Lagrangian bounds for large-scale multicommodity network design: a comparison between Volume and Bundle methods

Rui Sá Shibasaki, Mourad Baïou, Francisco Barahona, Philippe Mahey, Mauricio Souza

INTERNATIONAL TRANSACTIONS IN OPERATIONAL RESEARCH

WILEY

INTERNATIONAL TRANSACTIONS IN OPERATIONAL RESEARCH

Intl. Trans. in Op. Res. 28 (2021) 296–326
DOI: 10.1111/itor.12770

Lagrangian bounds for large-scale multicommodity network design: a comparison between Volume and Bundle methods

Rui S. Shibasaki[a,b], Mourad Baiou[b], Francisco Barahona[c], Philippe Mahey[b] and Mauricio C. de Souza[d,*]

[a] *PPGEP, Universidade Federal de Minas Gerais, Av. Antônio Carlos, 6627, 31270-901, Belo Horizonte, Brazil*
[b] *LIMOS, Université Clermont-Auvergne, 1 rue de la Chebarde, 63178, Aubière, France*
[c] *Thomas J. Watson Research Center, IBM, 1101 Kitchawan Road, Yorktown Heights, NY, USA*
[d] *DEP, Universidade Federal de Minas Gerais, Av. Antônio Carlos, 6627, 31270-901, Belo Horizonte, Brazil*
E-mail: ruishibasaki@gmail.com [Shibasaki]; baiou@isima.fr [Baiou]; barahon@us.ibm.com [Barahona]; philippe.mahey@isima.fr [Mahey]; prof.mauriciodesouza@gmail.com [de Souza]

Received 1 April 2019; received in revised form 19 December 2019; accepted 20 December 2019

Abstract

The Bundle Method and the Volume Algorithm are among the most efficient techniques to obtain accurate Lagrangian dual bounds for hard combinatorial optimization problems. We propose here to compare their performance 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 of the low computational cost per iteration compared to the Bundle Method.

Keywords: Bundle method; Volume algorithm; fixed-charge multicommodity network design problem

1. Introduction

Lagrangian Relaxation has long been widely used to generate lower bounds for difficult constrained minimization problems and to serve as a basis for developing efficient approximation schemes; see Geoffrion (1974), Lemaréchal (2001), Frangioni (2005) 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. In this paper,

*Corresponding author.

we aim at comparing two classical versions of these algorithms, namely the Bundle method, as proposed by Wolfe and Lemaréchal (Lemaréchal, 1989) and the Volume algorithm proposed by Barahona and Anbil (2000). Comparisons of nonsmooth optimization algorithms can be found in the literature (Briant et al., 2008; Frangioni et al., 2017) 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 work was done in Briant et al. (2008) where the authors compare different algorithms including Bundle, Column Generation, and Volume, for five different problems. The present study is inspired by the work of Lemaréchal (2001) and Frangioni and Gallo (1999) who contributed to the present success of modern Bundle methods that 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. It would be nice to compare with other methods like analytic center (Gondzio et al., 1996) or proximal cutting planes (Ouorou, 2009), deflected (Sherali and Choi, 1996) or majorize-minimize-mean (Hunter and Lange, 2004) subgradient algorithms, just to mention a few, but this is not feasible due to the large amount of programming work that it would require. Moreover, a state-of-the-art review on nondifferentiable optimization (NDO) methods is out of the scope of the present paper and we refer to Bonnans et al. (2006) for a broad history and practical guide on the subject. However, we make our code publicly available, so that other researchers can use it to benchmark their algorithms.

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 Briant et al. (2008), the Bundle algorithm obtained better bounds with fewer iterations, though we believe that its average time per iteration is a bit more expensive than the Volume one. Considering that, the present work focuses on the comparison on the computational total time rather than on the number of iterations.

In addition, Escudero et al. (2012) and Haouari et al. (2008) have also made comparisons. The former tested the performance of the Volume, a variant of the cutting-plane method and two other algorithms for a stochastic problem and concluded that the Volume algorithm provided stronger bounds in less time. The latter paper worked with the prize collecting Steiner tree problem and tested several techniques including multiple variants of deflected subgradient strategies and the Volume algorithm, finally concluding that the Volume was outperformed by the different deflected subgradient algorithms.

We have chosen to compare the performances of both algorithms on large-scale instances of the Fixed-Charge Multicommodity Capacitated Network Design (FCMC) problem because, besides the fact that it is NP-hard, it presents many different characteristics that are favorable to our objectives, such as the presence of different coupling constraints, potential candidates for the relaxation, the decomposable structure induced by these relaxations, and the possibility to build very large instances, out of reach of most exact approaches but with relatively small duality gaps (Crainic et al., 2001). Observe that what we mean by a "numerical comparison" is not only to compare their respective performance to produce good lower bounds, but to analyze their capacity to generate interesting primal fractional solutions (almost feasible for the continuous relaxation of the model) from which feasible mixed-integer solutions may be produced to reduce the gap. In that sense, we contribute to reduce the gap of very large instances of FCMC, unsolved in the literature; see Gendron et al. (1999), Gendron and Larose (2014).

The next sections will present the network design model followed by an explanation of the algorithms. Then, in Section 4, 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 5. Finally, conclusions are made and future work is discussed in Section 6.

2. Fixed-Charge Multicommodity Network Design problem

The Multicommodity Network Design problem consists in minimizing the total cost of multicommodity transport between pairs of origin–destination, so that the demand is satisfied and the arc capacities are respected. In the FCMC, the objective function includes transportation costs for each commodity and arc installation costs, the latter being associated with a single facility of given capacity. Many additional features should be added to model real life network design problems, like the ones faced in telecommunications or transportation networks, but the model is sufficiently challenging and well adapted to our current purpose.

In this paper, it is considered for a given directed graph $G = (N, A)$, N being the set of nodes and A the set of arcs, the problem of minimizing the total cost and satisfying the demands $q^k, k \in K$, associated with a set of pairs origin–destination K called commodities, while respecting the arc capacities $w_e, e \in A$. The total cost is represented by the sum of the transportation cost $c_e^k \geq 0$ for each unit of flow of the commodity k flowing on the arc e and the fixed charge $f_e \geq 0$ to install arc e. A single origin $O(k)$ and destination $D(k)$ are associated with each commodity k.[1] Introducing the variables x_{ij}^k for the flow quantity of commodity k on the arc $e = (ij)$ and the binary variables y_e for the arc installation ($y_e = 1$ if the arc e is used, otherwise $y_e = 0$), the model is presented as follows (Magnanti and Wong, 1984):

$$\text{Minimize} \sum_{k \in K} \sum_{e \in A} c_e^k x_e^k + \sum_{e \in A} f_e y_e; \tag{1}$$

$$\sum_{j \in N_i^+} x_{ij}^k - \sum_{j \in N_i^-} x_{ji}^k = \begin{cases} q^k, & \text{if} \quad i = O(k) \\ -q^k, & \text{if} \quad i = D(k) \\ 0, & \text{otherwise} \end{cases} \forall i \in N, k \in K; \tag{2}$$

$$\sum_{k \in K} x_e^k \leq w_e y_e \qquad \forall e \in A; \tag{3}$$

$$x_e^k \leq b_e^k y_e \qquad \forall e \in A, k \in K; \tag{4}$$

$$x_e^k \geq 0 \qquad \forall e \in A, k \in K; \tag{5}$$

$$y_e \in \{0, 1\} \qquad \forall e \in A; \tag{6}$$

[1] Aggregated commodities by origin or by destination are not considered here

where $N_i^+ = \{j \in N | e = (ij) \in A\}$ is the set of nodes j having an arc arriving from node i and $N_i^- = \{j \in N | e = (ji) \in A\}$ the set of nodes j having an arc arriving into the node i. The constants $b_e^k = \min\{w_e, q^k\}, \ \forall\, e \in A$ and $k \in K$ define upper bounds on the commodity flow through arc e.

The first constraints (2) guarantee the flow conservation in the network, then come the capacity constraints, and finally the domain of the variables. One can note that the *strong forcing constraints* (4) are redundant for the mixed-integer program, but they increase considerably the quality of the lower bound when solving the linear relaxation of the program (Gendron et al., 1999) as well as being known to be facet-defining for the convex hull of the solutions (Cornuejols et al., 1991). Many other valid inequalities for FCMC have been discussed in the literature, see Chouman et al. (2003) for a complete study, but we will not analyze their addition here.

3. Lagrangian Relaxation and decomposition of the FCMC

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, let us 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 that are to be relaxed (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 $Z_L = \min_x c.x \quad \text{s.t. } Ax = b, \ x \in \text{Conv}(S)$, where $\text{Conv}(S)$ is the convex hull of the set S.

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, that is, to maximize the dual function L on $\mathbb{R}^p$ that is indeed concave on any convex subset of its domain (see, for example, Lemaréchal, 1989). 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 $\text{Conv}(S)$). 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 (Geoffrion, 1974).

Subgradient algorithms were studied by Shor and Polyak in the 1960s (Polyak, 1969) and first applied to the Lagrangian Relaxation of hard combinatorial problems by Held and Karp (1971) in their seminal paper on the Traveling Salesman problem. Further improvements have been proposed later, either by the Russian school (Khachian, 1980; Shor, 1985) or by western researchers (Held et al., 1974). 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). As it can be seen below, Bundle methods and the Volume algorithm share the same strategy as these early methods, extending the direction-finding step to more than two former subgradients. We will now focus on the NDO solvers: the Volume and Bundle algorithms.

3.1. 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, $j \in J$, that is, the extreme points of Conv(S), the dual problem can be written as

$$\begin{aligned} &\text{Maximize} && Z \\ &\text{s.c.:} && Z \leq c.x^j + u.(b - Ax^j) \quad \forall j \in J \\ & && u \in \mathbb{R}^p, Z \in \mathbb{R}, \end{aligned} \tag{7}$$

a linear program with a generally exponential number of constraints that, dualized in its turn, yields the following so-called master program (of the Dantzig–Wolfe decomposition, see Lemaréchal, 1989):

$$\begin{aligned} &\text{Minimize} && \sum_{j \in J} (c.x^j)\lambda_j \\ &\text{s.t.:} && \sum_{j \in J} (Ax^j - b)\lambda_j = 0 \\ & && \sum_{j \in J} \lambda_j = 1 \\ & && \lambda_j \geq 0, \forall j \in J. \end{aligned} \tag{8}$$

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 λ_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 $\{Ax = b, x \in \text{Conv}(S)\}$.

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 $d_v^t \cdot (Ax^t - b) > 0$ (yielding 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, that is, such that

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

As stated in Barahona and Anbil (2000), at the end of an iteration t, $\{\theta, (1-\theta)\theta, (1-\theta)^2\theta, \ldots, (1-\theta)^t\theta\}$ can serve as an approximation for the primal variables $\lambda_1, \ldots, \lambda_t$ of the Dantzig–Wolfe's master problem, with respect to the dual constraints. Furthermore, those λ could be approximated by the volume between the active faces of (7) and the current lower bound $\bar{Z}$, which explains the name of the method. The choice of the key parameters is detailed in Section 5.2.

3.2. Bundle

Bundles were initially presented as extensions of the method of ϵ-subgradients (Wolfe, 1975; Lemaréchal, 1989); nevertheless, recent versions include different backgrounds. It is usual to say that bundle methods are stabilized versions of Kelley's (1960) cutting plane algorithm, 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. Many variants and extensions have been discussed in the literature but it is not the aim of the present paper to survey them. A remarkable work by Lemaréchal et al. (1995) has proposed to cover and analyze most of these stabilized variants under the name of "level methods." Later, Frangioni (2002) introduced a generalized bundle method and a version for cases in which the Lagrangian dual can be decomposed; see also Oliveira and Sagastizábal (2014) for a brief survey on bundle methods.

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

$$L(u) \leq \hat{L}(u) := \min\{L(u_i) + g_i.(u - u_i) : i \in \beta\} \quad \forall u \in \mathbb{R}^p. \tag{9}$$

The model at this point is represented by a group of affine functions that together form an easier NDO 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

$$L_s(\bar{u}) = \max_u \quad \hat{L}(u) - \frac{1}{2s_b}\|u - \bar{u}\|^2, \tag{10}$$

where s_b is the step size in the bundle method.

Assuming the bundle has m parts, thanks to the information transfer property (Lemaréchal, 1989), it is convenient to rewrite the bundle in terms of linearization errors computed at $\bar{u}$, such as $e_i := L(u_i) - L(\bar{u}) + g_i.(\bar{u} - u_i) \quad \forall i = 1, \ldots, m$. Then rewriting the calculation of $L_s(\bar{u})$ as a constrained program with an auxiliary scalar variable η:

$$\begin{aligned} \text{Maximize } \eta \ & -\frac{1}{2s_b}\|u - \bar{u}\|^2 \\ \eta \leq L(\bar{u}) + g_i(u - \bar{u}) + e_i \ & \forall i \in \beta \\ u \in \mathbb{R}^p, \eta \in \mathbb{R} & \end{aligned} \tag{11}$$

and further dualizing (11) with the dual coefficients $\alpha_i \geq 0$, $i \in \beta$, one obtains:

$$\begin{aligned} \text{Minimize} \quad & \frac{s_b}{2}\left\|\sum_{i\in\beta}\alpha_i g_i\right\|^2 + \sum_{i\in\beta}\alpha_i e_i + L(\bar{u}) \\ & \sum_{i\in\beta}\alpha_i = 1 \\ & \alpha_i \geq 0 \ \forall i \in \beta. \end{aligned} \tag{12}$$

The main search procedure is to get, at each iteration t, the solution α^t of (12) 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, that is, denoting u^t the current solution of (11):

$$\Delta_\beta = L(u^t) - L(\bar{u}) = s_b^t\left\|\sum_{i\in\beta}\alpha_i^t g_i\right\|^2 + \sum_{i\in\beta}\alpha_i^t e_i. \tag{13}$$

A weaker version can be obtained considering $L_s(\bar{u})$ the solution value of (12) and taking $\delta_\beta = L_s(\bar{u}) - L(\bar{u})$ as an estimation (see Section 5.2 for more implementation issues on 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. Frangioni (1996) introduced a specially tailored algorithm to solve such quadratic programs (12) in a way to reduce the computational cost and we have used his software in our experiments.

So to conclude this short presentation, it can be observed that both algorithms rely on similar features, namely a combination of subgradients from which a stability center is built for the dual problem and, simultaneously, a tentative primal solution converges to a feasible (but generally fractional) solution. Convergence issues will not be discussed here (see Frangioni, 2002, among others, for the bundle method). It can just be mentioned that the Volume algorithm can be slightly modified to be interpreted as a bundle method and thus inherit its convergence properties (Bahiense et al., 2002).

3.3. Decomposition of the pricing subproblem

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) (3)–(4) inducing a decomposition by commodity of the pricing step into shortest-path subproblems.
- Relaxing the flow conservation constraints (2) inducing a decomposition by arcs into knapsack subproblems.

Direct comparisons on these choices have been made in different works (Gendron et al., 1999; Crainic et al., 2001; Gendron, 2011) 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 (Ouorou et al., 2000) and also considered by Gendron and Larose (2014) for the present case of FCMC.[2]

Even if it depends on the type of instances, the relaxation of the flow conservation constraints has shown a better behavior and these comparisons are generally combined with discussions on the possibility to aggregate commodities by origins (or by destinations) or to force other valid inequalities in the model. We will focus on the model FCMC of Section 2, thus only including the strong forcing inequalities (4).

Let the dual multipliers be denoted by

- $\alpha_e \geq 0, e \in A$ for the capacity constraints (3),
- $\gamma_e^k \geq 0, e \in A, k \in K$ for the strong forcing inequalities (4),
- $\pi_i^k \in \mathbb{R}, i \in N, k \in K$ for the flow conservation Equations (2),

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

3.4. Solving the pricing subproblem

As it is generally considered in the literature to be the most effective strategy, one can choose the relaxation of the flow conservation constraints for illustration purpose. The Lagrange multipliers

[2]New strategies called "node-based Lagrangian Relaxation" (Kazemzadeh et al., 2019) have come recently to our knowledge and it could be valuable to test them on similar instances as the ones reported here.

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 $|A|^2$). The complete Lagrangian dual function is given by

$$L(\pi) = \min_{y} \sum_{e\in A}[f_e + g_e(\pi)]y_e + \sum_{k\in K} q^k \left(\pi^k_{D(k)} - \pi^k_{O(k)}\right), \tag{14}$$

$$y_e \in \{0, 1\},\ \forall e \in A, \tag{15}$$

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

$$g_e(\pi) = \min_{x} \sum_{k\in K} \left(c^k_{ij} + \pi^k_i - \pi^k_j\right) x^k_{ij}, \tag{16}$$

$$\sum_{k\in K} x^k_{ij} \le w_{ij}, \tag{17}$$

$$0 \le x^k_{ij} \le b^k_{ij},\ \forall k \in K. \tag{18}$$

Thus, computation of $g_e(\pi)$ is 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 $(c^k_e + \pi^k_i - \pi^k_j) \le 0$ 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.

4. 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 (14)–(18) at each Lagrangian iteration.

4.1. Initial solution

Let us 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 $0 \le y^{cf}_e \le 1, \forall e \in A$, be the frequency vector of opening arcs, that is, the ratio between the number of Lagrangian iterations 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, the frequency value is used 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 Φ 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, the attractive arcs that remained in set Φ can be identified. Lagrangian subproblems (14)–(18) are then succesively solved 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 $e \notin A_0 \cup \Phi$. As restricted subproblems have fewer arcs and some of them are already set to 1, it is expected that some unfixed arcs may no longer be necessary, while others may become more requested. Solving the FCMC Lagrangian dual with respect to $\Phi \cup A_0$ yields a new vector of frequencies y^f for arcs in Φ. The new frequencies serve as basis to fix arcs in Φ, 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. The value of b is then increased every time there is no change in Φ.

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., (7) if using Volume, or (11) 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 Φ **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, the objective is to 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 Gendron and Larose, 2014). 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, that is, 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 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 the computational experiments, a feasible solution has always been found with this procedure of successive shortest paths, it is important to note that there is no guarantee for a feasible solution to be found. As 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, that situation did never occur in all 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.

4.2. Searching for a high-quality solution

The search for a high-quality solution is performed in two phases. In a first phase, some perturbations are applied to the topology leading to the best solution found so far to build a pool of topologies. In a second phase, some combinations of arcs are generated 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), then 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 $|A_{best}|$, $n = 1, \ldots, 5$ topologies A_n are generated. 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 defined by (14)–(18) with variables y_e set to 1, if $e \in A_n$, 0 otherwise. For each value of p, a topology of minimum cost among the five generated is added to the pool. Thus, at the end of the perturbation phase, the pool has five 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, $n = 1, 2, 3$ topologies A_n are generated, 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, a new topology A_{round} is found. If its cost improves UB, the procedure is applied 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 this 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 $q \leq y^{cf}_e$ **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. A threshold of 0.05 is used, 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^{cf}_e 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. New arc frequencies y^f are computed while solving the FCMC Lagrangian dual defined over $\Phi \cup A_n$ with the arcs in Φ unfixed. Note that the p arcs that were removed from A_n do not belong to Φ. At most

p arcs are selected 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., (7) if using Volume, or (11) if using Bundle)
Given y^f, the frequency vector
Insert in A_n at most p arcs of Φ with the largest values of y^f, $y^f \geq 0.3$
return A_n

5. 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 GHz Intel Xeon machine, 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 Frangioni (2013).

5.1. Instances

Instances were elaborated using the generator available at http://www.di.unipi.it/optimize/Data/MMCF.html. The instance generator receives a number $|N|$ of nodes, a number $|A|$ of arcs, and a number $|K|$ of commodities as parameters. Two nodes are randomly connected until the number of arcs is achieved, with parallel arcs not allowed. A similar procedure is adopted for the commodities.

Costs, capacities, and demands are uniformly distributed in an interval given also as parameter. However, costs and capacities are recomputed in order to obtain different difficulty levels among the instances, as well as proposed in Crainic et al. (2001). Two ratios are used to do so: one for the capacities C and another F for fixed costs.

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

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

Table 1
Characteristics of the instances generated for this study

Group	Nodes	Arcs	Commodities	(C–F) ratios	Binary Var	Continuous Var	Constraints
A	100	1000	2000	(10–0.5), (8–0.5), (14–0.5)	1.0×10^3	2.0×10^6	2.2×10^6
B	100	1000	500	(10–10), (6–10), (14–10)	1.0×10^3	5.0×10^5	5.5×10^5
C	100	1000	800	(2–10), (2–0.001), (14–0.001)	1.0×10^3	8.0×10^5	8.8×10^5
D	100	1200	1000	(1–20), (14–12), (20–0.001)	1.2×10^3	1.2×10^6	1.3×10^6
E	100	2000	2000	(1–20), (1–0.001), (20–0.001)	2.0×10^3	4.0×10^6	4.2×10^6
F	100	5000	7000	(1–20), (1–0.001), (20–20)	5.0×10^3	3.5×10^7	3.6×10^7
G	100	8000	8000	(20–0.001), (1–0.001), (20–20)	8.0×10^3	6.4×10^7	6.5×10^7
H	200	12,000	10,000	(1–20), (1–0.001), (20–20)	1.2×10^4	1.2×10^8	1.2×10^8

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.

Eight groups of very large scale instances were generated for the experiments in this work. Table 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, five randomly instances were generated for each (C–F) ratio. For example, Group A has 15 instances of 100 nodes, 1000 arcs, and 2000 commodities, five 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 (1)–(6) for each instance of the group. The goal has been to test large scale instances with different levels of difficulty. The higher the ratios, the higher is the expectation to get a difficult instance, due to the large importance of fixed charges and capacity restrictions.

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 Frangioni and Gorgone (2014) and are available for download at http://www.di.unipi.it/optimize/Data/MMCF.html. In particular, there are 12 different network sizes, and four instances with different capacity and fixed-charge ratios were generated for each size. Table 2 presents these instances, a more detailed description can be found in Frangioni and Gorgone (2014). Analogously to the previous table, in Table 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 (1)–(6) for each instance of the group.

5.2. Computational settings

The subgradient algorithms were tuned in order to obtain the best trade-off between good-quality bounds and fast convergence. A limit of 1000 iterations and a time limit of 10,000 seconds have been set and different calibrations were tested to keep the best one on average. The chosen configuration was then applied on 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.

Table 2
Characteristics of the benchmark instances considered in this study

Instances	Nodes	Arcs	Commodities	Binary Var	Continuous Var	Constraints
cN/ 01-04	20	299	100	3.0×10^2	3.0×10^4	3.2×10^4
cN/ 05-08	20	298	200	3.0×10^2	6.0×10^4	6.4×10^4
cN/ 09-12	20	297	400	3.0×10^2	1.2×10^5	1.3×10^5
cN/ 13-16	20	300	800	3.0×10^2	2.4×10^5	2.6×10^5
cN/ 17-20	30	599	100	6.0×10^2	6.0×10^4	6.3×10^4
cN/ 21-24	30	599	200	6.0×10^2	1.2×10^5	1.3×10^5
cN/ 25-28	30	598	400	6.0×10^2	2.4×10^5	2.5×10^5
cN/ 29-32	30	597	800	6.0×10^2	4.8×10^5	5.0×10^5
cN/ 33-36	50	1200	100	1.2×10^3	1.2×10^5	1.3×10^5
cN/ 37-40	50	1200	200	1.2×10^3	2.4×10^5	2.5×10^5
cN/ 41-44	50	1200	400	1.2×10^3	4.8×10^5	5.0×10^5
cN/ 45-48	50	1200	800	1.2×10^3	9.6×10^5	1.0×10^6

To calibrate the Bundle implementation, the discussion made by Crainic et al. (2001) 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 new 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 Polyak (1969), 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 ρ, 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, that is, 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 for the Bundle method keeping $s_b^0 = 1.0$ and an initial value $\rho^0 = 0.1$, thanks to 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 $e^t \leq 0.3 \sum_{i \in \beta} \alpha_i e_i$. If the condition is true, the decrease is inhibited, assuming that an accurate first-order information has been gathered.

A minimum of four consecutive iterations without improvement before reducing the stepsize was established in both algorithms. It means that ρ is diminished after $nr_v = 4$ red iterations and s_b^t after $nn_b = 4$ null steps. Inversely, both ρ 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 four consecutive yellow iterations before ρ 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 ρ 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 Frangioni (1997) for a detailed explanation. Indeed, to update s_b^t, the maximizer of the quadratic one-dimensional function $\phi(s) = L(\bar{u} + sd_b^t)$ that satisfies $\phi(0) = L(\bar{u})$, $\phi(s_b^t) = L(u^t)$, and $\phi^{'}(s_b^t) = g^t \cdot d_b^t = \mu_b$ has been used. 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 $\phi^{'}(0) = \Delta_b eta$.

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, Escudero et al. (2016) obtained interesting results applying such 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 (12), which can be helpful in the choice of s_b^t. Sophisticated heuristics exploiting those information have been developed by Frangioni (1997). For the present work, the heuristic implemented tends to increase s_b^t whenever Δ_β 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 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, it is valuable 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 θ is chosen throughout the Volume algorithm. Concerning the bundle β, 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 α_i belongs to the optimal basis of the dual subproblem (12). Thus, items are automatically discarded, if inactive after 20 consecutive iterations. 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 α as its multiplier.

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

$$\theta_t = \begin{cases} \theta_0, & \text{if } \tau > 1 \\ \theta_0/10, & \text{if } \tau < 0 \\ \tau, & \text{otherwise} \end{cases} \qquad \tau = \max\left\{\theta_0, \frac{\theta_0 \|g^0\|^2 - d_v^t g^t}{\|g^0\|^2 - d_v^t g^t}\right\}. \tag{19}$$

The parameter θ_0 is initially set to 1.0 and then decreased. More precisely, θ_0 is multiplied by a factor 0.3, every time that the current solution value has not improved by at least 1%. According

to Barahona and Anbil (2000), 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 θ_t.

Finally, a third stopping criterium as stated by Frangioni et al. (2017) was put in practice for both algorithms. At an iteration t, if the current search direction d^t and its respective linearization error $\hat{e}^t$, with respect to 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 1° or 2° 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 ϵ 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 are 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 4 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 three for each pair of topology and for each percentage p of arcs, only five perturbations are performed. The number of rounds was set to 10, with a computational time limit of 24 hours for the heuristic to run.

Still aiming at the time consumption, the results obtained with the first relaxation (presented in 5.3) 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.

5.3. Computational results

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

5.3.1. Computational performance of the Volume and Bundle methods

Volume and Bundle methods will be compared in terms of the quality of the lower bound they provide, and in terms of time consumption. Because the Lagrangian subproblem resulting from the relaxation of flow conservation constraints, that is, (14)–(18), 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, the linear relaxation for the large instances with 2000 or more commodities could not be solved to optimality, keeping the linear relaxation bound for instances Canad-N and for the groups B–D.

The linear relaxation of instances with low fixed costs, C 02_001, C 14_001, and D 20_001, could be solved in few minutes, whereas it can take 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 3 GB in average for that group, the Bundle needed 14 GB of RAM, which can be an obstacle depending on the machine available to solve problem instances of that size.

Although Crainic et al. (2001) discuss 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 Frangioni et al. (2017). 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 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, the remaining lines correspond to average results for five instances with each C–F ratio in each group, see Table 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 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 five instances with a given C–F ratio in a group. The best average results for each group of instances are highlighted in bold face. The computational time is reported in column "Time" in seconds.

For the experiments conducted, Table 3 shows that, in 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 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, which 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 either the time or the iteration limits were attained.

We plot in Fig. 1 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 five iterations and the respective computation time at that precise moment. The curves for groups E–H (Figs. 1a–d) 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 other words, it was observed that the gain obtained at each step of the Bundle method is quite worthwhile at the beginning of the optimization process, but thanks to the faster iterations,

Table 3
Average computational performance of the Lagrangian methods

	Bundle		Volume	
Group (C_F)	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	10,006.8	**0.00**	9078.0
H 01_001	1.62	9493.7	**0.00**	4152.1

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.

5.3.2. Optimality gaps

From another perspective, it is worth to consider the impact of embedding the Lagrangian methods in the heuristic schemes proposed in Section 4 to produce feasible primal solutions for the FCMC. Table 4 presents results for the benchmark set of instances Canad-N. As, to the best of our knowledge, there are no upper bounds reported in the literature for instances Canad-N, CPLEX was run 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, the followings are reported (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)

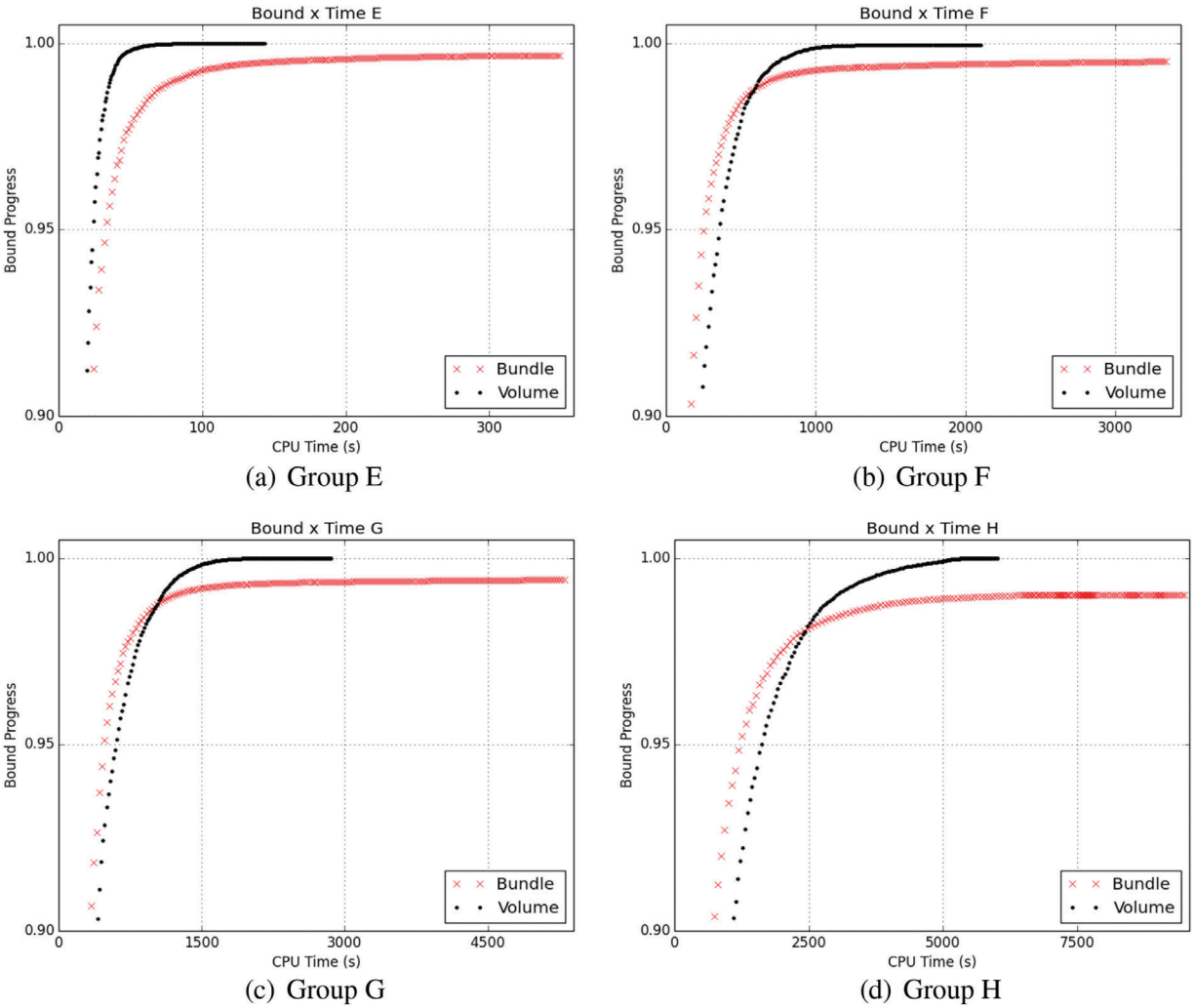

Fig. 1. Average bound progression for the largest instances with respect to computation time

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 is effectively 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

Table 4
Feasible solutions for benchmark instances Canad-N

	Bundle					Volume					CPLEX
Instance	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**	13,098	−66.97	10.94	655	4.34	10,775	−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**	10,840	−82.86	14.32	738	6.06	9655	−82.71	83.74

of perturbations and combinations in Algorithm 2 lead much better solutions, but at expense of higher computational times. Nevertheless, in less than four 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 20% for 18 out of 48 instances, including gaps larger than 50% for eight instances, and up to 84%. Therefore, regardless of the Lagrangian method used, the heuristic has been able to provide good quality solutions, being six 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 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 5–7, using the same column nomenclatures as in Table 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, that is, C/2_001, C/14_001, and D/20_001. In addition, specially with the Volume version, good quality bounds were provided for lightly congested instances, that is, D/1_20, F/1_001, and almost all instances of group E, because optimality gaps between 5% and 13% were obtained. Inversely, the hardest instances were the ones having high levels of F and C ratios.

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 (five per pertubation, being five different percentages of pertubation, and three per combination, being 10 combinations given a pool of five topologies) with 250 iterations allowed, which would have been impracticable if one had to deal with LPs. Moreover, 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, which also reduced computational times. Indeed, the Volume version of the heuristic is noticeably faster, taking the average as reference (lines "Avg"). 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 the group A, and some instances of group G, the heuristic with the Volume algorithm obtained better results in 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–E. For the very large instances of groups F–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. Moreover, the gap between lower and upper bounds includes the potentially high "natural" gap between the linear relaxation value and the optimal one. Therefore, even though there are gaps reaching 40%, the deviation to the optimal

Table 5
Computational results for large scale instances with 100 nodes and 1000 arcs

	Bundle				Volume			
Instance	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**	10,660	38.77	296	25.75	7279
A/14_05b	33.10	585	**23.06**	10,577	38.12	357	25.22	7274
A/14_05c	33.31	637	25.68	11,218	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 6
Computational results for large scale instances with 100 nodes and 1000 and 1200 arcs

	Bundle				Volume			
Instance	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	17,327	5.22	321	**4.03**	8848
E/1_001b	10.83	1022	2.25	13,132	2.19	246	**1.02**	5152
E/1_001c	13.37	1503	2.79	16,202	2.16	269	**1.33**	5848
E/1_001d	10.58	1580	4.48	17,786	3.66	637	**3.25**	11,458
E/1_001e	14.87	800	6.73	20,827	5.00	285	**3.97**	8143
E/20_001a	10.46	936	4.89	16,657	5.32	424	**3.76**	9224
E/20_001b	8.36	1256	2.11	13,114	2.64	317	**1.51**	6378
E/20_001c	13.04	1269	2.85	15,457	2.47	242	**1.61**	6130
E/20_001d	9.91	1615	4.45	18,267	3.62	912	**3.23**	13,154
E/20_001e	13.75	903	6.20	20,110	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	12,667	8.86	364	**5.60**	6511

value is possibly smaller. For example, the heuristic solution using Volume for cN/01 in Table 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 in the search for an improved 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. Now observe the case of instance H/1_001e, for which the Volume version of Algorithm 1 ran for a much long period, leaving practically no time for further improvements in phase two. However, in this case the Volume version of Algorithm 1

Table 7
Computational results for large scale instances with 100 and 200 nodes and up to 12,000 arcs

	Bundle				Volume			
Instance	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	25,285	23.91	3115	**12.74**	18,025
F/1_20b	36.39	5575	28.37	24,076	17.11	3771	**9.28**	18,502
F/1_20c	30.76	4691	23.63	19,545	23.57	3984	**12.78**	20,478
F/1_20d	54.91	3947	42.46	30,138	22.87	2516	**21.10**	15,588
F/1_20e	53.40	4188	41.74	30,421	26.68	2797	**21.17**	16,941
F/20_20a	53.50	7533	46.00	64,773	46.82	7573	**39.07**	45,601
F/20_20b	57.76	5583	47.09	73,930	61.81	5945	**46.01**	53,075
F/20_20c	49.10	8214	39.07	88,295	43.27	7389	**31.29**	65,425
F/20_20d	55.91	6374	47.85	52,991	59.87	6047	**47.69**	45,614
F/20_20e	54.59	6082	46.42	52,949	49.85	5864	**39.74**	42,237
F/1_001a	21.89	5509	11.10	61,139	10.58	2431	**9.87**	27,260
F/1_001b	21.41	6140	11.71	71,669	9.97	2429	**9.91**	27,203
F/1_001c	18.70	6522	9.52	49,269	7.86	2163	**7.31**	22,922
F/1_001d	21.89	4758	10.92	55,667	10.58	2282	**9.87**	25,699
F/1_001e	23.11	9239	13.47	92,924	10.93	16,835	**10.63**	82,305
Avg	39.52	5354	29.56	46,677	28.38	5009	**21.90**	35,125
G/20_001a	29.37	12,259	**14.01**	100,695	15.87	5036	15.19	37,794
G/20_001b	27.99	11,625	**14.62**	101,960	17.49	5979	17.24	45,540
G/20_001c	49.45	8115	31.82	87,603	16.58	7403	**15.85**	57,861
G/20_001d	35.54	12,027	**17.87**	89,948	20.79	5294	20.03	40,918
G/20_001e	43.86	9253	23.51	86,630	21.55	8014	**20.59**	61,858
G/20_20a	61.62	10,484	53.00	73,758	55.75	10,605	**44.54**	53,434
G/20_20b	50.13	14,308	43.20	91,850	49.46	11,756	**40.90**	70,321
G/20_20c	58.04	10,021	47.05	93,149	54.08	6705	**43.86**	46,828
G/20_20d	57.31	12,358	48.36	89,119	61.14	11,047	**43.25**	73,745
G/20_20e	56.79	9380	45.60	71,617	45.37	8538	**36.86**	46,055
G/1_001a	31.36	8826	**15.69**	74,962	25.86	5453	23.88	37,899
G/1_001b	31.17	11,190	**15.46**	77,377	25.27	4477	24.16	33,981
G/1_001c	27.95	10,582	**13.97**	77,408	20.65	4483	18.53	32,896
G/1_001d	27.95	10,855	**13.91**	76,142	20.65	4576	18.53	32,530
G/1_001e	42.34	7417	24.85	90,230	16.75	5439	**16.29**	57,802
Avg	42.06	10,580	28.19	85,497	31.15	6987	**26.65**	48,631
H/1_20a	36.85	10,642	35.07	105,030	17.25	11,086	**13.11**	58,245
H/1_20b	29.47	10,775	10.95	96,943	13.70	11,161	**7.99**	57,296
H/1_20c	39.34	10,761	37.26	71,769	15.30	11,090	**13.40**	59,788
H/1_20d	60.46	10,447	54.78	87,719	22.40	11,455	**22.40**	62,955
H/1_20e	62.99	9511	59.82	88,810	23.63	9362	**23.63**	56,172
H/20_20a	48.73	22,538	45.55	86,904	48.37	20,747	**43.09**	90,358
H/20_20b	46.82	37,007	37.82	90,910	44.54	22,426	**35.52**	90,787
H/20_20c	45.43	12,363	**42.75**	89,430	52.11	14,826	46.22	92,250
H/20_20d	60.05	10,923	54.96	87,368	51.63	14,838	**45.37**	88,430
H/20_20e	67.52	11,109	63.59	88,644	52.41	13,720	**49.81**	92,306
H/1_001a	57.13	19,209	53.71	96,211	38.60	19,091	**36.88**	90,135
H/1_001b	53.72	26,888	50.04	100,843	21.31	14,250	**20.48**	86,912
H/1_001c	60.18	21,270	54.21	94,343	45.42	16,653	**45.19**	88,491
H/1_001d	39.88	33,522	32.31	87,016	31.70	16,574	**29.97**	88,221
H/1_001e	41.50	23,836	37.88	96,031	17.24	76,101	**17.24**	89,783
Avg	50.00	18,053	44.71	91,198	33.04	18,892	**30.02**	79,475

obtained a good quality solution with an optimality gap of 17%. To conclude this section, it can be noticed that, in the improving phase, Algorithm 2 managed to significantly reduce gaps, specially when the time limitation was not an issue.

5.3.3. Comparisons with state-of-the-art heuristics

A third experiment was conducted to compare the performance of the heuristic schemes proposed in Section 4 with state-of-the-art heuristics from the literature for the FCMC. For such purpose, sets C and R of instances introduced by Crainic et al. (2000) were used, because these smaller instances have been served as benchmark for several heuristics proposed in the literature. Groups C and R have 43 and 153 instances, respectively. These instances are also available for download at http://www.di.unipi.it/optimize/Data/MMCF.html.

Our approach is compared to the following heuristics and metaheuristics: (i) CYCLE, the cycle-based tabu search by Ghamlouche et al. (2003); (ii) RELINK, the path relinking by Ghamlouche et al. (2004); (iii) MULTI, the multilevel cooperative search by Crainic et al. (2006); (iv) SCALE, the capacity scaling heuristic by Katayama et al. (2009); and (v) LCBR, the local branching heuristic by Rodríguez-Martín and Salazar-González (2010).

Table 8 presents results for the group 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 seven heuristics used in this comparison. A gap of zero means that the heuristic obtained the best result, and are highlighted in bold face. The first column in Table 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 Katayama et al. (2009), and the ones obtained with LCBR are reported in Rodríguez-Martín and Salazar-González (2010). The seventh and eighth columns present results of the best solutions obtained with the heuristic schemes proposed in Section 4 using Volume and Bundle to solve the Lagrangian problems, respectively. The last line of Table 8 presents the average gap. Table 9 presents results for instances of group R. For each size, there is a number of instances with different capacity and cost ratios. As far as we know, only Ghamlouche et al. (2003) and Katayama et al. (2009) have reported solution values for this group. Table 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 four heuristics used in this comparison.

For group 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 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 in 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. Note that SCALE due to Katayama et al. (2009) comes out of our experiments as the best performing heuristic in terms of average gaps to the best solution.

Table 8
Comparisons for benchmark instances of group C

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 9
Comparisons for benchmark instances of group R

$\|N\|$	$\|A\|$	$\|K\|$	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

6. 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 that 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 a future research, it is suggested to extend the computational comparisons to a set of nonstructured problems.

With respect to the estimation of good upper bounds, one can say that the results obtained with the Volume were better in average. Finally, we have provided 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.

Future work aims at embedding the Lagrangian Relaxation in a more sophisticated schemes like *Branch-and-Price-and-Cut* and *Relax-and-Cut*, inspired in Barahona and Ladányi (2006). The use of valid inequalities already known for the problem (and summarized in Chouman et al., 2009), eventually included in the Lagrangian Relaxation, could lead to better bounds and consequently decrease the gaps for large-scale instances, without deteriorating the time performances. Furthermore, alternative NDO methods, such as variable target value methods and the average direction strategy may be considered for comparison. As there is still no general agreement about which one performs better, an empirical comparison in the context of the fixed charge multicommodity network design problem that englobes a wider range of state-of-art nondifferentiable optimization methods would be worthwhile.

Acknowledgments

We would like to thank Antonio Frangioni for providing his Bundle implementation. The first author would like to thank FAPEMIG and the last author would like to thank CNPq for the financial support.

References

Bahiense, L., Maculan, N., Sagastizábal, C., 2002. The volume algorithm revisited: relation with bundle methods. *Mathematical Programming* 94, 41–60.

Barahona, F., Anbil, R., 2000. The volume algorithm: producing primal solutions with a subgradient method. *Mathematical Programming* 87, 385–399.

Barahona, F., Ladányi, L., 2006. Branch and cut based on the volume algorithm: Steiner trees in graphs and max-cut. *RAIRO-Operations Research* 40, 1, 53–73.

Bonnans, J.F., Gilbert, J.C., Lemaréchal, C., Sagastizábal, C.A., 2006. *Numerical Optimization*. Springer-Verlag, Berlin–Heidelberg.

Briant, O., Lemaréchal, C., Meurdesoif, P., Michel, S., Perrot, N., Vanderbeck, F., 2008. Comparison of bundle and classical column generation. *Mathematical Programming* 113, 2, 299–344.

Chouman, M., Crainic, T.G., Gendron, B., 2003. A cutting-plane algorithm based on cutset inequalities for multicommodity capacitated fixed charge network design. Technical report, Centre de recherche sur les transports, Université de Montréal, Montreal, Canada.

Chouman, M., Crainic, T.G., Gendron, B., 2009. A Cutting-Plane Algorithm for Multicommodity Capacitated Fixed-Charge Network Design. Technical Report, CIRRELT, Montreal, Canada.

Cornuejols, G., Sridharan, R., Thizy, J., 1991. A comparison of heuristics and relaxations for the capacitated plant location problem. *European Journal of Operational Research* 50, 280–297.

Crainic, T.G., Frangioni, A., Gendron, B., 2001. Bundle-based relaxation methods for multicommodity capacitated fixed charge network design. *Discrete Applied Mathematics* 112, 1–3, 73–99.

Crainic, T.G., Gendreau, M., Farvolden, J.M., 2000. A simplex-based tabu search for capacitated network design. *INFORMS Journal on Computing* 12, 223–236.

Crainic, T.G., Li, Y., Toulouse, M., 2006. A first multilevel cooperative algorithm for capacitated multicommodity network design. *Computers & Operations Research* 33, 9, 2602–2622.

Escudero, L.F., Garín, M.A., Pérez, G., Unzueta, A., 2012. Lagrangian decomposition for large-scale two-stage stochastic mixed 0-1 problems. *TOP* 20, 347–374.

Escudero, L.F., Garín, M.A., Unzueta, A., 2016. Cluster Lagrangean decomposition in multistage stochastic optimization. *Computers & Operations Research* 67, 48–62.

Frangioni, A., 1996. Solving semidefinite quadratic problems within nonsmooth optimization algorithms. *Computers & Operations Research* 23, 1099–1118.

Frangioni, A., 1997. Dual ascent methods and multicommodity flow problems. PhD thesis, Università di Pisa, Pisa, Italy.

Frangioni, A., 2002. Generalized bundle methods. *SIAM Journal on Optimization* 13, 117–156.

Frangioni, A., 2005. About Lagrangian methods in integer optimization. *Annals of Operations Research* 139, 1, 163–193.

Frangioni, A., 2013. The NDOSOlver + FiOracle Project.

Frangioni, A., Gallo, G., 1999. A bundle type dual-ascent approach to linear multicommodity min-cost flow problems. *INFORMS Journal on Computing* 11, 4, 370–393.

Frangioni, A., Gendron, B., Gorgone, E., 2017. On the computational efficiency of subgradient methods: a case study with Lagrangian bounds. *Mathematical Programming Computation* 9, 4, 573–604.

Frangioni, A., Gorgone, E., 2014. Bundle methods for sum-functions with "easy" components: applications to multicommodity network design. *Mathematical Programming, Series A* 145, 1–2, 133–161.

Gendron, B., 2011. Decomposition methods for network design. *Procedia–Social and Behavioral Sciences* 20, 31–37.

Gendron, B., Crainic, T., Frangioni, A., 1999. Multicommodity capacitated network design. In Sansò, B., Soriano, P. (eds) *Telecommunications Network Planning*. Kluwer Academic, Norwell, MA, pp. 1–19.

Gendron, B., Larose, M., 2014. Branch-and-price-and-cut for large-scale multicommodity capacitated fixed-charge network design. *EURO Journal on Computational Optimization* 2, 1–2, 55–75.

Geoffrion, A.M., 1974. Lagrangean relaxation for integer programming. In Balinski, M.L. (ed.) *Approaches to Integer Programming*, *Mathematical Programming Studies*, Vol. 2. Springer, Berlin–Heidelberg, pp. 82–114.

Ghamlouche, I., Crainic, T.G., Gendreau, M., 2003. Cycle-based neighbourhoods for fixed-charge capacitated multicommodity network design. *Operations Research* 51, 4, 655–667.

Ghamlouche, I., Crainic, T.G., Gendreau, M., 2004. Path relinking, cycle-based neighbourhoods and capacitated multicommodity network design. *Annals of Operations Research* 131, 109–133.

Gondzio, J., Du Merle, O., Sarkisian, R., Vial, J.P., 1996. ACCPM — A library for convex optimization based on an analytic center cutting plane method. *European Journal of Operational Research* 94, 1, 206–211.

Haouari, M., Layeb, S.B., Sherali, H.D., 2008. The prize collecting steiner tree problem: models and Lagrangian dual optimization approached. *Computational Optimization and Applications* 40, 1, 13–39.

Held, M., Karp, R.M., 1971. The traveling-salesman problem and minimum spanning trees: part II. *Mathematical Programming* 1, 1, 6–25.

Held, M., Wolfe, P., Crowder, H.P., 1974. Validation of subgradient optimization. *Mathematical Programming* 6, 62–88.

Hunter, D.R., Lange, K., 2004. A tutorial on MM algorithms. *The American Statistician* 58, 1, 30–37.

Katayama, N., Chen, M., Kubo, M., 2009. A capacity scaling heuristic for the multicommodity capacitated network design problem. *Journal of Computational and Applied Mathematics* 232, 1, 90–101.

Kazemzadeh, M.R.A., Bektaş, T., Crainic, T.G., Frangioni, A., Gendron, B., Gorgone, E., 2019. Node-based Lagrangian relaxations for multicommodity capacitated fixed-charge network design. Technical Report, CIRRELT-2019-21.

Kelley, J.E., 1960. The cutting-plane method for solving convex programs. *Journal of the Society for Industrial and Applied Mathematics* 8, 703–712.

Khachian, L.G., 1980. Polynomial algorithms in linear programming. *USSR Computational Mathematics and Mathematical Physics* 20, 1, 53–72.

Lemaréchal, C., 1989. Nondifferentiable optimization. In Nemhauser, G., Rinooy Kan, A., Todd, M. (eds) *Optimization, Handbooks in Operations Research and Management Science*, Vol. 1. Elsevier Science, Amsterdam, pp. 529–572.

Lemaréchal, C., 2001. Lagrangean relaxation. In Jünger, M., Naddef, D. (eds) *Computational Combinatorial Optimization*, Vol. 2241. Springer-Verlag, Berlin–Heidelberg, pp. 112–156.

Lemaréchal, C., Nemirovskii, A., Nesterov, Y., 1995. New variants of bundle methods. *Mathematical Programming* 69, 111–147.

Magnanti, T.L., Wong, R.T., 1984. Network design and transportation planning: Models and algorithms. *Transportation Science* 18, 1, 1–55.

Oliveira, W., Sagastizábal, C., 2014. Bundle methods in the XXIst century: a birds's-eye view. *Pesquisa Operacional* 34, 647–670.

Ouorou, A., 2009. A proximal cutting plane method using Chebychev center for nonsmooth convex optimization. *Mathematical Programming* 119, 2, 239–271.

Ouorou, A., Mahey, P., Vial, J.P., 2000. A survey of algorithms for convex multicommodity flow problems. *Management Science* 46, 126–147.

Polyak, B.T., 1969. Minimization of unsmooth functionals. *USSR Computational Mathematics and Mathematical Physics* 9, 3, 14–29.

Rodríguez-Martín, I., Salazar-González, J.J., 2010. A local branching heuristic for the capacitated fixed-charge network design problem. *Computers & Operations Research* 37, 3, 575–581.

Sherali, H.D., Choi, G., 1996. Recovery of primal solutions when using subgradient optimization methods to solve Lagrangian duals of linear programs. *Operations Research Letters* 19, 3, 105–113.

Shor, N.Z., 1985. *Minimization Methods for Non-Differentiable Functions*, Vol. 3. Springer-Verlag, Berlin–Heidelberg.

Wolfe, P., 1975. A method of conjugate subgradients for minimizing nondifferentiable functions. In Balinski, M., Wolfe, P. (eds) *Nondifferentiable Optimization*, Vol. 3. Springer, Berlin–Heidelberg, pp. 145–173.