 | 86400 | 7.59 | -0.06 | 10904 | 86466 |
| pN37 | 1189320 | 10.06 | 0.00 | 0 | 86401 | **9.00** | 0.00 | 4408 | 86518 |
| pN38 | 182726 | 7.93 | 0.00 | 14 | 86401 | **7.38** | 0.00 | 5867 | 86536 |
| pN39 | 995193 | 13.65 | 0.00 | 0 | 86401 | **12.26** | 0.00 | 4361 | 86518 |
| pN40 | 100499 | 7.44 | 0.00 | 26 | 86401 | **6.82** | 0.00 | 6799 | 86556 |
| pN41 | 273748 | 5.37 | 0.00 | 0 | 86402 | **5.04** | 0.00 | 2672 | 86507 |
| pN42 | 1605884 | 10.18 | 0.00 | 0 | 86403 | **9.28** | 0.00 | 2085 | 86493 |
| pN43 | 921440 | 7.28 | -0.58 | 0 | 86403 | **7.25** | 0.00 | 2253 | 86497 |
| pN44 | 1983636 | 9.26 | 0.00 | 0 | 86403 | **8.72** | 0.00 | 1982 | 86496 |
| pN45 | 1070118 | 6.31 | 0.00 | 0 | 86406 | **5.93** | 0.00 | 1192 | 86528 |
| pN46 | 580398 | 4.04 | 0.00 | 0 | 86406 | **3.84** | 0.00 | 1063 | 86525 |
| pN47 | 1298479 | 4.71 | 0.00 | 0 | 86406 | **4.47** | 0.00 | 798 | 86517 |
| pN48 | 2236171 | 5.11 | 0.00 | 0 | 86406 | **4.88** | 0.00 | 751 | 86458 |
| | Avg | 4.85 | -0.13 | 1135 | 79937 | 4.33 | 0.00 | 9670 | 79722 |

Table 5.3: Optimality gaps for benchmark instances Canad-N

As one can notice in Table 5.3, except for instances 'pN01' to 'pN04', both algorithms have run until the time limit, with the VA-B&C algorithm providing the best gaps in general. Given in boldface, for more than 80% of the instances (39/48), the proposed method provided better optimality gaps. In addition, for the first four instances, both methods have proven optimality, and for two of them, VA-B&C was faster.

For the largest instances, Cplex has struggled with the size of LPs, spending

the whole computational time in the root node running its cutting plane algorithm. Moreover, much effort is applied by Cplex in the first node trying to find a better feasible solution. Indeed, the negative values in 'UB' indicate an improvement in upper bounds, which is achieved by Cplex for 17 instances, providing better upper bounds on average.

We now report, in Tables 5.4 to 5.7, the results obtained with benchmark instances Canad-C and Canad-R. The performances were compared to the results presented in [50] for Cplex and the LP-based Branch-and-Cut algorithm (B&C) proposed in that paper. We stress that the authors did not consider Canad-N instances. Like in precedent tables, the initial upper bound, the optimality gaps, and the times in seconds are provided, along with the number of processed nodes for VA-B&C (see [50] for the number of processed nodes for the other methods). No improvements in upper bounds were perceived in any case, so columns 'UB' were omitted. Moreover, the last row of each table presents averages, and the gap values in boldface indicate that VA-B&C provided a better optimality gap for that instance.

Since the computational times reported in [50] are related to a different processor (Intel Xeon ES-2609 v2 @ 2,50 GHz, according to the authors), we present normalized times, computed as explained at the beginning of this section. Given that all methods were run in single-threaded mode, we use the PassMark Single Thread Rating of each CPU, which corresponds to 1298 for the CPU used in [50], and 1915, for the CPU used in this experiment. In that sense, the times reported by the authors were multiplied by 0.68, and the time limit was set to 6.8 hours, which is equivalent to the limit of 10 hours set in their work.

First analyzing the results obtained for Canad-C instances in Table 5.4, it can be seen that the computational times were similar on average, and the VA-B&C provided the best average optimality gap. For 29/43 instances, VA-B&C presented the best gaps, while for 10 of those instances, the given gap was strictly better. Moreover, optimality have been proved for instances 'c38' and 'c74', whose optimality gaps were strictly positive before.

In general, for easier instances, Cplex and B&C perform better than VA-B&C, providing the best gaps in less time. That is confirmed by the Canad-R results presented in Tables 5.5 to 5.7.

| | | Cplex | | B&C | | VA-B&C | | |
|---|---|---|---|---|---|---|---|---|
| Instance | $\bar{Z}_0$ | Gap (%) | Time (s) | Gap (%) | Time (s) | Gap (%) | Nodes | Time (s) |
| c33 | 423848 | 0.00 | 0 | 0.00 | 0 | 0.00 | 25 | 3 |
| c35 | 371475 | 0.00 | 0 | 0.00 | 3 | 0.00 | 350 | 64 |
| c36 | 643036 | 0.00 | 2 | 0.00 | 224 | 0.00 | 9166 | 1288 |
| c37 | 94213 | 1.98 | 24480 | 0.00 | 9612 | 0.00 | 3692 | 11473 |
| c38 | 137642 | 2.84 | 24480 | 0.39 | 24482 | **0.00** | 5369 | 19975 |
| c39 | 97914 | 0.00 | 2424 | 0.00 | 4308 | 0.00 | 1316 | 3165 |
| c40 | 135863 | 2.55 | 24480 | 0.69 | 24481 | **0.32** | 9054 | 24530 |
| c41 | 429398 | 0.00 | 0 | 0.00 | 0 | 0.00 | 14 | 4 |
| c42 | 586077 | 0.00 | 2 | 0.00 | 7 | 0.00 | 267 | 67 |
| c43 | 464509 | 0.00 | 1 | 0.00 | 9 | 0.00 | 722 | 134 |
| c44 | 604198 | 0.00 | 1 | 0.00 | 2 | 0.00 | 126 | 31 |
| c45 | 74811 | 1.92 | 24480 | 0.36 | 24481 | 0.40 | 9162 | 24523 |
| c46 | 115525 | 3.46 | 24480 | 1.56 | 24484 | **0.99** | 8733 | 24564 |
| c47 | 74991 | 0.00 | 2109 | 0.00 | 1671 | 0.00 | 2029 | 4587 |
| c48 | 107102 | 2.46 | 24480 | 0.64 | 24481 | **0.53** | 8531 | 24531 |
| c49 | 53958 | 0.62 | 24480 | 0.00 | 8050 | 0.11 | 29704 | 24569 |
| c50 | 93922 | 3.36 | 24480 | 0.82 | 24481 | **0.54** | 12162 | 24536 |
| c51 | 52046 | 0.60 | 24480 | 0.15 | 24480 | 0.51 | 13413 | 24585 |
| c52 | 97098 | 2.49 | 24480 | 1.39 | 24480 | **1.13** | 12986 | 24578 |
| c53 | 112774 | 0.66 | 24480 | 0.15 | 24485 | 0.43 | 946 | 24495 |
| c54 | 149094 | 1.31 | 24480 | 0.82 | 24497 | 0.97 | 199 | 24547 |
| c55 | 114640 | 0.30 | 24480 | 0.04 | 24482 | 0.25 | 909 | 24509 |
| c56 | 152414 | 1.53 | 24480 | 1.06 | 24496 | **0.98** | 736 | 24558 |
| c57 | 47603 | 0.00 | 21 | 0.00 | 25 | 0.00 | 108 | 170 |
| c58 | 59958 | 2.50 | 24480 | 0.74 | 24480 | **0.64** | 8483 | 24538 |
| c59 | 45872 | 0.91 | 24480 | 0.33 | 24480 | 0.56 | 6534 | 24545 |
| c60 | 54904 | 1.35 | 24480 | 0.61 | 24481 | 0.84 | 7011 | 24550 |
| c61 | 97845 | 1.08 | 24480 | 0.59 | 24489 | 0.66 | 445 | 24503 |
| c62 | 133976 | 2.64 | 24480 | 1.92 | 24496 | **1.02** | 805 | 24818 |
| c63 | 95250 | 1.07 | 24480 | 0.69 | 24485 | 0.94 | 172 | 24539 |
| c64 | 129869 | 1.53 | 24480 | 0.92 | 24494 | 0.99 | 500 | 24863 |
| c65 | 14941 | 0.00 | 3 | 0.00 | 5 | 0.00 | 117 | 11 |
| c66 | 49899 | 0.00 | 1 | 0.00 | 7 | 0.00 | 276 | 21 |
| c67 | 14712 | 0.00 | 0 | 0.00 | 0 | 0.00 | 2 | 0 |
| c68 | 37055 | 0.00 | 21 | 0.00 | 104 | 0.00 | 604 | 165 |
| c69 | 85530 | 0.00 | 4 | 0.00 | 41 | 0.00 | 1387 | 168 |
| c70 | 365272 | 0.00 | 0 | 0.00 | 0 | 0.00 | 1070 | 54 |
| c71 | 23949 | 0.00 | 96 | 0.00 | 5672 | 0.00 | 1542 | 1498 |
| c72 | 63753 | 5.03 | 24480 | 7.13 | 24480 | 6.58 | 91888 | 24592 |
| c73 | 28423 | 0.00 | 0 | 0.00 | 48 | 0.00 | 1386 | 1422 |
| c74 | 49018 | 2.31 | 24480 | 0.09 | 24481 | **0.00** | 3515 | 11579 |
| c75 | 136621 | 5.22 | 24480 | 7.09 | 24480 | 6.52 | 64214 | 24614 |
| c76 | 384802 | 0.00 | 20 | 0.35 | 24480 | 0.55 | 50634 | 24672 |
| | Avg | 1.16 | 13773 | 0.66 | 13789 | 0.62 | 8611.72 | 13875 |

Table 5.4: Optimality gaps for benchmark instances Canad-C

| | | Cplex | | B&C | | VA-B&C | | |
|---|---|---|---|---|---|---|---|---|
| Instance | $\bar{Z}_0$ | Gap (%) | Time (s) | Gap (%) | Time (s) | Gap (%) | Nodes | Time (s) |
| r01.1 | 74079 | 0.00 | 0 | 0.00 | 0 | 0.00 | 1 | 0 |
| r01.2 | 92403 | 0.00 | 0 | 0.00 | 0 | 0.00 | 1 | 0 |
| r01.3 | 115304 | 0.00 | 0 | 0.00 | 0 | 0.00 | 3 | 0 |
| r01.4 | 84908 | 0.00 | 0 | 0.00 | 0 | 0.00 | 12 | 0 |
| r01.5 | 113036 | 0.00 | 0 | 0.00 | 0 | 0.00 | 12 | 0 |
| r01.6 | 147599 | 0.00 | 0 | 0.00 | 0 | 0.00 | 26 | 1 |
| r02.1 | 232239 | 0.00 | 0 | 0.00 | 0 | 0.00 | 11 | 0 |
| r02.2 | 322453 | 0.00 | 0 | 0.00 | 0 | 0.00 | 15 | 0 |
| r02.3 | 419503 | 0.00 | 0 | 0.00 | 0 | 0.00 | 4 | 0 |
| r02.4 | 316437 | 0.00 | 0 | 0.00 | 0 | 0.00 | 10 | 0 |
| r02.5 | 431250 | 0.00 | 0 | 0.00 | 0 | 0.00 | 18 | 0 |
| r02.6 | 559578 | 0.00 | 0 | 0.00 | 0 | 0.00 | 7 | 0 |
| r03.1 | 484830 | 0.00 | 0 | 0.00 | 0 | 0.00 | 3 | 0 |
| r03.2 | 703362 | 0.00 | 0 | 0.00 | 0 | 0.00 | 4 | 1 |
| r03.3 | 944990 | 0.00 | 0 | 0.00 | 0 | 0.00 | 5 | 0 |
| r03.4 | 704247 | 0.00 | 0 | 0.00 | 0 | 0.00 | 27 | 1 |
| r03.5 | 932897 | 0.00 | 0 | 0.00 | 0 | 0.00 | 7 | 0 |
| r03.6 | 1188638 | 0.00 | 0 | 0.00 | 0 | 0.00 | 11 | 1 |
| r04.1 | 31730 | 0.00 | 0 | 0.00 | 0 | 0.00 | 1 | 0 |
| r04.2 | 48920 | 0.00 | 0 | 0.00 | 0 | 0.00 | 1 | 0 |
| r04.3 | 63767 | 0.00 | 0 | 0.00 | 0 | 0.00 | 1 | 0 |
| r04.4 | 33740 | 0.00 | 0 | 0.00 | 0 | 0.00 | 11 | 0 |
| r04.5 | 53790 | 0.00 | 0 | 0.00 | 0 | 0.00 | 7 | 0 |
| r04.6 | 74030 | 0.00 | 0 | 0.00 | 0 | 0.00 | 8 | 0 |
| r04.7 | 68292 | 0.00 | 0 | 0.00 | 0 | 0.00 | 144 | 3 |
| r04.8 | 113004 | 0.00 | 0 | 0.00 | 0 | 0.00 | 54 | 2 |
| r04.9 | 163208 | 0.00 | 0 | 0.00 | 0 | 0.00 | 23 | 1 |
| r05.1 | 123003 | 0.00 | 0 | 0.00 | 0 | 0.00 | 2 | 0 |
| r05.2 | 170060 | 0.00 | 0 | 0.00 | 0 | 0.00 | 2 | 0 |
| r05.3 | 221486 | 0.00 | 0 | 0.00 | 1 | 0.00 | 7 | 0 |
| r05.4 | 131608 | 0.00 | 0 | 0.00 | 0 | 0.00 | 19 | 1 |
| r05.5 | 204157 | 0.00 | 0 | 0.00 | 1 | 0.00 | 18 | 2 |
| r05.6 | 286524 | 0.00 | 1 | 0.00 | 4 | 0.00 | 63 | 8 |
| r05.7 | 278372 | 0.00 | 0 | 0.00 | 0 | 0.00 | 93 | 3 |
| r05.8 | 445810 | 0.00 | 0 | 0.00 | 0 | 0.00 | 18 | 1 |
| r05.9 | 625879 | 0.00 | 0 | 0.00 | 0 | 0.00 | 65 | 2 |
| r06.1 | 245936 | 0.00 | 0 | 0.00 | 0 | 0.00 | 9 | 1 |
| r06.2 | 401685 | 0.00 | 2 | 0.00 | 2 | 0.00 | 32 | 6 |
| r06.3 | 559477 | 0.00 | 5 | 0.00 | 12 | 0.00 | 58 | 15 |
| r06.4 | 286682 | 0.00 | 0 | 0.00 | 1 | 0.00 | 147 | 14 |
| r06.5 | 498266 | 0.00 | 8 | 0.00 | 29 | 0.00 | 339 | 90 |
| r06.6 | 734414 | 0.00 | 108 | 0.00 | 88 | 0.00 | 495 | 128 |
| r06.7 | 682921 | 0.00 | 0 | 0.00 | 0 | 0.00 | 35 | 4 |
| r06.8 | 1030479 | 0.00 | 0 | 0.00 | 0 | 0.00 | 16 | 2 |
| r06.9 | 423316 | 0.00 | 1 | 0.00 | 1 | 0.00 | 34 | 6 |
| r07.1 | 32807 | 0.00 | 0 | 0.00 | 0 | 0.00 | 1 | 0 |
| r07.2 | 47252 | 0.00 | 0 | 0.00 | 0 | 0.00 | 1 | 0 |
| r07.3 | 62962 | 0.00 | 0 | 0.00 | 0 | 0.00 | 2 | 0 |
| r07.4 | 37432 | 0.00 | 0 | 0.00 | 0 | 0.00 | 79 | 2 |
| r07.5 | 56475 | 0.00 | 0 | 0.00 | 1 | 0.00 | 83 | 3 |
| r07.6 | 77249 | 0.00 | 0 | 0.00 | 3 | 0.00 | 121 | 4 |
| r07.7 | 59947 | 0.00 | 0 | 0.00 | 1 | 0.00 | 157 | 8 |
| r07.8 | 99194 | 0.00 | 0 | 0.00 | 2 | 0.00 | 107 | 9 |
| r07.9 | 141692 | 0.00 | 1 | 0.00 | 5 | 0.00 | 184 | 15 |
| | Avg | 0.00 | 2 | 0.00 | 3 | 0.00 | 48 | 6 |

Table 5.5: Optimality gaps for benchmark instances Canad-R, part I

| | | Cplex | | B&C | | VA-B&C | | |
|---|---|---|---|---|---|---|---|---|
| Instance | $\bar{Z}_0$ | Gap (%) | Time (s) | Gap (%) | Time (s) | Gap (%) | Nodes | Time (s) |
| r08.1 | 102531 | 0.00 | 0 | 0.00 | 0 | 0.00 | 7 | 0 |
| r08.2 | 143894 | 0.00 | 0 | 0.00 | 0 | 0.00 | 2 | 0 |
| r08.3 | 182793 | 0.00 | 0 | 0.00 | 0 | 0.00 | 2 | 0 |
| r08.4 | 109325 | 0.00 | 0 | 0.00 | 0 | 0.00 | 13 | 1 |
| r08.5 | 157047 | 0.00 | 0 | 0.00 | 0 | 0.00 | 9 | 1 |
| r08.6 | 207540 | 0.00 | 1 | 0.00 | 1 | 0.00 | 36 | 3 |
| r08.7 | 154160 | 0.00 | 1 | 0.00 | 6 | 0.00 | 256 | 26 |
| r08.8 | 274867 | 0.00 | 4 | 0.00 | 39 | 0.00 | 374 | 54 |
| r08.9 | 415793 | 0.00 | 6 | 0.00 | 35 | 0.00 | 426 | 61 |
| r09.1 | 171512 | 0.00 | 0 | 0.00 | 0 | 0.00 | 17 | 1 |
| r09.2 | 296712 | 0.00 | 1 | 0.00 | 1 | 0.00 | 20 | 3 |
| r09.3 | 424266 | 0.00 | 12 | 0.00 | 6 | 0.00 | 49 | 15 |
| r09.4 | 192736 | 0.00 | 0 | 0.00 | 1 | 0.00 | 33 | 3 |
| r09.5 | 357318 | 0.00 | 3 | 0.00 | 6 | 0.00 | 98 | 14 |
| r09.6 | 522187 | 0.00 | 29 | 0.00 | 34 | 0.00 | 313 | 79 |
| r09.7 | 345057 | 0.00 | 0 | 0.00 | 2 | 0.00 | 207 | 22 |
| r09.8 | 646579 | 0.00 | 1 | 0.00 | 3 | 0.00 | 151 | 16 |
| r09.9 | 951136 | 0.00 | 5 | 0.00 | 14 | 0.00 | 883 | 88 |
| r10.1 | 200087 | 0.00 | 0 | 0.00 | 0 | 0.00 | 13 | 1 |
| r10.2 | 346814 | 0.00 | 22 | 0.00 | 8 | 0.00 | 66 | 19 |
| r10.3 | 488015 | 0.00 | 22 | 0.00 | 10 | 0.00 | 30 | 14 |
| r10.4 | 229196 | 0.00 | 2 | 0.00 | 28 | 0.00 | 715 | 106 |
| r10.5 | 411664 | 0.00 | 10620 | 0.00 | 324 | 0.00 | 1082 | 442 |
| r10.6 | 609103 | 1.27 | 24480 | 0.00 | 1486 | 0.00 | 1958 | 937 |
| r10.7 | 486895 | 0.00 | 3 | 0.00 | 63 | 0.00 | 3261 | 433 |
| r10.8 | 951056 | 0.00 | 7 | 0.00 | 58 | 0.00 | 2789 | 308 |
| r10.9 | 1421746 | 0.00 | 9 | 0.00 | 71 | 0.00 | 48162 | 3711 |
| r11.1 | 714431 | 0.00 | 5 | 0.00 | 5 | 0.00 | 234 | 65 |
| r11.2 | 1263713 | 1.38 | 24480 | 0.00 | 913 | 0.00 | 2218 | 1323 |
| r11.3 | 1843610 | 2.34 | 24480 | 0.00 | 3418 | 0.00 | 2379 | 2167 |
| r11.4 | 870451 | 0.00 | 46 | 0.00 | 201 | 0.00 | 3739 | 1218 |
| r11.5 | 1623640 | 0.00 | 9521 | 0.00 | 703 | 0.00 | 2144 | 1091 |
| r11.6 | 2414060 | 0.00 | 8529 | 0.00 | 1715 | 0.00 | 2425 | 1552 |
| r11.7 | 2294912 | 0.00 | 1 | 0.00 | 1 | 0.00 | 375 | 123 |
| r11.8 | 3507100 | 0.00 | 1 | 0.00 | 2 | 0.00 | 26400 | 6003 |
| r11.9 | 4579353 | 0.00 | 1 | 0.00 | 1 | 0.00 | 262 | 55 |
| r12.1 | 1639443 | 0.00 | 64 | 0.00 | 261 | 0.00 | 542 | 435 |
| r12.2 | 3396050 | 1.78 | 24480 | 0.00 | 13166 | 0.00 | 5972 | 8804 |
| r12.3 | 5228710 | 2.02 | 24480 | 0.00 | 15391 | 0.00 | 2865 | 5255 |
| r12.4 | 2303557 | 0.00 | 18 | 0.00 | 68 | 0.00 | 154 | 183 |
| r12.5 | 4669799 | 0.00 | 32 | 0.00 | 50 | 0.00 | 119 | 127 |
| r12.6 | 7100019 | 0.00 | 19 | 0.00 | 20 | 0.00 | 171 | 124 |
| r12.7 | 7635270 | 0.00 | 1 | 0.00 | 1 | 0.00 | 79 | 69 |
| r12.8 | 10067742 | 0.00 | 1 | 0.00 | 1 | 0.00 | 8 | 9 |
| r12.9 | 11967768 | 0.00 | 1 | 0.00 | 0 | 0.00 | 1 | 2 |
| r13.1 | 142947 | 0.00 | 0 | 0.00 | 0 | 0.00 | 6 | 2 |
| r13.2 | 263800 | 0.00 | 38 | 0.00 | 37 | 0.00 | 128 | 87 |
| r13.3 | 365836 | 0.00 | 87 | 0.00 | 76 | 0.00 | 135 | 152 |
| r13.4 | 150977 | 0.00 | 1 | 0.00 | 9 | 0.00 | 434 | 73 |
| r13.5 | 282682 | 1.45 | 24480 | 0.00 | 261 | 0.00 | 973 | 517 |
| r13.6 | 406789 | 3.30 | 24480 | 0.00 | 791 | 0.00 | 1370 | 1217 |
| r13.7 | 208088 | 0.00 | 13311 | 0.00 | 2136 | 0.00 | 92623 | 16025 |
| r13.8 | 444826 | 1.76 | 24480 | 0.00 | 12775 | 0.01 | 108616 | 24635 |
| r13.9 | 697966 | 2.25 | 24480 | 0.39 | 24480 | 0.71 | 71223 | 24595 |
| | Avg | 0.32 | 4866 | 0.01 | 1457 | 0.01 | 7159 | 1894 |

Table 5.6: Optimality gaps for benchmark instances Canad-R, part II

| | | Cplex | | B&C | | VA-B&C | | |
|---|---|---|---|---|---|---|---|---|
| Instance | $\bar{Z}_0$ | Gap (%) | Time (s) | Gap (%) | Time (s) | Gap (%) | Nodes | Time (s) |
| r14.1 | 403414 | 0.00 | 7 | 0.00 | 9 | 0.00 | 96 | 34 |
| r14.2 | 749503 | 2.90 | 24480 | 0.00 | 2171 | 0.00 | 2314 | 3697 |
| r14.3 | 1063097 | 3.66 | 24480 | 0.00 | 8106 | 0.00 | 2053 | 5069 |
| r14.4 | 437607 | 0.00 | 10 | 0.00 | 32 | 0.00 | 551 | 229 |
| r14.5 | 849163 | 2.61 | 24480 | 0.00 | 7953 | 0.00 | 10021 | 7765 |
| r14.6 | 1214608 | 2.70 | 24480 | 0.00 | 7215 | 0.00 | 2357 | 4194 |
| r14.7 | 668216 | 0.56 | 24480 | 0.08 | 24480 | 0.34 | 39396 | 24600 |
| r14.8 | 1613429 | 1.98 | 24480 | 0.73 | 24480 | 0.99 | 27809 | 24538 |
| r14.9 | 2602689 | 0.51 | 24480 | 0.07 | 24480 | **0.00** | 39848 | 21832 |
| r15.1 | 1000787 | 0.00 | 784 | 0.00 | 112 | 0.00 | 615 | 758 |
| r15.2 | 1966206 | 2.97 | 24480 | 0.55 | 24481 | **0.23** | 10611 | 24537 |
| r15.3 | 2876628 | 4.45 | 24480 | 2.01 | 24485 | **0.91** | 12073 | 24529 |
| r15.4 | 1148604 | 0.44 | 24480 | 0.00 | 6706 | 0.00 | 23595 | 22880 |
| r15.5 | 2476246 | 2.79 | 24480 | 1.11 | 24481 | **0.77** | 11657 | 24530 |
| r15.6 | 3826956 | 3.24 | 24480 | 1.60 | 24484 | **1.18** | 11354 | 24510 |
| r15.7 | 2297919 | 0.00 | 13087 | 0.00 | 10341 | 0.34 | 18737 | 24513 |
| r15.8 | 5573413 | 0.00 | 48 | 0.00 | 443 | 0.07 | 18767 | 24520 |
| r15.9 | 8696932 | 0.00 | 7 | 0.00 | 14 | 0.00 | 178 | 237 |
| r16.1 | 136161 | 0.00 | 1 | 0.00 | 0 | 0.00 | 1 | 0 |
| r16.2 | 239500 | 0.00 | 161 | 0.00 | 163 | 0.00 | 308 | 400 |
| r16.3 | 325671 | 0.00 | 220 | 0.00 | 393 | 0.00 | 365 | 575 |
| r16.4 | 138532 | 0.00 | 1 | 0.00 | 1 | 0.00 | 6 | 3 |
| r16.5 | 241801 | 0.00 | 23 | 0.00 | 22 | 0.00 | 22 | 28 |
| r16.6 | 337762 | 2.27 | 24480 | 0.00 | 173 | 0.00 | 103 | 139 |
| r16.7 | 169233 | 1.51 | 24480 | 0.15 | 24480 | 0.49 | 83242 | 24627 |
| r16.8 | 348167 | 3.23 | 24480 | 1.25 | 24480 | **1.24** | 63354 | 24603 |
| r16.9 | 529988 | 4.66 | 24480 | 2.06 | 24481 | **1.59** | 55833 | 24567 |
| r17.1 | 354138 | 0.00 | 6 | 0.00 | 4 | 0.00 | 17 | 14 |
| r17.2 | 645488 | 2.29 | 24480 | 0.00 | 3832 | 0.00 | 815 | 2869 |
| r17.3 | 910518 | 5.58 | 24480 | 0.00 | 24481 | 0.00 | 5482 | 16258 |
| r17.4 | 370590 | 0.00 | 35 | 0.00 | 97 | 0.00 | 750 | 407 |
| r17.5 | 706747 | 2.79 | 24480 | 0.00 | 5104 | 0.00 | 5044 | 8577 |
| r17.6 | 1019758 | 5.61 | 24480 | 1.00 | 24481 | **0.26** | 27802 | 24570 |
| r17.7 | 501635 | 0.85 | 24480 | 0.26 | 24480 | 0.52 | 28044 | 24575 |
| r17.8 | 1105083 | 2.36 | 24480 | 1.07 | 24480 | 1.13 | 23794 | 24560 |
| r17.9 | 1777763 | 3.17 | 24480 | 1.62 | 24481 | **1.52** | 18910 | 24520 |
| r18.1 | 828117 | 0.95 | 24480 | 0.00 | 3548 | 0.00 | 11248 | 17021 |
| r18.2 | 1533675 | 0.00 | 2974 | 0.00 | 3977 | 0.00 | 395 | 3442 |
| r18.3 | 2174276 | 3.78 | 24480 | 0.94 | 24483 | **0.44** | 2018 | 24523 |
| r18.4 | 919325 | 1.31 | 24480 | 0.24 | 24480 | 0.30 | 10712 | 24526 |
| r18.5 | 1823766 | 3.11 | 24480 | 1.13 | 24483 | **0.76** | 6877 | 24576 |
| r18.6 | 2701286 | 4.21 | 24480 | 2.40 | 24487 | **1.17** | 8762 | 24524 |
| r18.7 | 1477395 | 1.17 | 24480 | 0.78 | 24481 | 0.98 | 9966 | 24513 |
| r18.8 | 3887637 | 2.12 | 24480 | 0.53 | 24481 | 0.71 | 2789 | 24508 |
| r18.9 | 6361906 | 1.19 | 24480 | 0.23 | 24481 | 0.36 | 9100 | 24517 |
| | Avg | 1.80 | 17250 | 0.44 | 13311 | 0.36 | 13506 | 14587 |

Table 5.7: Optimality gaps for benchmark instances Canad-R, part III

Table 5.5 reports the results for the easiest instances. For that group, the VA-B&C is clearly outperformed by Cplex and B&C, in terms of times. Nevertheless, optimality is achieved in all cases by the three algorithms. For instances in Table 5.6, however, VA-B&C managed to be faster in some occasions like 'r12.2' and 'r12.3'. For this last group, B&C was more efficient, but VA-B&C did better than Cplex on average.

In Table 5.7, larger instances are considered. In this case, with slightly higher computational times, VA-B&C provided the best optimality gap on average. For

35/45 instances the proposed algorithm returned the best gaps, and for 12 of those instances, the gaps provided by VA-B&C were strictly better. Moreover, new optimality proof was obtained for 'r14.9'.

Finally, the results for the very large instances of groups A to E are presented in Tables 5.8 and 5.9. The instances F to H could not be tested due to limits in memory space. No comparison is made since Cplex could not provide any gaps for those instances within a time limit of 24 hours. Considering they are new, no literature results exist at this moment. Columns are identified as in precedent tables, and the averages results for each group are shown after its last instance.

As one can notice from the tables, instances in A and B are the most difficult, which was expected given the ratios of capacity tightness and fixed costs of those instances (see section 3.6.1 for details). Good quality gaps were obtained for instances C, D/20_001, D/14_001, and E, and optimality has been proved for instances C/02_001c, C/02_001e, and C/14_001e. Moreover, new better feasible solutions were found for 11 instances.

| Instance | $\bar{Z}_0$ | Gap (%) | UB | Nodes | Time (s) |
|---|---|---|---|---|---|
| A/08_05a | 143957028 | 17.08 | 0.00 | 434 | 86692 |
| A/08_05b | 142543732 | 16.21 | 0.00 | 436 | 86607 |
| A/08_05c | 144182939 | 16.93 | 0.00 | 418 | 86673 |
| A/08_05d | 497221 | 21.94 | 0.00 | 425 | 86708 |
| A/08_05e | 13923307 | 14.09 | 0.00 | 383 | 86694 |
| A/10_05a | 147903153 | 18.60 | 0.00 | 399 | 86596 |
| A/10_05b | 148076489 | 18.64 | 0.00 | 406 | 86661 |
| A/10_05c | 149823462 | 19.16 | 0.00 | 382 | 86645 |
| A/10_05d | 522125 | 24.84 | 0.00 | 378 | 86654 |
| A/10_05e | 14807132 | 17.72 | 0.00 | 404 | 86637 |
| A/14_05a | 158128681 | 21.72 | 0.00 | 362 | 86691 |
| A/14_05b | 157377772 | 21.24 | 0.00 | 361 | 86557 |
| A/14_05c | 164424149 | 23.85 | 0.00 | 348 | 86628 |
| A/14_05d | 551592 | 26.46 | 0.00 | 374 | 86713 |
| A/14_05e | 15575836 | 18.08 | 0.00 | 340 | 86671 |
| | Avg | 19.77 | 0.00 | 390 | 86655 |
| B/14_10a | 69846114 | 26.46 | 0.00 | 1575 | 86577 |
| B/14_10b | 65989759 | 25.28 | 0.00 | 1596 | 86559 |
| B/14_10c | 70551393 | 25.12 | 0.00 | 1583 | 86534 |
| B/14_10d | 75213876 | 27.91 | 0.00 | 1600 | 86588 |
| B/14_10e | 51353874 | 28.31 | 0.00 | 1640 | 86549 |
| B/10_10a | 65168024 | 26.37 | 0.00 | 1681 | 86552 |
| B/10_10b | 61131120 | 24.40 | 0.00 | 1664 | 86535 |
| B/10_10c | 67824166 | 27.02 | 0.00 | 1664 | 86522 |
| B/10_10d | 68211559 | 25.81 | 0.00 | 1671 | 86518 |
| B/10_10e | 46137950 | 25.33 | 0.00 | 1674 | 86516 |
| B/06_10a | 58116298 | 22.17 | 0.00 | 1730 | 86557 |
| B/06_10b | 53688788 | 17.74 | 0.00 | 1754 | 86557 |
| B/06_10c | 59078527 | 21.32 | 0.00 | 1736 | 86523 |
| B/06_10d | 60776805 | 21.59 | 0.00 | 1761 | 86645 |
| B/06_10e | 40429256 | 19.74 | 0.00 | 1746 | 86552 |
| | Avg | 24.30 | 0.00 | 1672 | 86552 |

Table 5.8: Optimality gaps for very large instances of groups A and B

| Instance | $\bar{Z}_0$ | Gap (%) | UB | Nodes | Time (s) |
|---|---|---|---|---|---|
| C/02_10a | 72920073 | 8.69 | 0.00 | 1051 | 86569 |
| C/02_10b | 45385926 | 3.44 | 0.00 | 575 | 86526 |
| C/02_10c | 32494545 | 8.46 | 0.00 | 1015 | 86581 |
| C/02_10d | 198264021 | 7.22 | 0.00 | 1024 | 86539 |
| C/02_10e | 811852234 | 17.99 | 0.00 | 1463 | 86563 |
| C/02_001a | 287761 | 0.01 | -0.06 | 4856 | 86791 |
| C/02_001b | 189967 | 0.01 | -0.03 | 4852 | 86805 |
| C/02_001c | 958652 | 0.00 | -0.15 | 2995 | 49447 |
| C/02_001d | 3755127 | 1.06 | 0.00 | 2749 | 86660 |
| C/02_001e | 1618977 | 0.00 | -0.21 | 740 | 25848 |
| C/14_001a | 292105 | 0.09 | -0.12 | 4615 | 86694 |
| C/14_001b | 190411 | 0.05 | -0.03 | 4991 | 86685 |
| C/14_001c | 957281 | 0.02 | 0.00 | 5219 | 86678 |
| C/14_001d | 3754341 | 1.05 | 0.00 | 2702 | 86674 |
| C/14_001e | 1617510 | 0.00 | 0.00 | 2461 | 49641 |
| | Avg | 3.21 | -0.04 | 2754 | 77647 |
| D/14_12a | 142895040 | 23.47 | 0.00 | 801 | 86558 |
| D/14_12b | 95045495 | 25.91 | 0.00 | 590 | 86594 |
| D/14_12c | 206994417 | 22.56 | 0.00 | 780 | 86509 |
| D/14_12d | 132884404 | 28.47 | 0.00 | 631 | 86562 |
| D/14_12e | 97150719 | 25.21 | 0.00 | 625 | 86568 |
| D/20_001a | 347368 | 0.13 | -0.14 | 2678 | 86673 |
| D/20_001b | 226417 | 0.07 | -0.13 | 3419 | 86705 |
| D/20_001c | 524614 | 0.06 | -0.02 | 3714 | 86675 |
| D/20_001d | 441022 | 1.34 | 0.00 | 1931 | 86660 |
| D/20_001e | 302521 | 0.04 | -0.18 | 2177 | 86655 |
| D/01_20a | 166196678 | 4.64 | 0.00 | 556 | 86582 |
| D/01_20b | 107499361 | 2.47 | 0.00 | 576 | 86633 |
| D/01_20c | 202610339 | 4.00 | 0.00 | 426 | 86611 |
| D/01_20d | 146654963 | 3.77 | 0.00 | 422 | 86534 |
| D/01_20e | 5624598 | 5.83 | 0.00 | 692 | 86606 |
| | Avg | 9.86 | -0.03 | 1335 | 86608 |
| E/01_001a | 20885 | 3.48 | 0.00 | 384 | 86711 |
| E/01_001b | 860538 | 0.82 | 0.00 | 97 | 87320 |
| E/01_001c | 649314 | 1.13 | 0.00 | 360 | 86746 |
| E/01_001d | 828565 | 3.15 | 0.00 | 509 | 86771 |
| E/01_001e | 1022024 | 3.03 | -0.05 | 367 | 86708 |
| E/20_001a | 20878 | 3.38 | 0.00 | 348 | 86814 |
| E/20_001b | 879399 | 1.23 | 0.00 | 314 | 86660 |
| E/20_001c | 651077 | 1.44 | 0.00 | 354 | 86746 |
| E/20_001d | 828518 | 3.16 | 0.00 | 498 | 86868 |
| E/20_001e | 1026167 | 3.46 | 0.00 | 383 | 86802 |
| E/01_20a | 161978808 | 7.97 | 0.00 | 143 | 86677 |
| E/01_20b | 302985811 | 12.98 | 0.00 | 173 | 86789 |
| E/01_20c | 29259370 | 5.61 | 0.00 | 78 | 86662 |
| E/01_20d | 72555417 | 8.52 | 0.00 | 156 | 86605 |
| E/01_20e | 299822209 | 19.44 | 0.00 | 326 | 86854 |
| | Avg | 5.25 | 0.00 | 299 | 86782 |

Table 5.9: Optimality gaps for very large instances of groups C to E

### 5.8.3 Feasilibity Pump embedded heuristic Branch-and-Cut

As mentioned earlier, a heuristic Branch-and-Cut algorithm was tested too. In this version, binary variables are fixed heuristically according to their values of $\tilde{y}$, provided by the Volume while solving each node LP. Namely, for all $a \in A_*$, we set variables to one, if $\tilde{y}_a > 1 - \sigma$, or to zero, if $\tilde{y}_a < \sigma$, at each search tree node. The parameter $\sigma$ is initially set to zero, and it is increased as the algorithm goes deeper

in the tree, in such a way that $\sigma = 1e{-}30(10^{dl})$, where $dl$ is the depth level of the current node. A maximum value is considered, such that $\sigma \leq 1e{-}4$. Then, the Feasibility Pump procedure, described in section 5.6.1, is applied in every node whose depth $dl$ is impair.

To accelerate node evaluations, no presolving step for child nodes is performed, and no strong or reliability branching is considered. Instead, the branching variable is chosen as the $argmax_{a \in A_*}\{w_a * min\{\tilde{y}_a, 1 - \tilde{y}_a\}\}$. Moreover, node LPs are never solved exactly by simplex algorithm.

We first present, in Tables 5.10 and 5.11, the results obtained for the large benchmark instances Canad-N, and the very large instances of groups A to E. Like in the previous section, the instances F to H could not be tested due to limits in memory space. The solutions obtained with the heuristic version were compared to the previous results presented in section 3.6.4. The best feasible solution found for each instance is shown in columns '$\bar{Z}$' of Tables 5.10 and 5.11. Therefore, the distance 'Dev' of the solution value $z$, given by each method, w.r.t. the best value is shown in columns 'Cplex', 'LH', and 'FP', respectively for the Cplex solver, running 24h, the Lagrangian heuristic (LH), and the Feasibility Pump Branch-and-Cut heuristic (FP), also running 24h. The lines 'Avg' report the averages concerning all instances in the table.

$$\text{Dev} = (\bar{Z} - z)/z * 100$$

Considering the best value obtained with either the Volume or the Bundle version of LH, results have shown that, in general, the Lagrangian heuristic LH remains the best approach for the set of large instances, providing good quality bounds in less computational time. However, when compared to Cplex, the FP heuristic is clearly more efficient on average. Except for the first four Canad-N instances, for which both methods found the optimal value, the algorithms stopped due to the time limit. Thus, for the same computational times, the FP algorithm provided far better solutions on average. For 31 Canad-N instances, FP performed better, and the distance to the best feasible solution remained under 8%, while Cplex provided more than 80% worse solutions in some cases. Moreover, new best feasible solutions could be found with the FP heuristic for 19 Canad-N instances. The values in boldface, in Tables 5.10 and 5.11, indicate when the FP algorithm was the unique method to provide the '$\bar{Z}$' value.

| Instance | $\bar{Z}$ | Cplex | LH | FP | Instance | $\bar{Z}$ | Cplex | LH | FP |
|---|---|---|---|---|---|---|---|---|---|
| pN01 | 2122239 | 0.00 | 0.36 | 0.00 | pN25 | 1685350 | 20.23 | 0.00 | 0.00 |
| pN02 | 1664693 | 0.00 | 1.12 | 0.00 | pN26 | 2509613 | 2.38 | 0.01 | **0.00** |
| pN03 | 2139957 | 0.00 | 1.28 | 0.00 | pN27 | 1545679 | 0.00 | 0.43 | 0.61 |
| pN04 | 2840339 | 0.00 | 1.46 | 0.00 | pN28 | 2246542 | 0.20 | 0.61 | **0.00** |
| pN05 | 125255 | 0.00 | 0.31 | 0.40 | pN29 | 972205 | 83.51 | 0.01 | **0.00** |
| pN06 | 81932 | 0.00 | 0.31 | 0.31 | pN30 | 857118 | 25.20 | 0.00 | 0.94 |
| pN07 | 120638 | 0.00 | 0.21 | 1.20 | pN31 | 572552 | 72.31 | 0.00 | 0.00 |
| pN08 | 83565 | 0.00 | 0.37 | 0.37 | pN32 | 923267 | 81.59 | 0.38 | **0.00** |
| pN09 | 374989 | 0.00 | 0.07 | 1.75 | pN33 | 2129079 | 0.53 | 1.62 | **0.00** |
| pN10 | 603615 | 0.00 | 2.15 | 2.15 | pN34 | 1061480 | 18.63 | 1.36 | **0.00** |
| pN11 | 473897 | 0.00 | 1.71 | 0.24 | pN35 | 2519584 | 3.62 | 2.86 | **0.00** |
| pN12 | 828747 | 0.70 | 0.35 | **0.00** | pN36 | 1223827 | 3.16 | 1.20 | **0.00** |
| pN13 | 354007 | 0.00 | 0.10 | 0.34 | pN37 | 1189320 | 36.59 | 2.03 | **0.00** |
| pN14 | 416131 | 1.47 | 0.00 | 0.00 | pN38 | 182726 | 28.40 | 0.50 | **0.00** |
| pN15 | 378156 | 0.00 | 0.36 | 0.71 | pN39 | 995193 | 41.29 | 0.78 | **0.00** |
| pN16 | 598857 | 1.98 | 0.36 | **0.00** | pN40 | 100499 | 49.46 | 0.31 | **0.00** |
| pN17 | 407057 | 0.00 | 2.15 | 1.13 | pN41 | 273748 | 27.51 | 0.00 | 0.50 |
| pN18 | 159099 | 0.33 | 1.78 | **0.00** | pN42 | 1605884 | 43.85 | 0.00 | 0.00 |
| pN19 | 358984 | 0.00 | 2.20 | 1.99 | pN43 | 916144 | 39.99 | 0.57 | **0.00** |
| pN20 | 149918 | 0.00 | 2.07 | 0.93 | pN44 | 1983636 | 41.48 | 0.00 | 0.14 |
| pN21 | 314877 | 2.98 | 0.00 | 0.58 | pN45 | 1070118 | 83.68 | 0.00 | 0.04 |
| pN22 | 555984 | 1.21 | 2.68 | **0.00** | pN46 | 580398 | 66.97 | 0.00 | 0.21 |
| pN23 | 435738 | 0.24 | 0.67 | **0.00** | pN47 | 1298479 | 80.28 | 0.00 | 0.29 |
| pN24 | 804167 | 0.18 | 1.01 | **0.00** | pN48 | 2236171 | 82.86 | 0.00 | 0.92 |
| | | | | | | Avg | 19.63 | 0.72 | 0.33 |

Table 5.10: Branch-and-Cut heuristic results for benchmark instances Canad-N

Remind that Cplex could not find any feasible solution for instances A to E, so the comparison, in this case, is made only between LH and FP. Again, considering the times presented in section 3.6.4, LH provided better performance, presenting, on average, better solutions in less time. Nevertheless, a new best solution was obtained with the FP algorithm for 25 instances among the 75 present in groups A-E. Moreover, a considerable improvement was obtained for instances C/02_10c and C/02_10e, as can be seen in Table 5.11.

| Instance | $\bar{Z}$ | LH | FP | Instance | $\bar{Z}$ | LH | FP |
|---|---|---|---|---|---|---|---|
| A/08_05a | 143957028 | 0.00 | 3.47 | C/02_001d | 3755127 | 0.00 | 0.20 |
| A/08_05b | 142543732 | 0.00 | 6.26 | C/02_001e | 1618977 | 0.07 | **0.00** |
| A/08_05c | 144182939 | 0.00 | 7.15 | C/14_001a | 292105 | 0.11 | **0.00** |
| A/08_05d | 497221 | 0.00 | 2.25 | C/14_001b | 190411 | 0.17 | **0.00** |
| A/08_05e | 13923307 | 0.00 | 7.22 | C/14_001c | 957281 | 0.12 | **0.00** |
| A/10_05a | 147903153 | 0.00 | 2.38 | C/14_001d | 3754341 | 0.10 | **0.00** |
| A/10_05b | 148076489 | 0.00 | 3.70 | C/14_001e | 1617510 | 0.21 | **0.00** |
| A/10_05c | 149823462 | 0.00 | 4.12 | D/14_12a | 142895040 | 0.73 | **0.00** |
| A/10_05d | 522125 | 0.00 | 9.07 | D/14_12b | 95045495 | 0.00 | 3.37 |
| A/10_05e | 14807132 | 0.00 | 11.38 | D/14_12c | 206994417 | 0.00 | 3.83 |
| A/14_05a | 158128681 | 0.82 | **0.00** | D/14_12d | 132884404 | 0.00 | 0.59 |
| A/14_05b | 157377772 | 0.00 | 3.08 | D/14_12e | 97150719 | 0.00 | 2.95 |
| A/14_05c | 164424149 | 0.00 | 1.31 | D/20_001a | 347368 | 0.14 | **0.00** |
| A/14_05d | 551592 | 0.00 | 5.31 | D/20_001b | 226417 | 0.00 | 0.01 |
| A/14_05e | 15575836 | 0.00 | 6.84 | D/20_001c | 524614 | 0.09 | **0.00** |
| B/14_10a | 69846114 | 0.00 | 10.28 | D/20_001d | 441022 | 0.30 | **0.00** |
| B/14_10b | 65989759 | 0.00 | 6.18 | D/20_001e | 302521 | 0.05 | **0.00** |
| B/14_10c | 70551393 | 0.00 | 5.64 | D/01_20a | 166196678 | 0.00 | 1.70 |
| B/14_10d | 75213876 | 0.00 | 5.04 | D/01_20b | 107499361 | 1.96 | **0.00** |
| B/14_10e | 51353874 | 0.00 | 0.42 | D/01_20c | 202610339 | 1.99 | **0.00** |
| B/10_10a | 65168024 | 0.00 | 6.18 | D/01_20d | 146654963 | 0.00 | 2.83 |
| B/10_10b | 61131120 | 0.00 | 4.92 | D/01_20e | 5624598 | 2.89 | **0.00** |
| B/10_10c | 67824166 | 0.00 | 11.94 | E/01_001a | 20885 | 0.32 | **0.00** |
| B/10_10d | 68211559 | 0.00 | 4.26 | E/01_001b | 860538 | 0.00 | 0.30 |
| B/10_10e | 46137950 | 0.00 | 5.86 | E/01_001c | 649314 | 0.00 | 0.33 |
| B/06_10a | 58116298 | 0.00 | 5.50 | E/01_001d | 828565 | 0.00 | 1.67 |
| B/06_10b | 53688788 | 0.00 | 3.01 | E/01_001e | 1022024 | 0.67 | **0.00** |
| B/06_10c | 59078527 | 0.00 | 7.44 | E/20_001a | 20878 | 0.11 | **0.00** |
| B/06_10d | 60776805 | 0.00 | 6.26 | E/20_001b | 879399 | 0.00 | 0.10 |
| B/06_10e | 40429256 | 0.00 | 10.17 | E/20_001c | 651077 | 0.00 | 0.19 |
| C/02_10a | 72920073 | 0.00 | 2.41 | E/20_001d | 828518 | 0.00 | 1.74 |
| C/02_10b | 45385926 | 0.00 | 3.17 | E/20_001e | 1026167 | 0.00 | 0.08 |
| C/02_10c | 32494545 | 5.64 | **0.00** | E/01_20a | 161978808 | 0.00 | 1.79 |
| C/02_10d | 198264021 | 0.00 | 0.94 | E/01_20b | 302985811 | 0.05 | **0.00** |
| C/02_10e | 811852234 | 3.99 | **0.00** | E/01_20c | 29259370 | 0.00 | 4.02 |
| C/02_001a | 287761 | 0.24 | **0.00** | E/01_20d | 72555417 | 0.00 | 6.17 |
| C/02_001b | 189967 | 0.20 | **0.00** | E/01_20e | 299822209 | 0.10 | **0.00** |
| C/02_001c | 958652 | 0.09 | **0.00** | | Avg | 0.28 | 2.77 |

Table 5.11: Branch-and-Cut heuristic results for large instances A to E

We now report the results for smaller benchmark instances Canad-C and Canad-R. The performances in terms of solution values were compared to other heuristics present in the literature, namely the ones considered in section 3.6.4: the capacity scaling heuristic (SCALE) [141], the path relinking algorithm (RELINK) [111], the multilevel cooperative algorithm (MULTI) [65], the local branching heuristic (LCBR) [193], and the LH heuristic proposed earlier (we stress that it is considered the best value obtained with either the Volume or the Bundle version of LH). Moreover, we add the relatively recent work of Paraskevopoulos *et al.* [184], who proposed an evolutionary algorithm (EVOL). For this second set of instances, the FP heuristic was let to run for 10 hours, and the solution obtained

after 5 hours of computation is also reported, if applicable. Those intermediary values were reported aiming to reduce the effect of computational time limits on different processors. The values related to the limit of 5 hours are identified by 'FP 5h', and 'FP 10h' identifies the ones related to the limit of 10 hours.

Remind that for the Canad-R instances, only solution values obtained with CYCLE, SCALE and LH can be found in the literature, so Tables 5.12 and 5.13 present the average deviation 'Dev' of the solution values obtained with each of those heuristic approaches, w.r.t. the best one among them (the detailed results are presented in the appendix D ). The average computational times are shown in column 'Time', in seconds, and the results are grouped by instance sizes, like in Table 3.9 of section 3.6.4.

As one can see in Table 5.12, for the smallest Canad-R instances (r01 to r09), the FP heuristic found the best solutions in less than 60 seconds. The values in boldface point out that the procedure was able to provide strictly better values in 7 out of 9 instance groups, and for the other two it provided the same quality solutions on average.

| Instances | CYCLE | SCALE | LH | FP | Time |
|---|---|---|---|---|---|
| r01.1-r01.6 | 0.00 | 0.63 | 0.00 | 0.00 | 0.24 |
| r02.1-r02.6 | 0.77 | 0.11 | 0.00 | 0.00 | 0.51 |
| r03.1-r03.6 | 1.59 | 0.07 | 0.01 | **0.00** | 0.81 |
| r04.1-r04.9 | 0.10 | 0.44 | 0.41 | **0.00** | 1.51 |
| r05.1-r05.9 | 0.48 | 0.51 | 0.60 | **0.00** | 2.95 |
| r06.1-r06.9 | 2.39 | 0.20 | 0.16 | **0.00** | 44.55 |
| r07.1-r07.9 | 0.41 | 1.23 | 0.42 | **0.00** | 9.88 |
| r08.1-r08.9 | 0.89 | 0.78 | 0.49 | **0.00** | 21.18 |
| r09.1-r09.9 | 2.76 | 0.31 | 0.31 | **0.00** | 49.02 |

Table 5.12: Branch-and-Cut heuristic results for benchmark instances Canad-RI : r01 to r10

In its turn, Table 5.13 presents the results for the greatest Canad-R instances (r10 to r18). For instances in 'r14', 'r15', 'r17', and 'r18', the algorithm ran for more than 5 hours, so the results at this point were reported in those cases. As one can observe, for all groups, given by the values in boldface, strictly better solutions on average were provided by the FP algorithm, within 5 hours or less, as for 'r10', 'r11', 'r12' and 'r16'. Moreover, better solutions on average were obtained by letting the algorithm run for bigger amounts of time.

| Instances | CYCLE | SCALE | LH | FP 5h | FP 10h | Time |
|---|---|---|---|---|---|---|
| r10.1-r10.9 | 2.67 | 0.68 | 0.77 | **0.00** | 0.00 | 594.07 |
| r11.1-r11.9 | 2.61 | 0.19 | 0.28 | **0.00** | 0.00 | 5973.02 |
| r12.1-r12.9 | 4.78 | 0.18 | 0.42 | **0.03** | 0.00 | 3107.05 |
| r13.1-r13.9 | 3.28 | 1.20 | 1.15 | **0.00** | 0.00 | 13178.43 |
| r14.1-r14.9 | 6.50 | 0.62 | 1.07 | **0.06** | 0.02 | 17854.29 |
| r15.1-r15.9 | 6.91 | 0.13 | 0.53 | **0.05** | 0.04 | 28264.11 |
| r16.1-r16.9 | 3.16 | 1.08 | 1.32 | **0.00** | 0.00 | 12386.27 |
| r17.1-r17.9 | 5.85 | 0.65 | 1.13 | **0.38** | 0.00 | 23473.50 |
| r18.1-r18.9 | 9.64 | 0.51 | 0.94 | **0.25** | 0.01 | 31623.60 |

Table 5.13: Branch-and-Cut heuristic results for benchmark instances Canad-R : r10 to r18

Finally, Table 5.14 presents the results for each Canad-C instance. Like in precedent tables, the deviation 'Dev' of the solution value obtained with each heuristic approach, w.r.t. the best one among them is shown under the respective column. Again, column 'Time' reports the computational times in seconds. The solution nominal values are presented in the appendix D.

Regarding the averages presented in the last line of Table 5.14, the FP heuristic remains competitive for Canad-C instances too, providing within a time limit of 5 hours the best solutions on average, which are generally improved by letting the algorithm run for 10 hours. Given in boldface, 'FP 5h' uniquely provided the best solution for instances 'c47', 'c68', and 'c72', and for other 5 instances, it could provide strictly better solutions within 10 hours. In total, for 17/43 instances the 'FP 5h' heuristic provided better or equal quality solutions. The same can be affirmed for a total of 23/43 instances, considering 'FP 10h'.

| Instance | SCALE | RELINK | MULTI | LCBR | EVOL | LH | FP 5h | FP 10h | Time |
|---|---|---|---|---|---|---|---|---|---|
| c33.dow | 0.05 | 0.13 | 0.67 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 42 |
| c35.dow | 0.12 | 0.09 | 0.00 | 0.00 | 0.00 | 0.04 | 0.00 | 0.00 | 413 |
| c36.dow | 0.22 | 0.39 | 1.51 | 0.00 | 0.02 | 0.27 | 0.20 | 0.00 | 36435 |
| c37.dow | 0.04 | 6.17 | 4.43 | 1.14 | 0.27 | 0.57 | 0.06 | **0.00** | 33791 |
| c38.dow | 0.00 | 6.99 | 3.85 | 4.05 | 0.94 | 0.51 | 0.44 | 0.44 | 36089 |
| c39.dow | 0.06 | 6.47 | 4.03 | 0.13 | 0.30 | 0.28 | 0.74 | **0.00** | 23801 |
| c40.dow | 0.00 | 7.74 | 3.58 | 3.54 | 0.73 | 0.68 | 0.36 | 0.36 | 36082 |
| c41.dow | 0.00 | 0.00 | 0.10 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 11 |
| c42.dow | 0.29 | 0.74 | 1.26 | 0.00 | 0.00 | 0.46 | 0.00 | 0.00 | 560 |
| c43.dow | 0.01 | 0.00 | 0.32 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 818 |
| c44.dow | 0.00 | 0.95 | 2.42 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 200 |
| c45.dow | 0.00 | 4.18 | 4.22 | 1.91 | 0.49 | 0.09 | 0.83 | 0.03 | 36124 |
| c46.dow | 0.21 | 6.43 | 5.26 | 3.02 | 1.08 | 0.60 | 0.80 | **0.00** | 36113 |
| c47.dow | 0.41 | 4.91 | 2.93 | 1.54 | 0.60 | 0.50 | **0.00** | 0.00 | 9994 |
| c48.dow | 0.31 | 5.34 | 3.28 | 2.08 | 0.02 | 0.00 | 0.15 | 0.15 | 36101 |
| c49.dow | 0.11 | 1.60 | 3.10 | 0.00 | 0.13 | 0.16 | 0.21 | 0.21 | 36138 |
| c50.dow | 0.62 | 7.69 | 5.62 | 2.13 | 0.43 | 0.78 | 0.34 | **0.00** | 36129 |
| c51.dow | 0.29 | 1.67 | 2.58 | 0.00 | 0.10 | 0.37 | 0.35 | 0.29 | 36162 |
| c52.dow | 0.99 | 7.80 | 4.51 | 3.21 | 0.00 | 0.89 | 1.78 | 1.78 | 36120 |
| c53.dow | 0.00 | 5.50 | 2.44 | 1.33 | 0.31 | 0.42 | 0.50 | 0.50 | 36121 |
| c54.dow | 0.00 | 8.38 | 4.57 | 5.25 | 1.12 | 0.63 | 0.94 | 0.84 | 36050 |
| c55.dow | 0.00 | 4.60 | 5.24 | 0.52 | 0.91 | 0.20 | 0.60 | 0.57 | 36072 |
| c56.dow | 0.00 | 6.68 | 4.66 | 9.38 | 1.09 | 0.53 | 0.84 | 0.62 | 36108 |
| c57.dow | 0.07 | 2.30 | 2.59 | 0.00 | 0.00 | 0.07 | 0.00 | 0.00 | 1185 |
| c58.dow | 0.00 | 4.59 | 5.59 | 0.13 | 0.57 | 0.73 | 0.74 | 0.74 | 36179 |
| c59.dow | 0.57 | 2.76 | 3.27 | 0.00 | 0.38 | 0.59 | 0.65 | 0.65 | 36180 |
| c60.dow | 0.46 | 2.60 | 3.17 | 0.00 | 0.06 | 0.24 | 0.18 | 0.06 | 36163 |
| c61.dow | 0.00 | 6.80 | 4.54 | 5.60 | 0.77 | 0.67 | 0.89 | 0.89 | 36083 |
| c62.dow | 0.00 | 6.87 | 6.20 | 20.44 | 1.49 | 0.43 | 1.29 | 1.23 | 36049 |
| c63.dow | 0.00 | 5.84 | 3.92 | 1.42 | 0.86 | 0.45 | 0.76 | 0.68 | 36160 |
| c64.dow | 0.00 | 7.70 | 5.87 | 10.20 | 1.72 | 0.41 | 1.15 | 0.71 | 36125 |
| c65.dow | 0.64 | 0.00 | 0.00 | 0.00 | 0.00 | 6.64 | 0.00 | 0.00 | 11 |
| c66.dow | 1.72 | 0.00 | 0.08 | 0.00 | 0.00 | 2.67 | 0.00 | 0.00 | 16 |
| c67.dow | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 1 |
| c68.dow | 1.11 | 1.59 | 1.47 | 0.72 | 0.72 | 2.05 | **0.00** | 0.00 | 191 |
| c69.dow | 0.32 | 1.04 | 1.08 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 174 |
| c70.dow | 0.00 | 0.03 | 0.03 | 0.00 | 0.00 | 0.11 | 0.00 | 0.00 | 16 |
| c71.dow | 2.09 | 0.30 | 0.30 | 3.00 | 0.00 | 2.23 | 0.00 | 0.00 | 1035 |
| c72.dow | 12.24 | 1.10 | 2.60 | 4.15 | 1.53 | 8.99 | **0.00** | 0.00 | 36241 |
| c73.dow | 0.01 | 0.22 | 0.46 | 0.00 | 0.00 | 0.28 | 0.00 | 0.00 | 935 |
| c74.dow | 5.65 | 4.49 | 2.85 | 1.71 | 0.91 | 4.44 | 0.65 | **0.00** | 23001 |
| c75.dow | 3.31 | 1.29 | 4.25 | 1.48 | 0.00 | 3.97 | 0.36 | 0.36 | 36144 |
| c76.dow | 0.02 | 0.03 | 0.12 | 0.00 | 0.05 | 2.34 | 0.12 | 0.12 | 36216 |
| Avg | 0.74 | 3.35 | 2.77 | 2.05 | 0.41 | 1.05 | 0.37 | 0.26 | 22409 |

Table 5.14: Branch-and-Cut heuristic results for benchmark instances Canad-C

## 5.9 Conclusion

In this chapter, a Volume-based Branch-and-Cut algorithm was proposed, embedding a Lagrangian Feasibility Pump heuristic. The results have shown that the exact method is competitive with the most efficient algorithms available in the literature. Indeed, for the largest instances, the proposed method performs

better, providing better optimality gaps on average, for practically the same computational time. Moreover, the gaps of three unsolved benchmark instances could be closed in the present work. With respect to very large-scale instances, the proposed scheme was the only one able to provide sharp optimality gaps. Three very large-scale instances were solved to optimality.

With respect to the Feasibility Pump embedded Branch-and-Cut heuristic version, the results have shown that the procedure is competitive to other efficient heuristics in the literature, providing better solution values on average, for benchmark instances. Moreover, new upper bounds could be obtained for some large-scale instances, with the Feasibility Pump heuristic as part of the exact approach.

# Chapter 6

# Conclusion

The present thesis addressed the fixed-charge multicommodity capacitated network design, a typical problem of telecommunications, logistics, and transportation fields. Already known to be NP-Hard, we have shown in chapter 2 that even small instances of the problem can be very difficult for one of the best solvers in the market. In that same chapter, a discussion about some complicating aspects of the problem was included. It was shown that sufficiently tight capacities, combined with high fixed-charges belonging to a not too wide range of values, can be complicating features for small instances of 10 nodes and complete patterns of arcs and commodities.

Further in this document, Lagrangian decomposition methods were proposed to produce good quality solutions for large and very large scale instances of the problem. We take profit from the fact that, by Lagrangian relaxation, one can decompose the problem into smaller subproblems, to accelerate the optimization process. Indeed, it is known that, with the capacity and forcing constraints relaxed, the problem decomposes into shortest-path subproblems, one for each commodity. A second alternative is to relax flow balancing constraints to obtain knapsack subproblems, one per arc. Moreover, those two approaches can be combined in a 'total' relaxation, so the problem decomposes into simple inspection subproblems, one for each arc-commodity pair. As mentioned, other Lagrangian relaxations have been proposed in the literature, that decompose by nodes, but resulting in harder subproblems. It has been shown that for the current purposes, the 'knapsack' relaxation was the most suitable.

As discussed, the Lagrangian relaxation results in the so-called Lagrangian dual problem, whose objective function is a nondifferentiable piecewise function. The performances of two state-of-the-art NDO solvers: the Volume Algorithm and the Bundle Method were compared while solving the dual Lagrangian problems. The important contribution of such a comparison was the surprising overall perfor-

mance of the Volume Algorithm, compared extensively to the Bundle-based best solver on very large-scale instances of FCMC (cf. [199]). The results have shown that for large-scale FCMC instances, the Volume Algorithm performed better, providing on average equal or better bounds in less computational time. Therefore, the whole development of methods presented in this document was based on Lagrangian relaxation and the Volume Algorithm.

The main goal of this thesis was to provide tight optimality gaps for large scale FCMC instances. In that sense, a Lagrangian heuristic was proposed, providing relatively good quality upper bounds for large-scale instances, far better than the ones provided by Cplex.

From another perspective, a Relax-and-Cut procedure was implemented in chapter 4, aiming at improving lower bounds. Two alternative ways to enhance performances were presented: sensitivity analysis and constraint scaling. Indeed, those features have provided better bounds for all instances, with practically no losses in time-consuming aspects. On average, the obtained lower bounds were close to 1% better than the ones obtained with the standard Volume Algorithm. Nevertheless, in some cases the bound improvements were up to 11% for benchmark instances.

Finally, the Relax-and-Cut algorithm was embedded into a Branch-and-Cut scheme to further improve lower and upper bounds. The proposed method could provide optimal bounds for three of the unsolved benchmark instances, and it was able to provide sharp bounds for some very large-scale instances, solving three of them to optimality. Moreover, a heuristic version based on a Lagrangian Feasibility Pump algorithm was tested, obtaining competitive results, when compared to the best heuristics in the literature.

We conclude the thesis presenting the next steps and future research avenues for the considered problem and related ones. Given the promising results obtained with the Volume-based Branch-and-Cut algorithm, a more fine-tuned set of parameters, which includes the Feasibility Pump heuristic, is being tested. A paper submission is contemplated to publish the whole study on the Branch-and-Cut algorithm detailed in chapter 5.

Future research avenues may include the development of fast Lagrangian Branch-and-Cut schemes in similar problems, for example with side constraints involved. Moreover, the application of sensitivity analysis along with relax-and-cut schemes for other problems, in which constraint scaling is not trivial, might be an interesting topic. In the presence of near-optimal solutions, the variable fixing procedures used in the Branch-and-Cut scheme could be applied to relax-and-cut contexts, possibly improving performances. Furthermore, the comparison between NDO

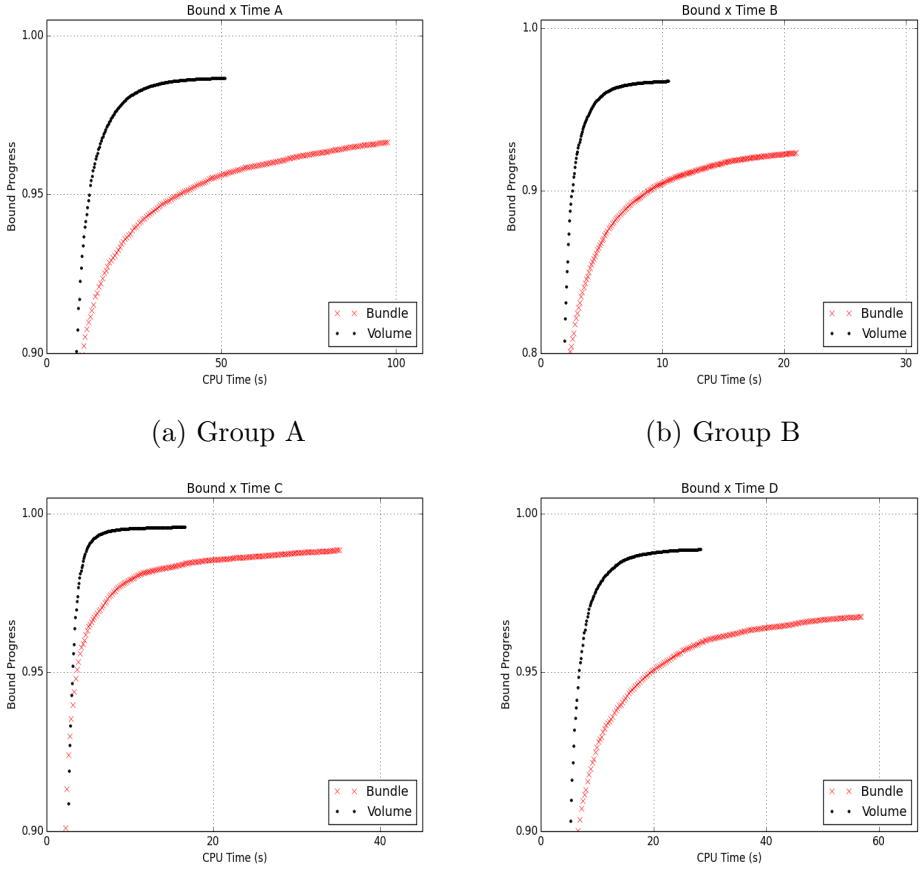
solvers may be enlarged to englobe other efficient algorithms, taking different problems into consideration too. Finally, the combination of Lagrangian relaxation and the Feasibility Pump heuristic could be tested on other integer problems, where integer variables are not binary.

# Appendix A

# Additional results for 'total' relaxation

We plot in Figures A.1 and A.2 the bound progression by the computation time for the larger instances. As in section 3.6.4, the curves indicate the ratio between the current value $LB_x^t$ and $LB_{best}$ measured every 5 iterations and the respective computation time at that precise moment. Note that, $LB_{best}$ is the best know lower bound for a given instance, obtained with either Volume or Bundle, and either 'knapsack' relaxation or 'total' relaxation.

Indeed, the curvers confirm that the 'total' relaxation provided worse bounds in average, since they remain relatively far form the best lower bound (1.00 in the graphics). That is even more perceivable with the Bundle Method. Morever, analysing those graphics, the same statements made in section 3.6.4 can be inferred in the present case.



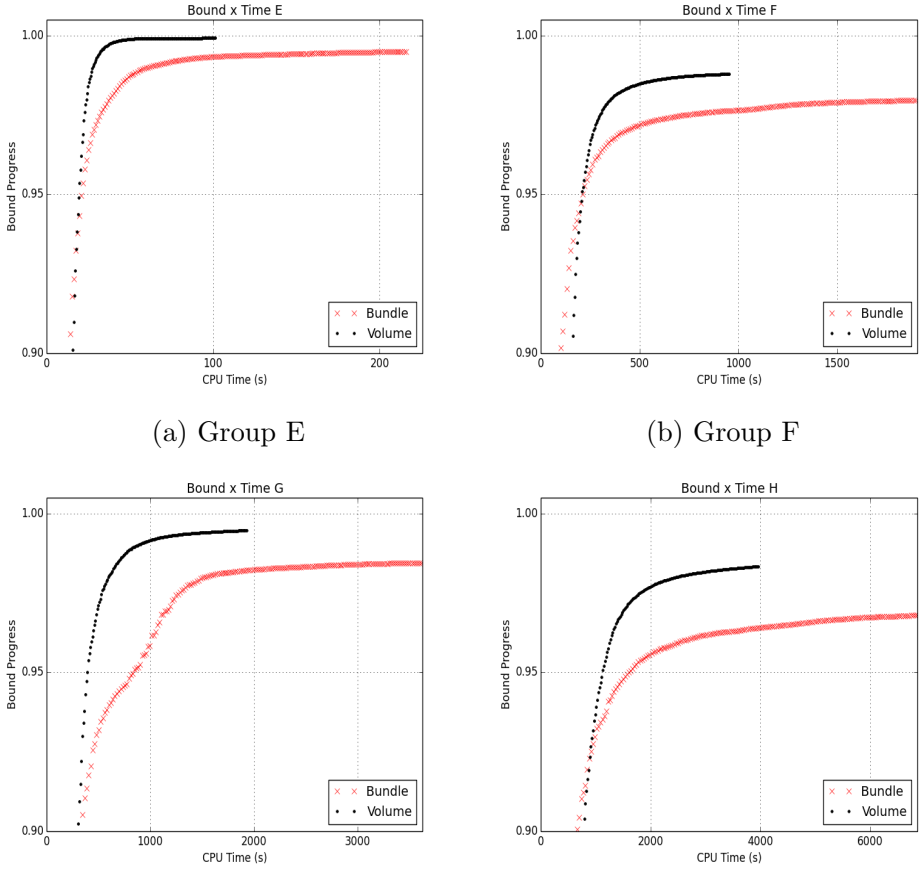

(a) Group A

(b) Group B

(c) Group C

(d) Group D

Figure A.1: 'Total' relaxation average bound progression with respect to computation time, for large instances of groups A-D




(a) Group E

(b) Group F

(c) Group G

(d) Group H

Figure A.2: 'Total' relaxation average bound progression with respect to computation time, for large instances of groups E-H

# Appendix B

# Additional results for ‘knapsack’ relaxation

We plot in Figures B.1 and B.2 the bound progression by the computation time for the instances A, B, C, D and Canad-N. As in section 3.6.4, the curves indicate the ratio between the current value $LB_x^t$ and $LB_{best}$ measured every 5 iterations and the respective computation time at that precise moment. Note that, $LB_{best}$ is the best know lower bound for a given instance, obtained with either Volume or Bundle. As one can notice, the remarks made in section 3.6.4 are valid for these instances too.

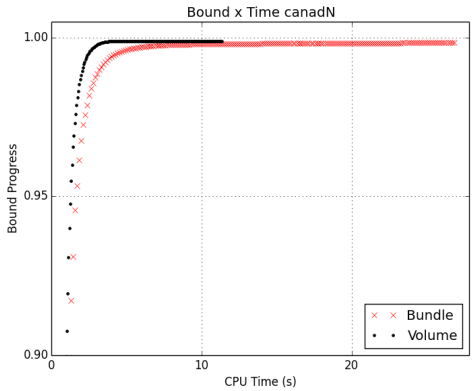
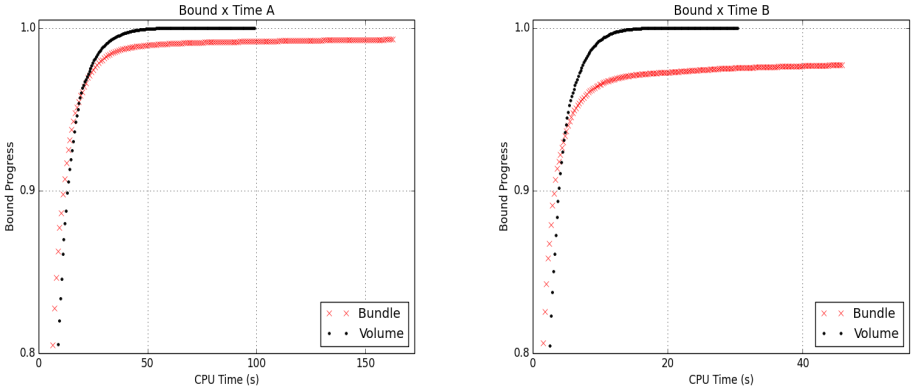
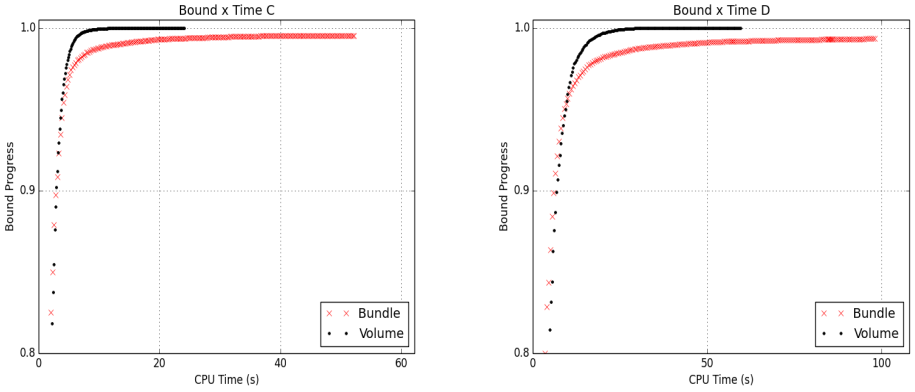



Figure B.1: ‘Knapsack’ relaxation average bound progression with respect to computation time, for large instances of group Canad-N




(a) Group A

(b) Group B




(c) Group C

(d) Group D

Figure B.2: 'Knapsack' relaxation average bound progression with respect to computation time, for large instances of groups A-D

# Appendix C

# Relax-and-Cut results for very large scale instances

Here, we present in Table C.1 the average results for very large scale FCMC instances (see section 3.6.1 for a detailed description of those instances). The notation used is the same as in section 4.8 : considering $z_l$ the bound given by the Lagrangian knapsack relaxation, solved by the regular Volume Algorithm, and $\bar{z}$ the bound provided by the Relax-and-Cut method, the deviation 'Dev' measures the benefit of the given procedure, relative to $z_l$. Therefore, columns 'Max Dev %' present the maximum deviation obtained for that related group of instances, while columns 'Avg Dev %' give the average deviation for that group. Futhermore, the average computational times are given in columns 't', measured in seconds, and columns 'Num' present the average number of cuts added during the optimization processes. Column 'Iters' show the average number of iterations performed before the algorithm stops.

A time limit of 3 hours, and a limit of 1000 iterations were set. In its turn, the Relax-and-Cut configuration chosed for this testbed adds only cover and minimum cardinality inequalities, and sensitivity analysis and constraint scaling were applied.

| | Max Dev % | Avg Dev % | Cover Num | Min Card Num | t | Iters |
|---|---|---|---|---|---|---|
| A/08_05 | 0.00 | 0.00 | 245 | 249 | 1674 | 1000 |
| A/10_05 | 0.13 | 0.03 | 227 | 226 | 1792 | 1000 |
| A/14_05 | 0.10 | 0.00 | 194 | 193 | 2027 | 1000 |
| B/14_10 | 1.17 | 0.79 | 219 | 215 | 503 | 1000 |
| B/10_10 | 1.25 | 0.92 | 266 | 258 | 445 | 1000 |
| B/06_10 | 0.70 | 0.51 | 249 | 251 | 390 | 1000 |
| C/02_10 | 0.57 | 0.13 | 270 | 268 | 497 | 1000 |
| C/02_001 | 0.01 | 0.00 | 0 | 0 | 271 | 1000 |
| C/14_001 | 0.02 | 0.01 | 0 | 0 | 284 | 1000 |
| D/14_12 | 0.20 | 0.12 | 217 | 216 | 1024 | 1000 |
| D/20_001 | 0.01 | 0.00 | 0 | 0 | 354 | 1000 |
| D/01_20 | 0.20 | 0.10 | 249 | 251 | 1424 | 1000 |
| E/01_001 | 0.01 | 0.01 | 0 | 0 | 2166 | 1000 |
| E/20_001 | 0.04 | 0.01 | 1 | 1 | 2284 | 1000 |
| E/01_20 | 0.07 | 0.04 | 264 | 266 | 1764 | 1000 |
| F/01_20 | 0.01 | 0.00 | 302 | 302 | 9305 | 991 |
| F/20_20 | 0.05 | 0.00 | 196 | 195 | 10132 | 834 |
| F/01_001 | 0.04 | 0.00 | 2 | 2 | 10190 | 750 |
| G/20_001 | 0.00 | 0.00 | 5 | 3 | 10111 | 670 |
| G/20_20 | 0.01 | 0.00 | 204 | 203 | 10157 | 654 |
| G/01_001 | 0.00 | 0.00 | 7 | 7 | 10203 | 597 |
| H/01_20 | 0.00 | 0.00 | 0 | 0 | 19687 | 50 |
| H/20_20 | 0.00 | 0.00 | 0 | 0 | 19809 | 50 |
| H/01_001 | 0.00 | 0.00 | 0 | 0 | 20472 | 50 |

Table C.1: Relax-and-Cut results for instances groups A to H

As one can observe, the average deviations remained under 1%, with the exception of group B/14_10 and B/10_10, for which slightly better improvements were obtained. For groups A to E the computational times did not reached 1 hour, and the algorithm stopped because of the limit of iterations, which reinforce the fact that sensitivity analysis do not increase computational times too much.

For the largest instances (groups F to H), the algorithm stopped because of the time limit, what deteriorated lower bound improvements, especially for group G. In fact, the most time-consuming feature in those cases was the cutset separation heuristic. Since it was runned for every node, every 50 iterations, given the size of the intances, such procedure turned out to be too demanding. In that sense, a

less extensive cutset separation method is preferable in those cases.

# Appendix D

# Branch-and-Cut detailed results for benchmark instances

In this appendix, the detailed results for the Feasibility Pump embedded Branch-and-Cut heuristic are presented for instances Canad-R, in Tables D.2 to D.5 (see section 5.8.3 for a description of the tables). The columns '$\bar{Z}$' show the nominal value of the best feasible solutions found, serving as basis for the computation of deviations 'Dev'.

For all instances 'r01' to 'r11', the 'FP 5h' heuristic have provided the best solutions, while for the remaining instances, in 49/63 cases it could provide the best soltuions. That number raises to 57/63, when 'FP 10h' is considered.

Furthermore, the nominal values for the the best solutions found for the Canad-C instances are given in Table D.1. Again columns '$\bar{Z}$' show those values.

| Instance | $\bar{Z}$ | Instance | $\bar{Z}$ | Instance | $\bar{Z}$ |
|---|---|---|---|---|---|
| c33 | 423848 | c48 | 107521.5 | c62 | 135064 |
| c35 | 371475 | c49 | 54026 | c63 | 95306 |
| c36 | 643036 | c50 | 94209.5 | c64 | 130148 |
| c37 | 94213 | c51 | 52129 | c65 | 14941 |
| c38 | 137642 | c52 | 97856 | c66 | 49899 |
| c39 | 97914 | c53 | 112846 | c67 | 14712 |
| c40 | 136130 | c54 | 149446 | c68 | 37055 |
| c41 | 429398 | c55 | 114641 | c69 | 85530 |
| c42 | 586077 | c56 | 152744 | c70 | 365272 |
| c43 | 464509 | c57 | 47603 | c71 | 23949 |
| c44 | 604198 | c58 | 60194 | c72 | 64562 |
| c45 | 74913 | c59 | 45905 | c73 | 28423 |
| c46 | 115539 | c60 | 55104 | c74 | 49018 |
| c47 | 74991 | c61 | 97972 | c75 | 139535 |
| | | | | c76 | 384809 |

Table D.1: Best heuristic solution values for Canad-C instances

| Instance | $\bar{Z}$ | CYCLE | SCALE | LH | FP | Time |
|---|---|---|---|---|---|---|
| r01.1 | 74079 | 0.00 | 0.00 | 0.00 | 0.00 | 0.0 |
| r01.2 | 92403 | 0.00 | 0.00 | 0.00 | 0.00 | 0.0 |
| r01.3 | 115304 | 0.00 | 0.00 | 0.00 | 0.00 | 0.1 |
| r01.4 | 84908 | 0.00 | 0.28 | 0.00 | 0.00 | 0.4 |
| r01.5 | 113036 | 0.00 | 1.33 | 0.00 | 0.00 | 0.4 |
| r01.6 | 147599 | 0.00 | 2.14 | 0.00 | 0.00 | 0.5 |
| r02.1 | 232239 | 0.00 | 0.00 | 0.00 | 0.00 | 0.5 |
| r02.2 | 322453 | 1.69 | 0.43 | 0.00 | 0.00 | 0.8 |
| r02.3 | 419503 | 1.72 | 0.00 | 0.00 | 0.00 | 0.4 |
| r02.4 | 316437 | 0.00 | 0.00 | 0.00 | 0.00 | 0.5 |
| r02.5 | 431250 | 0.51 | 0.23 | 0.00 | 0.00 | 0.6 |
| r02.6 | 559578 | 0.71 | 0.00 | 0.00 | 0.00 | 0.4 |
| r03.1 | 484830 | 0.00 | 0.00 | 0.00 | 0.00 | 0.5 |
| r03.2 | 703362 | 1.21 | 0.22 | 0.00 | 0.00 | 0.9 |
| r03.3 | 944990 | 3.74 | 0.00 | 0.00 | 0.00 | 0.7 |
| r03.4 | 704247 | 0.28 | 0.00 | 0.00 | 0.00 | 0.7 |
| r03.5 | 932897 | 2.20 | 0.00 | 0.09 | 0.00 | 1.0 |
| r03.6 | 1188638 | 2.10 | 0.20 | 0.00 | 0.00 | 1.1 |

Table D.2: Branch-and-Cut heuristic detailed results for benchmark instances Canad-R : r01 to r03

| Instance | $\bar{Z}$ | CYCLE | SCALE | LH | FP | Time |
|---|---|---|---|---|---|---|
| r04.1 | 31730 | 0.00 | 0.00 | 0.00 | 0.00 | 0.0 |
| r04.2 | 48920 | 0.00 | 0.00 | 0.00 | 0.00 | 0.0 |
| r04.3 | 63767 | 0.00 | 0.00 | 0.00 | 0.00 | 0.0 |
| r04.4 | 33740 | 0.00 | 0.06 | 1.31 | 0.00 | 0.5 |
| r04.5 | 53790 | 0.00 | 0.00 | 0.93 | 0.00 | 0.5 |
| r04.6 | 74030 | 0.00 | 1.44 | 1.26 | 0.00 | 0.5 |
| r04.7 | 68292 | 0.00 | 0.32 | 0.00 | 0.00 | 5.8 |
| r04.8 | 113004 | 0.20 | 0.20 | 0.20 | 0.00 | 4.4 |
| r04.9 | 163208 | 0.74 | 1.95 | 0.00 | 0.00 | 1.9 |
| r05.1 | 123003 | 0.00 | 0.00 | 0.22 | 0.00 | 0.7 |
| r05.2 | 170060 | 0.24 | 0.24 | 2.39 | 0.00 | 0.5 |
| r05.3 | 221486 | 0.00 | 0.64 | 0.80 | 0.00 | 1.1 |
| r05.4 | 131608 | 0.00 | 0.14 | 0.01 | 0.00 | 0.9 |
| r05.5 | 204157 | 0.78 | 0.21 | 0.00 | 0.00 | 3.0 |
| r05.6 | 286524 | 1.96 | 1.99 | 1.95 | 0.00 | 15.5 |
| r05.7 | 278372 | 0.00 | 0.00 | 0.00 | 0.00 | 1.6 |
| r05.8 | 445810 | 0.82 | 0.02 | 0.02 | 0.00 | 2.1 |
| r05.9 | 625879 | 0.50 | 1.39 | 0.00 | 0.00 | 1.1 |
| r06.1 | 245936 | 1.08 | 0.00 | 0.00 | 0.00 | 5.5 |
| r06.2 | 401685 | 2.57 | 0.00 | 0.00 | 0.00 | 14.8 |
| r06.3 | 559477 | 3.33 | 0.00 | 0.00 | 0.00 | 17.5 |
| r06.4 | 286682 | 0.62 | 0.31 | 0.07 | 0.00 | 33.6 |
| r06.5 | 498266 | 3.43 | 0.69 | 0.39 | 0.00 | 135.1 |
| r06.6 | 734414 | 4.83 | 0.66 | 0.86 | 0.00 | 174.3 |
| r06.7 | 682921 | 0.10 | 0.02 | 0.00 | 0.00 | 3.1 |
| r06.8 | 1030479 | 2.03 | 0.00 | 0.00 | 0.00 | 8.8 |
| r06.9 | 423316 | 3.54 | 0.09 | 0.09 | 0.00 | 8.3 |
| r07.1 | 32807 | 0.00 | 0.00 | 0.00 | 0.00 | 0.0 |
| r07.2 | 47252 | 0.00 | 0.00 | 0.00 | 0.00 | 0.0 |
| r07.3 | 62962 | 0.00 | 0.00 | 0.00 | 0.00 | 0.1 |
| r07.4 | 37432 | 0.00 | 0.00 | 0.00 | 0.00 | 8.7 |
| r07.5 | 56475 | 0.20 | 0.77 | 0.20 | 0.00 | 9.6 |
| r07.6 | 77249 | 2.06 | 2.38 | 2.07 | 0.00 | 6.3 |
| r07.7 | 59947 | 0.00 | 0.27 | 0.23 | 0.00 | 27.4 |
| r07.8 | 99194 | 0.96 | 4.77 | 1.28 | 0.00 | 13.3 |
| r07.9 | 141692 | 0.44 | 2.91 | 0.00 | 0.00 | 23.5 |
| r08.1 | 102531 | 0.02 | 0.11 | 0.00 | 0.00 | 0.7 |
| r08.2 | 143894 | 0.00 | 0.00 | 0.00 | 0.00 | 0.1 |
| r08.3 | 182793 | 0.00 | 0.00 | 0.00 | 0.00 | 0.2 |
| r08.4 | 109325 | 0.00 | 0.00 | 0.09 | 0.00 | 2.2 |
| r08.5 | 157047 | 0.71 | 0.43 | 0.83 | 0.00 | 1.5 |
| r08.6 | 207540 | 0.29 | 0.44 | 1.52 | 0.00 | 6.9 |
| r08.7 | 154160 | 0.79 | 1.28 | 0.10 | 0.00 | 45.5 |
| r08.8 | 274867 | 2.92 | 2.14 | 1.56 | 0.00 | 56.4 |
| r08.9 | 415793 | 3.28 | 2.60 | 0.30 | 0.00 | 77.1 |
| r09.1 | 171512 | 0.48 | 0.24 | 0.00 | 0.00 | 2.0 |
| r09.2 | 296712 | 3.36 | 0.48 | 0.00 | 0.00 | 4.0 |
| r09.3 | 424266 | 2.60 | 0.00 | 1.34 | 0.00 | 22.8 |
| r09.4 | 192736 | 0.26 | 0.05 | 0.01 | 0.00 | 6.9 |
| r09.5 | 357318 | 3.95 | 0.00 | 0.48 | 0.00 | 22.7 |
| r09.6 | 522187 | 6.07 | 0.78 | 0.37 | 0.00 | 144.5 |
| r09.7 | 345057 | 0.93 | 0.17 | 0.13 | 0.00 | 38.8 |
| r09.8 | 646579 | 3.47 | 0.09 | 0.09 | 0.00 | 50.9 |
| r09.9 | 951136 | 3.73 | 0.95 | 0.39 | 0.00 | 148.6 |

Table D.3: Branch-and-Cut heuristic detailed results for benchmark instances Canad-R : r04 to r09

| Instance | $\bar{Z}$ | CYCLE | SCALE | LH | FP 5h | FP 10h | Time |
|---|---|---|---|---|---|---|---|
| r10.1 | 200087 | 0.26 | 0.00 | 0.16 | 0.00 | 0.00 | 1.9 |
| r10.2 | 346814 | 1.07 | 1.24 | 2.03 | 0.00 | 0.00 | 158.6 |
| r10.3 | 488015 | 3.77 | 0.89 | 0.72 | 0.00 | 0.00 | 45.7 |
| r10.4 | 229196 | 1.41 | 0.14 | 0.37 | 0.00 | 0.00 | 269.7 |
| r10.5 | 411664 | 4.91 | 1.61 | 0.03 | 0.00 | 0.00 | 1282.2 |
| r10.6 | 609104 | 4.92 | 0.57 | 1.03 | 0.00 | 0.00 | 2142.8 |
| r10.7 | 486895 | 0.38 | 0.19 | 0.28 | 0.00 | 0.00 | 711.6 |
| r10.8 | 951056 | 2.95 | 0.99 | 1.34 | 0.00 | 0.00 | 265.4 |
| r10.9 | 1421862 | 4.40 | 0.46 | 1.00 | 0.00 | 0.00 | 468.8 |
| r11.1 | 714431 | 1.51 | 0.00 | 0.05 | 0.00 | 0.00 | 182.2 |
| r11.2 | 1263713 | 3.24 | 0.36 | 0.64 | 0.00 | 0.00 | 4961.4 |
| r11.3 | 1843611 | 3.68 | 0.60 | 1.05 | 0.00 | 0.00 | 5758.2 |
| r11.4 | 870451 | 0.73 | 0.09 | 0.09 | 0.00 | 0.00 | 2963.2 |
| r11.5 | 1623640 | 4.20 | 0.11 | 0.22 | 0.00 | 0.00 | 1412.9 |
| r11.6 | 2414060 | 7.43 | 0.54 | 0.37 | 0.00 | 0.00 | 2284.5 |
| r11.7 | 2294912 | 0.04 | 0.02 | 0.05 | 0.00 | 0.00 | 42.7 |
| r11.8 | 3507100 | 1.72 | 0.00 | 0.04 | 0.00 | 0.00 | 36127.7 |
| r11.9 | 4579353 | 0.92 | 0.00 | 0.00 | 0.00 | 0.00 | 24.4 |
| r12.1 | 1639443 | 4.33 | 0.09 | 0.09 | 0.00 | 0.00 | 877.3 |
| r12.2 | 3396050 | 9.35 | 0.44 | 1.17 | 0.26 | 0.00 | 21869.4 |
| r12.3 | 5228711 | 13.86 | 1.04 | 1.97 | 0.00 | 0.00 | 4289.7 |
| r12.4 | 2303557 | 0.97 | 0.07 | 0.19 | 0.00 | 0.00 | 289.3 |
| r12.5 | 4669799 | 6.00 | 0.00 | 0.31 | 0.00 | 0.00 | 290.4 |
| r12.6 | 7100019 | 7.04 | 0.00 | 0.02 | 0.00 | 0.00 | 224.1 |
| r12.7 | 7635270 | 0.03 | 0.00 | 0.03 | 0.00 | 0.00 | 76.7 |
| r12.8 | 10067742 | 0.53 | 0.00 | 0.00 | 0.00 | 0.00 | 33.8 |
| r12.9 | 11967768 | 0.92 | 0.00 | 0.00 | 0.00 | 0.00 | 12.9 |
| r13.1 | 142947 | 0.83 | 0.06 | 0.00 | 0.00 | 0.00 | 1.7 |
| r13.2 | 263800 | 2.41 | 0.47 | 0.78 | 0.00 | 0.00 | 404.4 |
| r13.3 | 365836 | 2.44 | 1.19 | 1.26 | 0.00 | 0.00 | 310.2 |
| r13.4 | 150977 | 0.35 | 0.13 | 0.21 | 0.00 | 0.00 | 370.1 |
| r13.5 | 282682 | 3.03 | 0.54 | 0.80 | 0.00 | 0.00 | 4129.6 |
| r13.6 | 406790 | 3.15 | 1.28 | 3.69 | 0.00 | 0.00 | 4572.4 |
| r13.7 | 208533 | 1.84 | 0.50 | 0.46 | 0.00 | 0.00 | 36303.2 |
| r13.8 | 446675 | 7.73 | 3.60 | 1.69 | 0.00 | 0.00 | 36266.9 |
| r13.9 | 699806 | 7.76 | 3.03 | 1.49 | 0.00 | 0.00 | 36247.3 |
| r14.1 | 403414 | 2.82 | 0.22 | 0.00 | 0.00 | 0.00 | 82.6 |
| r14.2 | 749503 | 6.70 | 0.52 | 0.70 | 0.00 | 0.00 | 12180.8 |
| r14.3 | 1063098 | 8.02 | 0.45 | 1.21 | 0.00 | 0.00 | 4657.0 |
| r14.4 | 437607 | 3.44 | 0.15 | 0.17 | 0.00 | 0.00 | 1854.8 |
| r14.5 | 850816 | 6.76 | 0.75 | 0.70 | 0.37 | 0.00 | 36172.4 |
| r14.6 | 1214609 | 8.91 | 0.15 | 2.64 | 0.00 | 0.00 | 16302.1 |
| r14.7 | 669847 | 4.61 | 0.00 | 0.27 | 0.20 | 0.20 | 36123.1 |
| r14.8 | 1615125 | 7.65 | 0.97 | 0.68 | 0.00 | 0.00 | 36085.3 |
| r14.9 | 2606942 | 9.57 | 2.34 | 3.31 | 0.00 | 0.00 | 17230.7 |

Table D.4: Branch-and-Cut heuristic detailed results for benchmark instances Canad-R : r10 to r14

| Instance | $\bar{Z}$ | CYCLE | SCALE | LH | FP 5h | FP 10h | Time |
|---|---|---|---|---|---|---|---|
| r15.1 | 1000787 | 4.63 | 0.00 | 0.17 | 0.00 | 0.00 | 1637.8 |
| r15.2 | 1974403 | 8.54 | 0.17 | 0.00 | 0.07 | 0.07 | 36067.0 |
| r15.3 | 2895321 | 7.67 | 0.47 | 0.28 | 0.00 | 0.00 | 36155.7 |
| r15.4 | 1148604 | 5.47 | 0.06 | 0.22 | 0.05 | 0.00 | 36086.6 |
| r15.5 | 2484342 | 9.88 | 0.00 | 1.05 | 0.29 | 0.21 | 36065.3 |
| r15.6 | 3834074 | 12.56 | 0.42 | 1.13 | 0.00 | 0.00 | 36093.8 |
| r15.7 | 2301798 | 2.29 | 0.00 | 0.61 | 0.07 | 0.07 | 36031.3 |
| r15.8 | 5579451 | 5.85 | 0.04 | 0.91 | 0.01 | 0.00 | 36029.5 |
| r15.9 | 8696932 | 5.27 | 0.00 | 0.37 | 0.00 | 0.00 | 210.1 |
| r16.1 | 136161 | 0.28 | 0.00 | 0.00 | 0.00 | 0.00 | 0.1 |
| r16.2 | 239500 | 3.30 | 0.30 | 1.11 | 0.00 | 0.00 | 782.2 |
| r16.3 | 325671 | 3.88 | 0.05 | 0.85 | 0.00 | 0.00 | 1251.8 |
| r16.4 | 138532 | 1.03 | 0.00 | 0.00 | 0.00 | 0.00 | 17.4 |
| r16.5 | 241801 | 1.71 | 0.00 | 1.24 | 0.00 | 0.00 | 174.8 |
| r16.6 | 337762 | 5.02 | 1.42 | 1.56 | 0.00 | 0.00 | 584.3 |
| r16.7 | 169336 | 1.70 | 2.34 | 0.81 | 0.00 | 0.00 | 36272.0 |
| r16.8 | 348167 | 4.67 | 3.03 | 3.24 | 0.00 | 0.00 | 36205.0 |
| r16.9 | 530911 | 6.84 | 2.56 | 3.09 | 0.00 | 0.00 | 36189.1 |
| r17.1 | 354138 | 4.31 | 0.02 | 0.00 | 0.00 | 0.00 | 94.4 |
| r17.2 | 645488 | 6.25 | 1.50 | 2.86 | 0.00 | 0.00 | 2225.3 |
| r17.3 | 910518 | 6.24 | 1.04 | 1.64 | 3.12 | 0.00 | 28647.2 |
| r17.4 | 370590 | 2.69 | 0.01 | 0.34 | 0.00 | 0.00 | 1586.0 |
| r17.5 | 706747 | 6.17 | 0.78 | 1.25 | 0.00 | 0.00 | 34188.6 |
| r17.6 | 1022696 | 7.71 | 0.98 | 0.93 | 0.00 | 0.00 | 36139.7 |
| r17.7 | 502144 | 4.18 | 0.49 | 1.34 | 0.00 | 0.00 | 36110.9 |
| r17.8 | 1110945 | 7.04 | 0.03 | 0.38 | 0.00 | 0.00 | 36116.0 |
| r17.9 | 1787898 | 8.08 | 0.98 | 1.41 | 0.27 | 0.00 | 36153.5 |
| r18.1 | 828119 | 5.13 | 0.40 | 0.32 | 0.35 | 0.00 | 36113.6 |
| r18.2 | 1533675 | 10.66 | 0.84 | 0.31 | 0.00 | 0.00 | 7880.1 |
| r18.3 | 2182638 | 8.20 | 1.01 | 0.11 | 0.00 | 0.00 | 36071.0 |
| r18.4 | 923609 | 5.31 | 0.00 | 0.48 | 0.06 | 0.06 | 36098.1 |
| r18.5 | 1829317 | 10.24 | 0.19 | 0.43 | 0.00 | 0.00 | 36085.6 |
| r18.6 | 2704287 | 8.84 | 1.22 | 1.53 | 1.71 | 0.00 | 36088.5 |
| r18.7 | 1483219 | 8.59 | 0.00 | 0.56 | 0.15 | 0.05 | 36186.5 |
| r18.8 | 3898384 | 14.82 | 0.40 | 2.85 | 0.00 | 0.00 | 36049.5 |
| r18.9 | 6376813 | 15.02 | 0.51 | 1.90 | 0.00 | 0.00 | 24039.5 |

Table D.5: Branch-and-Cut heuristic detailed results for benchmark instances Canad-R : r14 to r18

# Bibliography


[1] Achterberg, T., and Berthold, T. Improving the feasibility pump. *Discrete Optimization 4*, 1 (2007), 77–86.

[2] Achterberg, T., Koch, T., and Martin, A. Branching rules revisited. *Operations Research Letters 33*, 1 (2005), 42–54.

[3] Agarwal, Y. K. k-partition-based facets of the network design problem. *Networks: An International Journal 47*, 3 (2006), 123–139.

[4] Agarwal, Y. K., and Aneja, Y. P. Fixed charge multicommodity network design using p-partition facets. *European Journal of Operational Research 258*, 1 (2017), 124–135.

[5] Ahuja, R., Magnanti, T., and Orlin, J. *Network Flows: Theory, Algorithms, and Applications*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1993.

[6] Ahuja, R. K., Magnanti, T. L., Orlin, J. B., and Reddy, M. R. Applications of network optimization. In *Network Models*, M. O. Ball, T. L. Magnanti, C. L. Monma, and G. L. Nemhauser, Eds., vol. 7 of *Handbooks in Operations Research and Management Science*. Elsevier, 1995, ch. 1, pp. 1–83.

[7] Alvarez, A. M., González-Velarde, J. L., and De-Alba, K. GRASP embedded scatter search for the multicommodity capacitated network design problem. *Journal of Heuristics 11*, 3 (2005), 233–257.

[8] Alvarez, A. M., González-Velarde, J. L., and De-Alba, K. Scatter search for network design problem. *Annals of Operations Research 138*, 1 (2005), 159–178.

[9] Álvarez-Miranda, E., and Sinnl, M. A relax-and-cut framework for large-scale maximum weight connected subgraph problems. *Computers & Operations Research 87* (2017), 63–82.

[10] Armacost, A. P., Barnhart, C., and Ware, K. A. Composite variable formulations for express shipment service network design. *Transportation science 36*, 1 (2002), 1–20.

[11] Armacost, A. P., Barnhart, C., Ware, K. A., and Wilson, A. M. UPS optimizes its air network. *Interfaces 34*, 1 (2004), 15–25.

[12] Assad, A. A. Multicommodity network flows—a survey. *Networks 8*, 1 (1978), 37–91.

[13] Atamtürk, A. Flow pack facets of the single node fixed-charge flow polytope. *Operations Research Letters 29*, 3 (2001), 107–114.

[14] Atamtürk, A. On capacitated network design cut–set polyhedra. *Mathematical Programming 92*, 3 (2002), 425–437.

[15] Atamtürk, A. Cover and pack inequalities for (mixed) integer programming. *Annals of Operations Research 139*, 1 (2005), 21–38.

[16] Atamtürk, A., Gómez, A., and Küçükyavuz, S. Three-partition flow cover inequalities for constant capacity fixed-charge network flow problems. *Networks 67*, 4 (2016), 299–315.

[17] Atamtürk, A., and Rajan, D. On splittable and unsplittable flow capacitated network design arc–set polyhedra. *Mathematical Programming 92*, 2 (2002), 315–333.

[18] Avella, P., Mattia, S., and Sassano, A. Metric inequalities and the network loading problem. *Discrete Optimization 4*, 1 (2007), 103–114.

[19] Bahiense, L., Maculan, N., and Sagastizábal, C. The volume algorithm revisited: relation with bundle methods. *Mathematical Programming 94* (2002), 41–60.

[20] Balakrishnan, A. LP extreme points and cuts for the fixed-charge network design problem. *Mathematical programming 39*, 3 (1987), 263–284.

[21] Balakrishnan, A., Magnanti, T. L., and Wong, R. T. A dual-ascent procedure for large-scale uncapacitated network design. *Operations Research 37*, 5 (1989), 716–740.

[22] Balas, E. Facets of the knapsack polytope. *Mathematical Programming 8*, 1 (1975), 146–164.

[23] Balas, E., and Christofides, N. A restricted Lagrangean approach to the traveling salesman problem. *Mathematical Programming 21*, 1 (1981), 19–46.

[24] Balas, E., and Zemel, E. Facets of the knapsack polytope from minimal covers. *SIAM Journal on Applied Mathematics 34*, 1 (1978), 119–148.

[25] Barahona, F. Network design using cut inequalities. *SIAM Journal on optimization 6*, 3 (1996), 823–837.

[26] Barahona, F., and Anbil, R. The volume algorithm: producing primal solutions with a subgradient method. *Mathematical Programming 87* (2000), 385–399.

[27] Barahona, F., and Ladányi, L. Branch and cut based on the volume algorithm: Steiner trees in graphs and max-cut. *RAIRO-Operations Research 40*, 1 (2006), 53–73.

[28] Barnhart, C., Hane, C. A., and Vance, P. H. Using branch-and-price-and-cut to solve origin-destination integer multicommodity flow problems. *Operations Research 48*, 2 (2000), 318–326.

[29] Bektaş, T., Chouman, M., and Crainic, T. G. Lagrangean-based decomposition algorithms for multicommodity network design problems with penalized constraints. *Networks: An International Journal 55*, 3 (2010), 171–180.

[30] Belotti, P., Malucelli, F., and Brunetta, L. Multicommodity network design with discrete node costs. *Networks: An International Journal 49*, 1 (2007), 90–99.

[31] Benders, J. F. Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik 4*, 1 (1962), 238–252.

[32] Bénichou, M., Gauthier, J.-M., Girodet, P., Hentges, G., Ribière, G., and Vincent, O. Experiments in mixed-integer linear programming. *Mathematical Programming 1*, 1 (1971), 76–94.

[33] Bertacco, L., Fischetti, M., and Lodi, A. A feasibility pump heuristic for general mixed-integer problems. *Discrete Optimization 4*, 1 (2007), 63–76.

[34] Bessiere, C. Constraint propagation. In *Handbook of Constraint Programming*, F. Rossi, P. [van Beek], and T. Walsh, Eds., vol. 2 of *Foundations of Artificial Intelligence*. Elsevier, 2006, pp. 29–83.

[35] Bienstock, D. *Potential Function Methods for Approximately Solving Linear Programming Problems: Theory and Practice*. International Series in Operations Research & Management Science. Kluwer Academic Publishers, 2002.

[36] Bienstock, D., Chopra, S., Günlük, O., and Tsai, C.-Y. Minimum cost capacity installation for multicommodity network flows. *Mathematical programming 81*, 2 (1998), 177–199.

[37] Bienstock, D., and Günlük, O. Computational experience with a difficult mixed-integer multicommodity flow problem. *Mathematical Programming 68*, 1-3 (1995), 213–237.

[38] Bienstock, D., and Günlük, O. Capacitated network design—polyhedral structure and computation. *Informs journal on Computing 8*, 3 (1996), 243–259.

[39] Bilde, O., and Krarup, J. Sharp lower bounds and efficient algorithms for the simple plant location problem. In *Studies in Integer Programming*, P. L. Hammer, E. L. Johnson, B. H. Korte, and G. L. Nemhauser, Eds., vol. 1 of *Annals of Discrete Mathematics*. Elsevier, 1977, pp. 79–97.

[40] Bonnans, J. F., Gilbert, J. C., Lemaréchal, C., and Sagastizábal, C. A. *Numerical Optimization - Theoretical and Practical Aspects*. Universitext. Springer-Verlag Berlin Heidelberg, 2006.

[41] Boyd, S. P., and Vandenberghe, L. *Convex optimization*. Cambridge University Press, 2004.

[42] Briant, O., Lemaréchal, C., Meurdesoif, P., Michel, S., Perrot, N., and Vanderbeck, F. Comparison of bundle and classical column generation. *Mathematical Programming 113*, 2 (2008), 299–344.

[43] Castro, J. Solving difficult multicommodity problems with a specialized interior-point algorithm. *Annals of Operations Research 124*, 1-4 (2003), 35–48.

[44] Cavalcante, V. F., de Souza, C. C., and Lucena, A. A relax-and-cut algorithm for the set partitioning problem. *Computers & Operations Research 35*, 6 (2008), 1963–1981.

[45] Chouman, M., and Crainic, T. A MIP-tabu search hybrid framework for multicommodity capacitated fixed-charge network design. Tech. Rep. 31, CIRRELT, 2010.

[46] Chouman, M., Crainic, T. G., and Gendron, B. A cutting-plane algorithm based on cutset inequalities for multicommodity capacitated fixed charge network design. Tech. rep., CIRRELT, 2003.

[47] Chouman, M., Crainic, T. G., and Gendron, B. Revue des inégalités valides pertinentes aux problèmes des conception de réseaux. *INFOR: Information Systems and Operational Research 41*, 1 (2003), 5–33.

[48] Chouman, M., Crainic, T. G., and Gendron, B. A cutting-plane algorithm for multicommodity capacitated fixed-charge network design. Tech. rep., CIRRELT, 2009.

[49] Chouman, M., Crainic, T. G., and Gendron, B. Commodity representations and cut-set-based inequalities for multicommodity capacitated fixed-charge network design. *Transportation Science 51*, 2 (2017), 650–667.

[50] Chouman, M., Crainic, T. G., and Gendron, B. The impact of filtering in a branch-and-cut algorithm for multicommodity capacitated fixed charge network design. *EURO Journal on Computational Optimization 6*, 2 (2018), 143–184.

[51] Chung, C.-S., Hung, M. S., and Rom, W. O. A hard knapsack problem. *Naval Research Logistics (NRL) 35*, 1 (1988), 85–98.

[52] Chvátal, V. Hard knapsack problems. *Operations Research 28*, 6 (1980), 1402–1411.

[53] Codato, G., and Fischetti, M. Combinatorial Benders' cuts for mixed-integer linear programming. *Operations Research 54*, 4 (2006), 756–766.

[54] Contreras, I., and Fernández, E. General network design: A unified view of combined location and network design problems. *European Journal of Operational Research 219*, 3 (2012), 680–697.

[55] Cordeau, J.-F., Pasin, F., and Solomon, M. M. An integrated model for logistics network design. *Annals of Operations Research 144*, 1 (2006), 59–82.

[56] Cornuejols, G., Sridharan, R., and Thizy, J. A comparison of heuristics and relaxations for the capacitated plant location problem. *European Journal of Operational Research 50*, 3 (1991), 280–297.

[57] Costa, A. M. A survey on Benders decomposition applied to fixed-charge network design problemsenders decomposition applied to fixed-charge network design problems. *Computers & Operations Research 32*, 6 (2005), 1429–1450.

[58] Costa, A. M., Cordeau, J.-F., and Gendron, B. Benders, metric and cutset inequalities for multicommodity capacitated network design. *Computational Optimization and Applications 42*, 3 (2009), 371–392.

[59] Costa, A. M., Cordeau, J.-F., Gendron, B., and Laporte, G. Accelerating Benders decomposition with heuristic master problem solutions. *Pesquisa Operacional 32*, 1 (2012), 03–20.

[60] Crainic, T. G. Service network design in freight transportation. *European Journal of Operational Research 122*, 2 (2000), 272–288.

[61] Crainic, T. G., Frangioni, A., and Gendron, B. Bundle-based relaxation methods for multicommodity capacitated fixed charge network design. *Discrete Applied Mathematics 112*, 1-3 (2001), 73–99.

[62] Crainic, T. G., and Gendreau, M. Cooperative parallel tabu search for capacitated network design. *Journal of Heuristics 8*, 6 (2002), 601–627.

[63] Crainic, T. G., Gendreau, M., and Farvolden, J. M. A simplex-based tabu search method for capacitated network design. *INFORMS journal on Computing 12*, 3 (2000), 223–236.

[64] Crainic, T. G., Gendron, B., and Hernu, G. A slope scaling/Lagrangean perturbation heuristic with long-term memory for multicommodity capacitated fixed-charge network design. *Journal of Heuristics 10*, 5 (2004), 525–545.

[65] Crainic, T. G., Li, Y., and Toulouse, M. A first multilevel cooperative algorithm for capacitated multicommodity network design. *Computers & Operations Research 33*, 9 (2006), 2602–2622.

[66] Croxton, K. L., Gendron, B., and Magnanti, T. L. A comparison of mixed-integer programming models for nonconvex piecewise linear cost minimization problems. *Management Science 49*, 9 (2003), 1268–1273.

[67] Croxton, K. L., Gendron, B., and Magnanti, T. L. Variable disaggregation in network flow problems with piecewise linear costs. *Operations Research 55*, 1 (2007), 146–157.

[68] da Cunha, A. S., Lucena, A., Maculan, N., and Resende, M. G. A relax-and-cut algorithm for the prize-collecting Steiner problem in graphs. *Discrete Applied Mathematics 157*, 6 (2009), 1198–1217.

[69] Dantzig, G. B., and Wolfe, P. Decomposition principle for linear programs. *Operations Research 8*, 1 (1960), 101–111.

[70] de Paula, M. R., Mateus, G. R., and Ravetti, M. G. A non-delayed relax-and-cut algorithm for scheduling problems with parallel machines, due dates and sequence-dependent setup times. *Computers & Operations Research 37*, 5 (2010), 938–949.

[71] De Santis, M., Lucidi, S., and Rinaldi, F. A new class of functions for measuring solution integrality in the feasibility pump approach. *SIAM Journal on Optimization 23*, 3 (2013), 1575–1606.

[72] de Souza, M. C., Mahey, P., and Gendron, B. Cycle-based algorithms for multicommodity network flow problems with separable piecewise convex costs. *Networks: An International Journal 51*, 2 (2008), 133–141.

[73] Desrosiers, J., and Lübbecke, M. E. Branch-price-and-cut algorithms. In *Wiley Encyclopedia of Operations Research and Management Science*, J. J. Cochran, Ed. John Wiley & Sons, Inc., 2011, pp. 1–13.

[74] Dongarra, J. J. Performance of various computers using standart linear equations software. Tech. Rep. CS-89-85, University of Manchester, 2014.

[75] Erlenkotter, D. A dual-based procedure for uncapacitated facility location. *Operations Research 26*, 6 (1978), 992–1009.

[76] Escudero, L. F., Garín, M. A., and Unzueta, A. Cluster Lagrangean decomposition in multistage stochastic optimization. *Computers & Operations Research 67* (2016), 48–62.

[77] Escudero, L. F., Guignard, M., and Malik, K. A Lagrangian relax-and-cut approach for the sequential ordering problem with precedence relationships. *Annals of Operations Research 50*, 1 (1994), 219–237.

[78] Feo, T. A., and Resende, M. G. C. Greedy randomized adaptive search procedures. *Journal of global optimization 6*, 2 (1995), 109–133.

[79] Ferreira, R. P. M., Luna, H. P. L., Mahey, P., and Souza, M. C. d. Global optimization of capacity expansion and flow assignment in multicommodity networks. *Pesquisa Operacional 33*, 2 (2013), 217–234.

[80] Fischetti, M., Glover, F., and Lodi, A. The feasibility pump. *Mathematical Programming 104*, 1 (2005), 91–104.

[81] Fischetti, M., and Lodi, A. Local branching. *Mathematical programming 98*, 1-3 (2003), 23–47.

[82] Fischetti, M., and Salvagnin, D. Feasibility pump 2.0. *Mathematical Programming Computation 1*, 2-3 (2009), 201–222.

[83] Fischetti, M., and Salvagnin, D. A relax-and-cut framework for Gomory mixed-integer cuts. *Mathematical Programming Computation 3*, 2 (2011), 79–102.

[84] Fisher, M. L. Optimal solution of scheduling problems using Lagrange multipliers: Part i. *Operations Research 21*, 5 (1973), 1114–1127.

[85] Fisher, M. L. A dual algorithm for the one-machine scheduling problem. *Mathematical programming 11*, 1 (1976), 229–251.

[86] Fisher, M. L. The Lagrangian relaxation method for solving integer programming problems. *Management science 27*, 1 (1981), 1–18.

[87] Frangioni, A. Solving semidefinite quadratic problems within nonsmooth optimization algorithms. *Computers & Operations Research 23*, 11 (1996), 1099–1118.

[88] Frangioni, A. *Dual-ascent methods and multicommodity flow problems*. PhD thesis, Dipartimento di Informatica, Università di Pisa, 1997.

[89] Frangioni, A. Generalized bundle methods. *SIAM Journal on Optimization 13*, 1 (2002), 117–156.

[90] Frangioni, A. About Lagrangian methods in integer optimization. *Annals of Operations Research 139*, 1 (2005), 163–193.

[91] Frangioni, A. *The NDOSolver + FiOracle Project*, 2013.

[92] Frangioni, A., and Gallo, G. A bundle type dual-ascent approach to linear multicommodity min-cost flow problems. *INFORMS Journal on Computing 11*, 4 (1999), 370–393.

[93] Frangioni, A., and Gendron, B. 0-1 reformulations of the multicommodity capacitated network design problem. *Discrete Applied Mathematics 157*, 6 (2009), 1229–1241.

[94] Frangioni, A., and Gendron, B. A stabilized structured Dantzig–Wolfe decomposition method. *Mathematical Programming 140*, 1 (2013), 45–76.

[95] Frangioni, A., Gendron, B., and Gorgone, E. On the computational efficiency of subgradient methods: a case study with Lagrangian bounds. *Mathematical Programming Computation 9*, 4 (2017), 573–604.

[96] Frangioni, A., and Gorgone, E. Bundle methods for sum-functions with "easy" components: applications to multicommodity network design. *Mathematical Programming, Series A 145*, 1-2 (2014), 133–161.

[97] Gavish, B. Augmented Lagrangean based algorithms for centralized network design. *IEEE Transactions on Communications 33*, 12 (1985), 1247–1257.

[98] Gavish, B. Topological design of telecommunication networks-local access design methods. *Annals of Operations Research 33*, 1 (1991), 17–71.

[99] Geissler, B., Morsi, A., Schewe, L., and Schmidt, M. Penalty alternating direction methods for mixed-integer optimization: A new view on feasibility pumps. *SIAM Journal on Optimization 27*, 3 (2017), 1611–1636.

[100] Gendron, B. A note on a dual ascent approach to the fixed-charge capacitated network design problem. *European Journal of Operational Research 138*, 3 (2002), 671–675.

[101] Gendron, B. Decomposition methods for network design. *Procedia - Social and Behavioral Sciences 20* (2011), 31 – 37.

[102] Gendron, B. Revisiting Lagrangian relaxation for network design. *Discrete Applied Mathematics 261* (2019), 203–218.

[103] Gendron, B., and Crainic, T. G. Relaxations for multicommodity capacitated network design problems. Tech. Rep. CRT-965, Centre de recherche sur les transports, Université de Montréal, 1994.

[104] Gendron, B., and Crainic, T. G. Bounding procedures for multicommodity capacitated fixed charge network design problems. Tech. Rep. CRT-96-06, Centre de recherche sur les transports, Université de Montréal, 1996.

[105] Gendron, B., Crainic, T. G., and Frangioni, A. Multicommodity capacitated network design. In *Telecommunications Network Planning*,

B. Sansò and P. Soriano, Eds., Centre for Research on Transportation. Springer, Boston, MA, Boston, MA, 1999, pp. 1–19.

[106] Gendron, B., Hanafi, S., and Todosijević, R. Matheuristics based on iterative linear programming and slope scaling for multicommodity capacitated fixed charge network design. *European Journal of Operational Research 268*, 1 (2018), 70–81.

[107] Gendron, B., and Larose, M. Branch-and-price-and-cut for large-scale multicommodity capacitated fixed-charge network design. *EURO Journal on Computational Optimization 2*, 1-2 (2014), 55–75.

[108] Geoffrion, A. M. Duality in nonlinear programming: a simplified applications-oriented development. *SIAM review 13*, 1 (1971), 1–37.

[109] Geoffrion, A. M. Lagrangean relaxation for integer programming. In *Approaches to integer programming*, M. L. Balinski, Ed., vol. 2 of *Mathematical Programming Studies*. Springer, 1974, pp. 82–114.

[110] Ghamlouche, I., Crainic, T. G., and Gendreau, M. Cycle-based neighbourhoods for fixed-charge capacitated multicommodity network design. *Operations Research 51*, 4 (2003), 655–667.

[111] Ghamlouche, I., Crainic, T. G., and Gendreau, M. Path relinking, cycle-based neighbourhoods and capacitated multicommodity network design. *Annals of Operations Research 131*, 1-4 (2004), 109–133.

[112] Glover, F. Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research 13*, 5 (1986), 533–549.

[113] Glover, F., Laguna, M., and Martí, R. Fundamentals of scatter search and path relinking. *Control and cybernetics 29*, 3 (2000), 653–684.

[114] Gondzio, J., Du Merle, O., Sarkisian, R., and Vial, J. P. ACCPM - A library for convex optimization based on an analytic center cutting plane method. *European Journal of Operational Research 94*, 1 (1996), 206–211.

[115] Gu, Z., Nemhauser, G. L., and Savelsbergh, M. W. P. Lifted cover inequalities for 0-1 integer programs: Complexity. *INFORMS Journal on Computing 11*, 1 (1999), 117–123.

[116] Gu, Z., Nemhauser, G. L., and Savelsbergh, M. W. P. Lifted flow cover inequalities for mixed 0–1 integer programs. *Mathematical Programming 85*, 3 (1999), 439–467.

[117] Guignard, M. Efficient cuts in Lagrangean 'relax-and-cut' schemes. *European Journal of Operational Research 105*, 1 (1998), 216–223.

[118] Guignard, M. Lagrangean relaxation. *Top 11*, 2 (2003), 151–200.

[119] Günlük, O. A branch-and-cut algorithm for capacitated network design problems. *Mathematical programming 86*, 1 (1999), 17–39.

[120] Hassin, R. On multicommodity flows in planar graphs. *Networks 14*, 2 (1984), 225–235.

[121] Held, M., and Karp, R. M. The traveling-salesman problem and minimum spanning trees. *Operations Research 18*, 6 (1970), 1138–1162.

[122] Held, M., and Karp, R. M. The traveling-salesman problem and minimum spanning trees: Part ii. *Mathematical programming 1*, 1 (1971), 6–25.

[123] Held, M., Wolfe, P., and Crowder, H. P. Validation of subgradient optimization. *Mathematical Programming 6* (1974), 62–88.

[124] Hellstrand, J., Larsson, T., and Migdalas, A. A characterization of the uncapacitated network design polytope. *Operations Research Letters 12*, 3 (1992), 159–163.

[125] Herrmann, J. W., Ioannou, G., Minis, I., and Proth, J. A dual ascent approach to the fixed-charge capacitated network design problem. *European Journal of Operational Research 95*, 3 (1996), 476–490.

[126] Hewitt, M., Nemhauser, G., and Savelsbergh, M. W. P. Branch-and-price guided search for integer programs with an application to the multicommodity fixed-charge network flow problem. *INFORMS Journal on Computing 25*, 2 (2013), 302–316.

[127] Hewitt, M., Nemhauser, G. L., and Savelsbergh, M. W. P. Combining exact and heuristic approaches for the capacitated fixed-charge network flow problem. *INFORMS Journal on Computing 22*, 2 (2010), 314–325.

[128] Holmberg, K., Joborn, M., and Melin, K. Lagrangian based heuristics for the multicommodity network flow problem with fixed costs on paths. *European Journal of Operational Research 188*, 1 (2008), 101–108.

[129] Holmberg, K., and Yuan, D. A Lagrangean approach to network design problems. *International Transactions in Operational Research 5*, 6 (1998), 529–539.

[130] Holmberg, K., and Yuan, D. A Lagrangian heuristic based branch-and-bound approach for the capacitated network design problem. *Operations Research 48*, 3 (2000), 461–481.

[131] Holmberg, K., and Yuan, D. A multicommodity network-flow problem with side constraints on paths solved by column generation. *INFORMS Journal on Computing 15*, 1 (2003), 42–57.

[132] Hu, T. C. Multi-commodity network flows. *Operations Research 11*, 3 (1963), 344–360.

[133] Hunter, D. R., and Lange, K. A tutorial on MM algorithms. *The American Statistician 58*, 1 (2004), 30–37.

[134] Iri, M. On an extension of the maximum-flow minimum-cut theorem to multicommodity flows. *Journal of the Operations Research Society of Japan 13*, 3 (1971), 129–135.

[135] Jaillet, P., Song, G., and Yu, G. Airline network design and hub location problems. *Location science 4*, 3 (1996), 195–212.

[136] Johnson, D. S., Lenstra, J. K., and Kan, A. R. The complexity of the network design problem. *Networks 8*, 4 (1978), 279–285.

[137] Jones, K. L., Lustig, I. J., Farvolden, J. M., and Powell, W. B. Multicommodity network flows: The impact of formulation on decomposition. *Mathematical Programming 62*, 1-3 (1993), 95–117.

[138] Jörnsten, K., and Näsberg, M. A new Lagrangian relaxation approach to the generalized assignment problem. *European Journal of Operational Research 27*, 3 (1986), 313–323.

[139] Jukna, S., and Schnitger, G. Yet harder knapsack problems. *Theoretical computer science 412*, 45 (2011), 6351–6358.

[140] Katayama, N. A combined capacity scaling and local branching approach for capacitated multi-commodity network design problem. *Far East Journal of Applied Mathematics 92*, 1 (2015), 1–30.

[141] Katayama, N., Chen, M., and Kubo, M. A capacity scaling heuristic for the multicommodity capacitated network design problem. *Journal of computational and applied mathematics 232*, 1 (2009), 90–101.

[142] Kazemzadeh, M. R. A., Bektaş, T., Crainic, T. G., Frangioni, A., Gendron, B., and Gorgone, E. Node-based Lagrangian relaxations for multicommodity capacitated fixed-charge network design. Tech. Rep. 21, CIRRELT, June 2019.

[143] Keha, A. B., de Farias Jr, I. R., and Nemhauser, G. L. Models for representing piecewise linear cost functions. *Operations Research Letters 32*, 1 (2004), 44–48.

[144] Keha, A. B., de Farias Jr, I. R., and Nemhauser, G. L. A branch-and-cut algorithm without binary variables for nonconvex piecewise linear optimization. *Operations Research 54*, 5 (2006), 847–858.

[145] Kelley, J. E. The cutting-plane method for solving convex programs. *Journal of the Society for Industrial and Applied Mathematics 8*, 4 (1960), 703–712.

[146] Kennington, J. L. A survey of linear cost multicommodity network flows. *Operations Research 26*, 2 (1978), 209–236.

[147] Khachian, L. G. Polynomial algorithms in linear programming. *USSR Computational Mathematics and Mathematical Physics 20*, 1 (1980), 53–72.

[148] Kim, D., and Pardalos, P. M. A solution approach to the fixed charge network flow problem using a dynamic slope scaling procedure. *Operations Research Letters 24*, 4 (1999), 195–203.

[149] Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. Optimization by simulated annealing. *Science 220*, 4598 (1983), 671–680.

[150] Kliewer, G., and Timajev, L. Relax-and-cut for capacitated network design. In *Algorithms – ESA 2005* (2005), G. S. Brodal and S. Leonardi, Eds., vol. 3669 of *Lecture Notes in Computer Science*, Springer, Berlin, Heidelberg, pp. 47–58.

[151] Klose, A. A Lagrangean relax-and-cut approach for the two-stage capacitated facility location problem. *European Journal of Operational Research 126*, 2 (2000), 408–421.

[152] Kochetov, Y., and Ivanenko, D. Computationally difficult instances for the uncapacitated facility location problem. In *Metaheuristics: Progress as real problem solvers*, T. Ibaraki, K. Nonobe, and M. Yagiura, Eds., vol. 32 of *Operations Research/Computer Science Interfaces Series*. Springer, 2005, pp. 351–367.

[153] Lawler, E. L., and Wood, D. E. Branch-and-bound methods: A survey. *Operations Research 14*, 4 (1966), 699–719.

[154] Lederer, P. J., and Nambimadom, R. S. Airline network design. *Operations Research 46*, 6 (1998), 785–804.

[155] Lemaréchal, C. Nondifferentiable optimization. In *Optimization*, G. L. Nemhauser, A. H. G. R. Kan, and M. J. Todd, Eds., vol. 1 of *Handbooks in Operations Research and Management Science*. Elsevier, 1989, ch. VII, pp. 529–572.

[156] Lemaréchal, C. Lagrangean relaxation. In *Computational Combinatorial Optimization*, M. Jünger and D. Naddef, Eds., vol. 2241 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2001.

[157] Letchford, A. N., and Souli, G. New valid inequalities for the fixed-charge and single-node flow polytopes. *Operations Research Letters 47*, 5 (2019), 353–357.

[158] Letchford, A. N., and Souli, G. Valid inequalities for mixed-integer programs with fixed charges on sets of variables. *Operations Research Letters 48*, 3 (2020), 240–244.

[159] Lucena, A. Non delayed relax-and-cut algorithms. *Annals of Operations Research 140*, 1 (2005), 375–410.

[160] Lucena, A. Lagrangian relax-and-cut algorithms. In *Handbook of Optimization in Telecommunications*, M. G. C. Resende and P. M. Pardalos, Eds. Springer, Boston, MA, 2006, pp. 129–145.

[161] Luna, H. P. L., and Mahey, P. Bounds for global optimization of capacity expansion and flow assignment problems. *Operations Research Letters 26*, 5 (2000), 211–216.

[162] Magnanti, T. L., Mirchandani, P., and Vachani, R. The convex hull of two core capacitated network design problems. *Mathematical programming 60*, 1-3 (1993), 233–250.

[163] Magnanti, T. L., Mirchandani, P., and Vachani, R. Modeling and solving the two-facility capacitated network loading problem. *Operations Research 43*, 1 (1995), 142–157.

[164] Magnanti, T. L., and Wong, R. T. Network design and transportation planning: Models and algorithms. *Transportation science 18*, 1 (1984), 1–55.

[165] Mahey, P., Benchakroun, A., and Boyer, F. Capacity and flow assignment of data networks by generalized Benders decomposition. *Journal of Global Optimization 20*, 2 (2001), 169–189.

[166] Mahey, P., and de Souza, M. C. Local optimality conditions for multicommodity flow problems with separable piecewise convex costs. *Operations Research Letters 35*, 2 (2007), 221–226.

[167] Mahey, P., and Souza, M. C. d. Multicommodity network flows with nonconvex arc costs. *Pesquisa Operacional 37*, 3 (2017), 571–595.

[168] Martello, S., Pisinger, D., and Toth, P. Dynamic programming and strong bounds for the 0-1 knapsack problem. *Management Science 45*, 3 (1999), 414–424.

[169] Martello, S., and Toth, P. Upper bounds and algorithms for hard 0-1 knapsack problems. *Operations Research 45*, 5 (1997), 768–778.

[170] Martin, A. General mixed integer programming: Computational issues for branch-and-cut algorithms. In *Computational combinatorial optimization*, M. Jünger and D. Naddef, Eds., vol. 2241 of *Lecture Notes in Computer Science*. Springer, Berlin, Heidelberg, 2001, pp. 1–25.

[171] Minoux, M. Networks synthesis and optimum network design problems: Models, solution methods and applications. *Networks 19*, 3 (1989), 313–360.

[172] Minoux, M. Discrete cost multicommodity network optimization problems and exact solution methods. *Annals of Operations Research 106*, 1-4 (2001), 19–46.

[173] Mitchell, J. E. Branch-and-cut algorithms for combinatorial optimization problems. *Handbook of applied optimization 1* (2002), 65–77.

[174] Mitten, L. G. Branch-and-bound methods: General formulation and properties. *Operations Research 18*, 1 (1970), 24–34.

[175] Momeni, M., and Sarmadi, M. A genetic algorithm based on relaxation induced neighborhood search in a local branching framework for capacitated multicommodity network design. *Networks and Spatial Economics 16*, 2 (2016), 447–468.

[176] Munguía, L.-M., Ahmed, S., Bader, D. A., Nemhauser, G. L., Goel, V., and Shao, Y. A parallel local search framework for the fixed-charge multicommodity network flow problem. *Computers & Operations Research 77* (2017), 44–57.

[177] Nemhauser, G. L., and Wolsey, L. A. *Integer and Combinatorial Optimization*. Wiley-Interscience Series in Discrete Mathematics and Optimization. John Wiley & Sons, Inc, 1988.

[178] Okamura, H., and Seymour, P. D. Multicommodity flows in planar graphs. *Journal of Combinatorial Theory, Series B 31*, 1 (1981), 75–81.

[179] Oliveira, W., and Sagastizábal, C. Bundle methods in the XXIst century: a birds's-eye view. *Pesquisa Operacional* (2014).

[180] Onaga, K., and Kakusho, O. On feasibility conditions of multicommodity flows in networks. *IEEE Transactions on Circuit Theory 18*, 4 (1971), 425–429.

[181] Ouorou, A., Mahey, P., and Vial, J.-P. A survey of algorithms for convex multicommodity flow problems. *Management science 46*, 1 (2000), 126–147.

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

[183] Padberg, M. W., Van Roy, T. J., and Wolsey, L. A. Valid linear inequalities for fixed charge problems. *Operations Research 33*, 4 (1985), 842–861.

[184] Paraskevopoulos, D. C., Bektaş, T., Crainic, T. G., and Potts, C. N. A cycle-based evolutionary algorithm for the fixed-charge capacitated multi-commodity network design problem. *European Journal of Operational Research 253*, 2 (2016), 265–279.

[185] Paraskevopoulos, D. C., Gürel, S., and Bektaş, T. The congested multicommodity network design problem. *Transportation Research Part E: Logistics and Transportation Review 85* (2016), 166–187.

[186] Pisinger, D. A minimal algorithm for the 0-1 knapsack problem. *Operations Research 45*, 5 (1997), 758–767.

[187] Pisinger, D. Where are the hard knapsack problems? *Computers & Operations Research 32*, 9 (2005), 2271–2284.

[188] Polyak, B. T. Minimization of unsmooth functionals. *USSR Computational Mathematics and Mathematical Physics 9*, 3 (1969), 14–29.

[189] Raack, C., Koster, A. M. C. A., Orlowski, S., and Wessäly, R. On cut-based inequalities for capacitated network design polyhedra. *Networks 57*, 2 (2011), 141–156.

[190] Rahmaniani, R., Crainic, T. G., Gendreau, M., and Rei, W. The Benders decomposition algorithm: A literature review. *European Journal of Operational Research 259*, 3 (2017), 801–817.

[191] Rardin, R. L., and Wolsey, L. A. Valid inequalities and projecting the multicommodity extended formulation for uncapacitated fixed charge network flow problems. *European Journal of Operational Research 71*, 1 (1993), 95–109.

[192] Rockafellar, R. T. *Convex analysis*. No. 28 in Princeton Landmarks in Mathematics and Physics. Princeton university press, Princeton, New Jersey, 1970.

[193] Rodríguez-Martín, I., and Salazar-González, J. J. A local branching heuristic for the capacitated fixed-charge network design problem. *Computers & Operations Research 37*, 3 (2010), 575–581.

[194] Ross, G. T., and Soland, R. M. A branch and bound algorithm for the generalized assignment problem. *Mathematical programming 8*, 1 (1975), 91–103.

[195] Rothschild, B., and Whinston, A. Feasibility of two commodity network flows. *Operations Research 14*, 6 (1966), 1121–1129.

[196] Sakarovitch, M. Two commodity network flows and linear programming. *Mathematical Programming 4*, 1 (1973), 1–20.

[197] Sellmann, M., Kliewer, G., and Koberstein, A. Lagrangian cardinality cuts and variable fixing for capacitated network design. In *Algorithms — ESA 2002* (2002), R. Möhring and R. Raman, Eds., vol. 2461 of *Lecture Notes in Computer Science*, Springer, Berlin, Heidelberg, pp. 845–858.

[198] Sherali, H. D., and Choi, G. Recovery of primal solutions when using subgradient optimization methods to solve Lagrangian duals of linear programs. *Operations Research Letters 19*, 3 (1996), 105–113.

[199] Shibasaki, R. S., Baïou, M., Barahona, F., Mahey, P., and Souza, M. C. d. Lagrangian bounds for large-scale multicommodity network design: a comparison between volume and bundle methods. *International Transactions in Operational Research*, to appear (2020).

[200] Shor, N. Z. *Minimization Methods for Non-Differentiable Functions*, vol. 3 of *Springer Series in Computational Mathematics*. Springer-Verlag Berlin Heidelberg, 1985.

[201] Smith-Miles, K., and Lopes, L. Measuring instance difficulty for combinatorial optimization problems. *Computers & Operations Research 39*, 5 (2012), 875–889.

[202] Sridhar, V., and Park, J. S. Benders-and-cut algorithm for fixed-charge capacitated network design problem. *European Journal of Operational Research 125*, 3 (2000), 622–632.

[203] Stallaert, J. I. A. The complementary class of generalized flow cover inequalities. *Discrete Applied Mathematics 77*, 1 (1997), 73–80.

[204] Tomlin, J. Minimum-cost multicommodity network flows. *Operations Research 14*, 1 (1966), 45–51.

[205] Van Roy, T. J., and Wolsey, L. A. Valid inequalities for mixed 0–1 programs. *Discrete Applied Mathematics 14*, 2 (1986), 199 – 213.

[206] Vu, D. M., Crainic, T. G., and Toulouse, M. A three-phase matheuristic for capacitated multi-commodity fixed-cost network design with design-balance constraints. *Journal of heuristics 19*, 5 (2013), 757–795.

[207] Wolfe, P. A method of conjugate subgradients for minimizing nondifferentiable functions. In *Nondifferentiable Optimization*, M. L. Balinski and P. Wolfe, Eds., vol. 3 of *Mathematical Programming Studies*. Springer Berlin Heidelberg, 1975.

[208] Yaghini, M., and Akhavan, R. Multicommodity network design problem in rail freight transportation planning. *Procedia-Social and Behavioral Sciences 43* (2012), 728–739.

[209] Yaghini, M., Karimi, M., Rahbar, M., and Sharifitabar, M. H. A cutting-plane neighborhood structure for fixed-charge capacitated multicommodity network design problem. *INFORMS Journal on Computing 27*, 1 (2015), 48–58.

[210] Yaghini, M., Momeni, M., and Sarmadi, M. A simplex-based simulated annealing algorithm for node-arc capacitated multicommodity network design. *Applied Soft Computing 12*, 9 (2012), 2997–3003.

[211] Yaghini, M., Rahbar, M., and Karimi, M. A hybrid simulated annealing and column generation approach for capacitated multicommodity network design. *Journal of the Operational Research Society 64*, 7 (2013), 1010–1020.

[212] Yazar, B., Arslan, O., Karaşan, O. E., and Kara, B. Y. Fiber optical network design problems: A case for Turkey. *Omega 63* (2016), 23–40.

[213] Zemel, E. Lifting the facets of zero-one polytopes. *Mathematical Programming 15*, 1 (1978), 268–277.