CIRRELT

Centre interuniversitaire de recherche
sur les réseaux d'entreprise, la logistique et le transport

Interuniversity Research Centre
on Enterprise Networks, Logistics and Transportation

# A Lagrangian-Based Matheuristic for Multilayer Single Flow-Type Multicommodity Capacitated Fixed-Charge Network Design

**Mohammad Rahim Akhavan Kazemzadeh**
**Teodor Gabriel Crainic**
**Bernard Gendron**

**August 2018**

**CIRRELT-2018-37**

Bureaux de Montréal :
Université de Montréal
Pavillon André-Aisenstadt
C.P. 6128, succursale Centre-ville
Montréal (Québec)
Canada H3C 3J7
Téléphone : 514 343-7575
Télécopie : 514 343-7121

Bureaux de Québec :
Université Laval
Pavillon Palasis-Prince
2325, de la Terrasse, bureau 2642
Québec (Québec)
Canada G1V 0A6
Téléphone : 418 656-2073
Télécopie : 418 656-2624

www.cirrelt.ca

# A Lagrangian-Based Matheuristic for Multilayer Single Flow-Type Multicommodity Capacitated Fixed-Charge Network Design

**Mohammad Rahim Akhavan Kazemzadeh[1,2,*], Teodor Gabriel Crainic[1,3], Bernard Gendron[1,2]**

[1] Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT)

[2] Department of Computer Science and Operations Research, Université de Montréal, P.O. Box 6128, Station Centre-Ville, Montréal, Canada H3C 3J7

[3] Department of Management and Technology, Université du Québec à Montréal, P.O. Box 8888, Station Centre-Ville, Montréal, Canada H3C 3P8

**Abstract.** In multilayer network design, there are several networks, each at a given layer. To route a set of commodities from the origins to the destinations, at each layer, appropriate capacitated links have to be opened by paying a fixed cost per link. So-called design connectivity constraints need to be enforced. They impose that a link is opened in a layer only if a chain of supporting links is opened in another layer. The problem is to find a minimum cost design and routing for all layers that satisfies capacity and connectivity constraints as well as demands. In this paper, we address the Multilayer Single flow-type Multicommodity Capacitated Fixed- Charge Network Design problem (MSMCFND), where commodities are routed only in the first layer. The amount of flow on each link in a particular layer is equal to the amount of flow on its supported links in the first layer. We propose a general formulation for the MSMCFND, for which we develop an efficient solution method based on Lagrangian relaxation to address large-scale instances. The resulting Lagrangian-based matheuristic method is empowered by intensification, diversification, and post-optimization procedures using long-term and short-term memories. The results show that the proposed algorithm is competitive with (and often significantly better than) a state-of-the-art MIP solver on a large set of randomly generated instances, not only with respect to the obtained upper bounds, but also in terms of optimality gaps.

**Keywords**. Multilayer network design, Lagrangian relaxation, slope scaling, subgradient method, matheuristics.

**Acknowledgements.** While working on this project, the second author was Adjunct Professor with the Département d'informatique et de recherche opérationnelle (DIRO), Université de Montréal. The first author gratefully acknowledges the support of the Université de Montréal, through its end-of-doctoral-studies scholarship and the excellence scholarship of the DIRO, as well as the support of CIRRELT through its doctoral excellence scholarship. Partial funding for this project has also been provided by the Natural Sciences and Engineering Research Council of Canada (NSERC), through its Discovery Grant program. We also gratefully acknowledge the support of Fonds de recherche du Québec through their infrastructure grants.



---

* Corresponding author: Mohammad.Akhavan@cirrelt.ca

Dépôt légal – Bibliothèque et Archives nationales du Québec
Bibliothèque et Archives Canada, 2018

## 1. Introduction

Applications of network design models are extensive in the fields of transportation, telecommunications, logistics and production planning (Magnanti and Wong, 1984; Minoux, 1989). In network design, given a potential network with capacitated links, several commodities such as goods, data or people, have to be routed between the different origin and destination points. A network has to be constructed by opening appropriate links to route the commodities. A *fixed design cost* has to be paid to open a link. In addition to the fixed design costs, a per unit routing cost, or *flow variable cost*, is associated with each link and has to be paid to route each unit of commodity demand on the link. The problem is to select a set of links and route the demands from the origins to the destinations on the constructed network such that the costs are minimized and capacities are respected.

In this paper, we consider multilayer network design models, which have applications in the fields of transportation (Cordeau et al., 2001; Zhu et al., 2014; Crainic et al., 2014) and telecommunications (Dahl et al., 1999; Knippel and Lardeux, 2007). Multilayer network design models differ from a typical single-layer network design model in two main aspects. First, instead of a single network, there are several networks, each at a given layer. Each network has its nodes, potential links with capacities, and possibly commodities. To route the demands of the commodities from the origins to the destinations, appropriate links have to be opened by paying a fixed cost per link. Second, in multilayer network design, there are *design connectivity* and *flow connectivity* requirements between the layers. Design connectivity means that opening a link in a layer depends on opening a chain of supporting links (a path) in another layer, named as supporting layer. Flow connectivity states that the amount of flow on each link in a particular layer depends on the flow on its supported links in another layer. It is possible that some layers have no commodities to route, but these layers also support the routing of other layers through constraints that connect the layers. In particular, when there is only one layer that has commodities to route, the problem is a *multilayer single flow-type network design*.

In this paper, we address the Multilayer Single flow-type Multicommodity Capacitated Fixed-Charge Network Design problem (MSMCFND), where commodities are routed only in layer $l = 1$, given a set of layers $L = \{1, 2, .., |L|\}$. The amount of flow on each link in a particular layer $l \geq 2$ is equal to the amount of flow on its supported links in layer $l - 1$ and consequently to the amount of flow on its supported links in layer $l = 1$. A comprehensive review and taxonomy on multilayer network design models and solution methods for such problems can be found in Akhavan Kazemzadeh et al. (2018). The MSMCFND represents a set of service network design problems with multiple consolidation levels, which arise in transportation applications, see Zhu et al. (2014) and Crainic et al. (2014) for example.

Lagrangian relaxation is among the most efficient methods to address the single-layer multicommodity capacitated fixed-charge network design problem (MCFND), see Holmberg and Yuan (2000), Sellmann et al. (2002), Crainic et al. (2004), and Kliewer and Timajev (2005) for example. Motivated by these developments, we propose a Lagrangian-based matheuristic for the MSMCFND. Although there are efficient Lagrangian-based heuristics for the MCFND, these methods cannot be adapted in a straightforward way to the MSMCFND. Indeed, the challenge in multilayer network design is how to handle the significant additional complexity incurred by the design connectivity constraints between the layers. On the one hand, these additional constraints complicate the task of developing Lagrangian relaxations that keep a balance between the quality of the lower bound and the computational efficiency of solving the Lagrangian subproblem. On the other hand, the derivation of effective feasible solutions becomes significantly more difficult in the presence of these constraints.

We propose a Lagrangian-based matheuristic solution method based on a *slope scaling* scheme. The idea of slope scaling is to iteratively solve a linear multicommodity flow formulation, and to use the flow distribution to adjust the linear approximation at the next iteration. When the slope scaling method stalls, a perturbation move changes the initial linear approximation to start a new slope scaling procedure; for more details on slope scaling, see Crainic et al. (2004) and Kim and Pardalos (1999). Note that Zhu et al. (2014) also used a slope scaling procedure but did not integrate it with a Lagrangian relaxation scheme, as proposed in this paper. In particular, our Lagrangian relaxation approach provides not only the lower bounds to assess the quality of the obtained feasible solutions, but also guides the slope scaling search by providing initial solutions and by defining the perturbation moves.

Our main contribution, apart from presenting a general formulation for the MSMCFND, is to propose an efficient solution method based on Lagrangian relaxation to solve large-scale MSMCFND instances. In the proposed algorithm, a subgradient method is used to find lower bounds. A primal heuristic procedure based on slope scaling is developed to find upper bounds using the design information derived from the Lagrangian subproblem. The best upper bound is then used to guide the subgradient method. We are comparing our results to those obtained by a state-of-the-art mixed-integer programming (MIP) solver. We observe that the algorithm produces better results not only in terms of the upper bounds (for 90% of the instances) but also in terms of the optimality gaps (for 60% of the instances).

The paper is organized as follows. We state the problem and the mathematical formulation in Section 2. In Section 3, we describe how to compute the lower and upper bounds through Lagrangian relaxation and slope scaling. Section 4 summarizes the algorithm that combines the Lagrangian relaxation and the slope scaling procedure. In Section 5, we present the numerical

results on randomly generated instances. In Section 6, we summarize this work and propose future research directions.

## 2. Problem Statement and Formulation

In this section, we first formally define the problem. We then present a formulation for the MSMCFND. Finally, we describe a preprocessing procedure that reduces the flow capacities.

### 2.1. Problem Definition

In the MSMCFND, multiple networks are considered, one per layer. Given a set of layers $L = \{1, 2, ..., |L|\}$, each layer $l \in L$ consists of nodes and potential links. There is a fixed design cost to open a link in each layer, while the flow costs are defined only for the links of layer $l = 1$, called the *base layer*, for which there are commodities with specific demands that have to be routed between their origin and destination nodes. In the MSMCFND, there are two types of connectivity requirements between the layers: design connectivity and flow connectivity. The first one, design connectivity, means that a link opened in the base layer requires a chain of *supporting links* (a path) to be opened in layer 2. Similarly, a link opened in a given layer $l$ requires a path to be opened in its upper layer, i.e., layer $l+1$. If link $a$ of layer $l$ requires a path (that includes links $b$ and $c$) to be opened in layer $l+1$, then $l$ is said to be *supported* by $l+1$, and $l+1$ is said to be *supporting* layer of $l$. In addition, $a$ is *supported* by $b$ and $c$, while $b$ and $c$ are *supporting* links of $a$. Note that, a link in a particular layer might be in more than one path; therefore, it might support more than one link of its lower layer, which means that the fixed design cost of the link has to be paid as soon as one of the potential supported links of the lower layer is designed. Design connectivity requirements include *design capacity* constraints, which, for link $b$ in layer $l$, limits the number of designed links in layer $l-1$ supported by link $b$. The second type of connectivity constraints, flow connectivity, imposes that the flow on link $b$ in a given layer $l \geq 2$ is equal to the summation of the flows of all the links in the base layer supported by link $b$. In summary, in the MSMCFND, the goal is to determine a minimum cost design and routing in all layers to satisfy demands, while considering capacity constraints, as well as flow and design connectivity requirements.

One of the applications of the MSMCFND is the integrated rail freight service network design problem described in Zhu et al. (2014). They proposed a three-layer network to model the problem where a double consolidation policy is performed. First, cars are grouped into so-called blocks (blocking process). Then, they are grouped into services to make up trains (train make-up process). The next step is to select the services and define their schedules. To propose a formulation determining blocking, train make-up, and service selection decisions simultaneously, Zhu et al. (2014) considered a three-layer network including: 1) a car layer that consists of links on which cars are

moved in each terminal; 2) a block layer that includes block links from the origins to the destinations of the blocks to support car movements; and 3) a service layer that includes service links to support block movements. In the car layer, the flows of commodities are moved via the car links and the projected block links from the block layer. Appropriate block links have to be designed or opened to route the cars. To design a block link, a chain of supporting services has to be opened in the service layer. The flow of each service is equal to the summation of the flows on all its supported blocks. The problem is to find a minimum cost blocking and service design, and flow routing of the cars to satisfy demands, while considering flow capacity of the blocks and the services, blocking capacity of each terminal, and design connectivity of the block and the service layers. Zhu et al. (2014) proposed a mixed-integer programming (MIP) formulation for the problem. We generalize the problem setting and formulation to an arbitrary number of layers. Furthermore, since we focus on the interaction between the layers, we do not consider application-specific requirements such as terminal block-building workload constraints.

### 2.2. Formulation

We are given a set of layers $L$ and a network $G_l = (N, A_l)$ for each layer $l \in L$, where $N$ and $A_l$ are the sets of nodes (the same for all layers) and directed arcs for layer $l \in L$, respectively. The sets $A_l^+(n)$ and $A_l^-(n)$ represent, respectively, the outward and inward arcs of node $n \in N$ in layer $l \in L$. A set $K$ of commodities have to be routed through the base layer network. The amount of each commodity $k \in K$ that must be routed from its origin $O(k) \in N$ to its destination $D(k) \in N$ is $d^k > 0$. We define $u_{al} > 0$ and $v_{al}$ (a positive integer) as the flow capacity and the design capacity of arc $a \in A_l$ in layer $l \in L$, $c_{a1}^k \geq 0$ as the variable flow cost of commodity $k \in K$ on arc $a \in A_1$, and $f_{al} \geq 0$ as the fixed design cost of arc $a \in A_l$ of layer $l \in L$. We also use the following additional notation: for each $n \in N$ and $k \in K$, $w_n^k = d^k$, if $n = O(k)$, $w_n^k = -d^k$, if $n = D(k)$, and $w_n^k = 0$, if $n \neq O(k), D(k)$; for each $l \in L$, $a \in A_l$ and $k \in K$, $h_{al}^k = \min\{d^k, u_{al}\}$.

To model the connectivity constraints, we introduce the following notation: $B^{al}$ is the set of arcs in layer $l-1$ that are supported by arc $a \in A_l$ for $l \geq 2$ and $B_1^{al}$ is the set of arcs in layer 1 that are supported by arc $a \in A_l$, $l \in L$ (we assume that $B_1^{a1} = \{a\}$ for arc $a \in A_1$). Two sets of decision variables are introduced to formulate the problem: binary design variables and continuous flow variables. The design variable $y_{al}$ is 1 if arc $a \in A_l$ is selected in layer $l \in L$, and 0 otherwise. The flow variable $x_{a1}^k$ is the amount of flow of commodity $k \in K$ on arc $a \in A_1$. The MSMCFND model takes the following form:

$$\min \quad \sum_{a \in A_1} \sum_{k \in K} c_{a1}^k x_{a1}^k + \sum_{l \in L} \sum_{a \in A_l} f_{al} y_{al} \tag{1}$$

$$\sum_{a\in A_1^+(n)} x_{a1}^k - \sum_{a\in A_1^-(n)} x_{a1}^k = w_n^k \qquad \forall n \in N, \quad \forall k \in K \qquad (2)$$

$$\sum_{k\in K}\sum_{b\in B_1^{al}} x_{b1}^k \leq u_{al} y_{al} \qquad \forall l \in L, \quad \forall a \in A_l \qquad (3)$$

$$\sum_{b\in B_1^{al}} x_{b1}^k \leq h_{al}^k y_{al} \qquad \forall l \in L, \quad \forall a \in A_l, \quad \forall k \in K \qquad (4)$$

$$\sum_{b\in B^{al}} y_{b(l-1)} \leq v_{al} y_{al} \qquad \forall l \in L, \quad l \geq 2, \quad \forall a \in A_l \qquad (5)$$

$$y_{b(l-1)} \leq y_{al} \qquad \forall l \in L, \quad l \geq 2, \quad \forall a \in A_l, \quad \forall b \in B^{al} \qquad (6)$$

$$x_{al}^k \geq 0 \qquad \forall a \in A, \quad \forall k \in K, \quad \forall l \in L \qquad (7)$$

$$y_{al} \in \{0,1\} \qquad \forall a \in A, \quad \forall l \in L \qquad (8)$$

The objective of the model, (1), is to minimize the total routing and design costs. Constraints (2) are the usual *flow conservation equations*, which ensure that the demands are routed from the origins to the destinations in the base layer. The *flow capacity constraints* (3) ensure that the total flow on each arc $a \in A_l$ in layer $l \in L$, which is equal to the sum of the flows on the arcs supported by $a$ in the base layer, does not exceed its flow capacity $u_{al}$. These constraints also impose that: 1) no commodity can be routed on a link unless it is opened; and 2) if there is any flow on an arc in a particular layer, then this flow contributes to the flow of its supporting arcs in the upper layer, and consequently the supporting arcs will be forced to be opened. The *strong linking constraints* (4) impose that no flow of any commodity can circulate on a closed arc. Although these inequalities are redundant for the MIP formulation, our tests show that they improve the linear programming (LP) and the Lagrangian relaxation lower bounds significantly. Similar inequalities also have been used in the literature on network design problems to improve the lower bounds; see Chouman et al. (2016) for example. The *design capacity constraints* (5) ensure that, for each arc $a \in A_l$, the number of designed arcs that are supported by $a$ in layer $l-1$ does not exceed its design capacity $v_{al}$. *Upward design linking constraints* (6) ensure that an arc can be opened only if all its supporting arcs in the upper layer are opened. It is easy to see that these constraints are already satisfied by constraints (3) or by constraints (5). Although constraints (6) are redundant for the MIP formulation, our tests show that they improve the LP and the Lagrangian relaxation lower bounds significantly. Constraints (7) and (8) define the domains of the variables.

### 2.3. Preprocessing Procedure

Using the multilayer structure of the MSMCFND, we can reduce the flow capacities in two ways. First, we know that each arc $a$ in layer $l \leq |L| - 1$ corresponds to a path in the upper layer.

According to constraints (3), the total flow on $a$ (hence, the capacity $u_{al}$) has to be less than or equal to the flow capacity of each corresponding arc in the upper layer:

$$u_{al} \leq \min_{b \in F_a^{l+1}} \{u_{b(l+1)}\} \qquad \forall l \in L, \quad l \leq |L|-1, \quad \forall a \in A_l, \tag{9}$$

where $F_a^{l+1} = \{b \in A_{l+1} | a \in B^{b(l+1)}\}$. Second, we know that any given arc $a$ in layer $l \geq 2$ supports several arcs in the lower layer, and that at most $v_{al}$ arcs can be supported by arc $a$. Therefore, the total flow on $a$ (hence, its capacity $u_{al}$) has to be less than or equal to the sum of the flow capacities of the first $v_{al}$ arcs in layer $l-1$ supported by $a$, where the arcs are sorted in non-increasing order of to their flow capacities:

$$u_{al} \leq \sum_{b \in \widetilde{B}^{al}} u_{b(l-1)} \qquad \forall l \in L, \quad l \geq 2, \quad \forall a \in A_l, \tag{10}$$

where $\widetilde{B}^{al}$ is the set containing the first $v_{al}$ arcs supported by arc $a \in A_l$ in the lower layer.

Based on the above arguments, we derive a preprocessing procedure that: 1) scans each layer in descending order (from $|L|-1$ to 1) and reduces the flow capacities based on (9); 2) scans each layer in ascending order (from 2 to $|L|$) and reduces the flow capacities based on (10); and 3) if at least one flow capacity is reduced in 2), goes back to 1).

## 3. Bounding Procedures

In this section we describe how to compute the lower and upper bounds through Lagrangian relaxation and slope scaling, respectively. The algorithm combining these bounding procedures is presented in the next section.

### 3.1. Computing Lower Bounds through Lagrangian Relaxation

Relaxing constraints (3) and (4) in a Lagrangian way, we obtain the following Lagrangian subproblem:

$$\begin{aligned}(\mathcal{P}_{\mathcal{LR}}(\alpha,\beta)) \quad Z_{LR}(\alpha,\beta) = \min \quad & \sum_{a \in A_1} \sum_{k \in K} c_{a1}^k x_{a1}^k + \sum_{l \in L} \sum_{a \in A_l} f_{al} y_{al} \\ & + \sum_{l \in L} \sum_{a \in A_l} \alpha_{al} \left( \sum_{k \in K} \sum_{b \in B_1^{al}} x_{b1}^k - u_{al} y_{al} \right) + \sum_{l \in L} \sum_{a \in A_l} \sum_{k \in K} \beta_{al}^k \left( \sum_{b \in B_1^{al}} x_{b1}^k - d_{al}^k y_{al} \right) \\ & s.t.\,(2),(5),(6),(7),(8),\end{aligned} \tag{11}$$

where $\alpha_{al} \geq 0$ and $\beta_{al}^k \geq 0$ are the Lagrange multipliers corresponding to constraints (3) and (4), respectively. The Lagrangian subproblem decomposes into $|K|+1$ subproblems: $|K|$ shortest path subproblems, $\mathcal{P}_{SP}^k(\alpha,\beta)$, one for each commodity $k \in K$, in the base layer, and a subproblem that

depends only on the design variables, called the *design subproblem* and denoted $\mathcal{P}_D(\alpha, \beta)$. The cost of each arc $a \in A_1$ in each shortest path subproblem $\mathcal{P}^k_{SP}(\alpha, \beta)$, $k \in K$, is given by:

$$c^k_{a1} + \sum_{l \in L} \sum_{b \in F^l_{a1}} \left(\alpha_{bl} + \beta^k_{bl}\right) \geq 0, \tag{12}$$

where $F^l_{a1} = \left\{b \in A_l | a \in B^{bl}_1\right\}$. The shortest path subproblems are solved using Dijkstra algorithm.

The design subproblem is as follows:

$$\begin{aligned} (\mathcal{P}_D(\alpha, \beta)) \quad Z_{\mathcal{P}_D(\alpha,\beta)} = \min \quad & \sum_{l \in L} \sum_{a \in A_l} \left( f_{al} - \alpha_{al} u_{al} - \sum_{k \in K} \beta^k_{al} d^k_{al} \right) y_{al} \\ s.t. \; & (5), (6), (8) \end{aligned} \tag{13}$$

Before commenting on how we solve the design subproblem, we first ask ourselves whether or not it has the integrality property (Geoffrion, 1974), i.e., if its LP relaxation always has an integral optimal solution.

**Proposition 1.** *The design subproblem does not have the integrality property.*

*Proof.* We define an instance with two layers where all design capacities are equal to 1. Then, the upward design linking constraints (6) are redundant for the LP relaxation of the design subproblem. Further, we assume that all design costs on layer $l = 2$ are equal to 0. As a result, all design variables for layer $l = 2$ assume value 1 in an optimal solution of the LP relaxation of the design subproblem. The resulting LP relaxation of the design subproblem reduces to the LP relaxation of the well-known set packing problem, which has fractional optimal solutions (Padberg, 1973). ■

**Corollary 1.**

$$Z_{LD} = \max_{\alpha, \beta \geq 0} Z_{LR}(\alpha, \beta) \geq Z_{LP},$$

*where $Z_{LP}$ is the optimal value of the LP relaxation. There are instances where the inequality is strict.*

This result illustrates a significant difference between the single-layer MCFND and the MSMCFND: when we relax the capacity and the strong linking contraints in the MCFND, the Lagrangian subproblem has the integrality property and the Lagrangian dual bound $Z_{LD}$ is equal to the LP relaxation lower bound $Z_{LP}$ (Crainic et al., 2001).

In Section 4, where we describe the subgradient method for solving the Lagrangian dual, we see that the design subproblem is rarely solved to optimality: either we solve its LP relaxation or we use a state-of-the-art MIP solver as a heuristic. In both cases, the design subproblem produces a design solution, which can be fractional when solving the LP relaxation, but is necessarily integral when using a MIP solver in a heuristic fashion. Note that, this approach allows us to keep the Lagrangian subproblem tractable, but might result in a lower bound that is inferior to $Z_{LP}$, even for instances where $Z_{LD} > Z_{LP}$.

### 3.2. Computing Upper Bounds through Slope Scaling

To obtain upper bounds, we develop a primal heuristic based on a slope scaling procedure, which is repetitively called during the subgradient method described in Section 4. The following *multilayer multicommodity capacitated network flow problem* (MMCNF) is solved at each iteration of the slope scaling procedure:

$$\min \quad \sum_{a\in A_1}\sum_{k\in K} \overline{c}^k_{a1} x^k_{a1} \tag{14}$$

$$\text{s.t.} \quad (2), (7) \quad \text{and}$$

$$\sum_{k\in K}\sum_{b\in B^{al}_1} x^k_{b1} \le u_{al} \qquad \forall l \in L, \quad \forall a \in A_l, \tag{15}$$

where the linearized costs $\bar{c}$ are initialized, as follows, based on an input design solution $\widehat{y}$ (fractional or integral) of the design subproblem obtained when performing the subgradient method:

$$\overline{c}^k_{a1} = (c^k_{a1} + \rho^0_{a1})(1 + M(1 - \widehat{y}_{a1})) \qquad \forall a \in A_1, \forall k \in K, \tag{16}$$

where $\rho^0_{a1} = \sum_{l\in L}\sum_{b\in F^l_{a1}} f_{bl}/u_{bl}$, and $M$ is a large positive value to avoid routing on the closed arcs. When $\widehat{y}_{a1} = 1$, $a \in A_1$, the costs of the corresponding arcs are equal to the costs in the LP relaxation of the model obtained from the MSMCFND by removing constraints (4)-(6). If $\widehat{y}_{a1} = 0$, $a \in A_1$, then the costs are set to large positive values to avoid any flow on the corresponding arc. Note that, even if $\widehat{y}_{a1}$ is fractional, formula (16) can still be used: arcs with small fractional values are then assigned large costs.

After solving the MMCNF, a flow solution $\overline{x}$ is obtained and the linearized costs are updated using the following equation to trigger the next iteration:

$$\overline{c}^k_{a1} = \begin{cases} c^k_{a1} + \rho_{a1} & \text{if } \overline{x}^k_{a1} > 0 \\ \overline{c}^k_{a1} & \text{if } \overline{x}^k_{a1} = 0 \end{cases} \qquad \forall a \in A_1, \forall k \in K, \tag{17}$$

where $\rho_{a1} = (\sum_{l\in L}\sum_{b\in F^l_{a1}} f_{bl})/\sum_{k\in K}\overline{x}^k_{a1}$. When $\overline{x}^k_{a1}$ is positive, this equation ensures that, at the next iteration, if the flow solution remains the same, the cost on each arc is equal to the fixed design cost plus the sum of the flow variable costs. If $\overline{x}^k_{a1} = 0$, we keep the same cost because it is large enough to avoid any flow. The termination condition of the slope scaling procedure is either to obtain the same objective value for two successive iterations, as suggested in Kim and Pardalos (1999), or to reach a predefined maximum number of slope scaling iterations $it^s_{max}$.

At each slope scaling iteration, a feasible solution might be produced for the MSMCFND. Given the flow solution $\overline{x}$, a design solution $\overline{y}$ is obtained using the following rule:

$$\overline{y}_{al} = \begin{cases} 1 \text{ if } \sum_{k\in K}\sum_{b\in B^{al}_1} \overline{x}^k_{b1} > 0 \\ 0 \quad \text{otherwise} \end{cases} \tag{18}$$

When the obtained design solution is feasible, i.e., it satisfies the design capacity constraints (5), then an upper bound is found for the original problem.

At the end of the slope scaling procedure, an *intensification* step is performed using up to $n$ of the best feasible solutions obtained during the procedure, where $n$ is a parameter typically set to a small value (less than 10). For each of these feasible solutions, a restricted MMCNF is created by: 1) fixing the design variables to their values in the solution, and 2) setting the costs to the original flow variable costs. The resulting MMCNF is solved in order to obtain an optimal flow distribution when the design variables are fixed.

If the initial MMCNF yields a feasible solution to the MSMCFND, the slope scaling procedure proceeds as described above. If, however, this is not the case, we solve the following *auxiliary design subproblem* to obtain a feasible solution:

$$\min \quad \sum_{l \in L} \sum_{a \in \widehat{A}_l} f_{al} y_{al} \tag{19}$$

$$\text{s.t.} \quad (2), (3), (5), (7), (8),$$

where $\widehat{A}_l = \{a \in A | \widehat{y}_{al} = 0\}$. The solution to this auxiliary design subproblem is used as input design solution $\widehat{y}$, and the linearized costs are re-initialized using formula (16). The objective of the auxiliary design subproblem is to select a minimum cost design among the closed arcs of the initial design solution $\widehat{y}$ in such a way that all the constraints that define a feasible solution to the MSMCFND are satisfied. Note that the redundant constraints (4) and (6) are removed in the formulation of the auxiliary design subproblem. Since the resulting model has no commodity-dependent costs and capacities, we can reduce the number of commodities by aggregating the commodities with the same origin into a single commodity. This significantly reduces the computational complexity of solving the auxiliary design subproblem. Besides, we stop solving the auxiliary design subproblem after finding the very first feasible solution. Note that, solving the auxiliary design subproblem only guarantees the feasibility at the first iteration of the slope scaling procedure, but not at every iteration.

We provide a pseudo code for the primal heuristic in Algorithm 1. When a new upper bound is found, the best upper bound found so far (since the beginning of the subgradient method), $UB^{best}$, is updated if it is improved by the current solution. We also define the set of $m$ best feasible solutions found so far, $F$, which is updated whenever a new solution is found, where $m$ is a large value (we use 1000 in our experiments). The set $F$ is a long-term memory that will be used for further improvements described in details in the next section.

**Algorithm 1:** Primal heuristic

```
Input design solution ŷ
Initialize the linearized costs of MMCNF problem using ŷ and formula (16)
Solve MMCNF problem → x̄, ȳ
if ȳ is feasible then
    Compute upper bound
    Update UB^best and feasible solution set F
else
    Solve auxiliary design subproblem → ŷ
    Re-initialize the linearized costs of MMCNF problem using ŷ and formula (16)
    Solve MMCNF problem → x̄, ȳ
end if
repeat
    Update objective function of MMCNF problem using x̄ and formula (17)
    Solve MMCNF problem → x̄, ȳ
    if ȳ is feasible then
        Compute upper bound
        Update UB^best and feasible solution set F
    end if
until the number of iterations is less than it^s_max and slope scaling does not stall
Perform intensification
```

## 4. The Lagrangian-Based Matheuristic

The proposed algorithm consists of three main phases: 1) **initialization**, where the LP relaxation of the original problem, $\mathcal{P}_{LP}$, is solved to initialize the Lagrange multipliers; 2) **Lagrangian** phase, including a subgradient procedure to find lower bounds and the primal heuristic empowered by *intensification* and *diversification* procedures to obtain upper bounds; 3) **post-optimization**, where a restricted MIP problem, $\mathcal{P}_{PO}$, is solved to improve the upper bounds. The algorithm is equipped by *long* and *short term memories* storing specific information obtained during the algorithm execution to run the post-optimization phase as well as the intensification and diversification procedures. In the following subsections, 4.1 to 4.3, we describe the three phases of the algorithm. Subsection 4.4 summarizes the algorithm.

### 4.1. Initialization Phase

The first step in the initialization phase is to solve $\mathcal{P}_{LP}$, the LP relaxation of the MSMCFND formulation (1)-(8), by a cutting plane method performed for a limited time $t^{LP}$, where the violated strong linking constraints (4) are added iteratively to the model. Solving $\mathcal{P}_{LP}$ helps to obtain good initial Lagrange multipliers and lower bound. The Lagrange multipliers are initialized to the values of the dual variables associated to constraints (3) and (4). Note that, some of the strong linking constraints do not appear in the final LP relaxation of the cutting plane procedure. For these constraints, we initialize the corresponding Lagrange multipliers to zero. Preliminary tests show that adding the upward design linking constraints (6) increases significantly the complexity

of the LP relaxations. Therefore, taking into account the limited time to perform the cutting plane method, these constraints are not added to the LP relaxation.

In the next step of the initialization phase, a *graph traversal algorithm*, presented in Chouman et al. (2017), is performed to reduce the size of the shortest path subproblems. Starting from each node, the algorithm traverses the network twice, a forward and a backward traversal, to find the arcs that are reachable for each commodity, i.e., the arcs that belong to at least one path between the origin and the destination of each commodity. We then restrict the network of each shortest path subproblem to these reachable arcs, thus significantly improving the computational complexity of the shortest path computations.

At the end of the initialization phase, a slope scaling procedure is called using the final design solution of $\mathcal{P}_{LP}$ as the input design solution $\widehat{y}$. The goal is to obtain an initial upper bound to be used in the subgradient method described in the next subsection.

### 4.2. Lagrangian Phase

The Lagrangian dual takes the form of a non-differentiable optimization problem:

$$Z_{LD} = \max_{\mu \geq 0} Z_{LR}(\mu), \tag{20}$$

where $\mu = (\alpha, \beta)$. We solve the Lagrangian dual with a subgradient method. At each iteration $i$ of the subgradient method, the current point, $\mu^i$, is moved to a new point, $\mu^{i+1}$, using the formula $\mu^{i+1} = \mu^i + t^i d^i$, where $d^i$ and $t^i$ are the direction and the step size, respectively. The following formula is used to compute the direction:

$$d^i = g^i + \theta^i d^{i-1}, \tag{21}$$

where $g^i$ is a subgradient and $\theta^i \in (0, 1)$ is a parameter adjusted by the following *modified Camerini-Fratta-Maffioli rule* (Camerini et al., 1975):

$$\theta^i = \begin{cases} ||g^i||/||d^{i-1}|| & \text{if } g^i d^{i-1} < 0 \\ 0 & \text{otherwise} \end{cases} \tag{22}$$

This rule is suggested in Crainic et al. (2001) for a subgradient method when the capacity and strong linking constraints are relaxed in the MCFND problem. It is also recommended in Crainic et al. (2001) to project the direction $d^i$ in equation (21), meaning to set to 0 the negative components of $d^i$. In our preliminary tests, the suggested modified Camerini-Fratta-Maffioli rule worked well, while the results were not promising when we projected the direction in equation (21). Therefore, in our final implementation, we use the modified Camerini-Fratta-Maffioli rule without projecting the direction.

The step size is computed using the following formula:

$$t^i = \lambda^i \left( Z^e_{DL} - Z_{LR}\left(\mu^i\right)\right) / g^i d^i, \tag{23}$$

where $Z^e_{DL}$ is an estimation of the value of $Z_{DL}$, and $\lambda^i$ is a scaling factor.

The performance of a subgradient method depends on the values of its parameters, in particular, $\lambda^i$ and $Z^e_{DL}$ (Crainic et al., 2001). The parameter $\lambda^i$ is initialized to $\lambda^0$ and is divided by a parameter $\gamma_1$ (typically, 2), whenever $LB^{best}$ (the best known lower bound) is not improved for $\gamma_2$ consecutive iterations. The minimum value of $\lambda^i$ is restricted to $\lambda_{min}$. In our algorithm, $Z^e_{DL}$ is initially set to the upper bound provided by the slope scaling step of the initialization phase and is updated to the best upper bound found so far by the primal heuristic, $UB^{best}$. The subgradient procedure is limited to a maximum number of iterations $\gamma_3$. For more details on the subgradient method see Crainic et al. (2001).

The subgradient method follows a *three-phase approach* when handling the design subproblem. During the first phase, performed for a limited number of subgradient iterations, $it^{ph1}_{max}$, the LP relaxation of $\mathcal{P}_D(\alpha, \beta)$ is solved by a cutting-plane method, where the upward design linking constraints (6) are added iteratively. During this phase, at the beginning of any given iteration of the subgradient method, the design subproblem has the current set of added cuts, and only the costs are modified. Therefore, we run the primal simplex, because the initial primal basis is feasible, only the optimality must be achieved. When we enter the cutting-plane procedure, we switch to the dual simplex, because the basis is optimal or dual feasible, only the primal feasibility must be reached. During the second phase, performed for a limited number of subgradient iterations, $it^{ph2}_{max}$, the root node of the branch-and-bound tree of the IP design subproblem is solved. At the beginning of phase 2, the final LP formulation of the first phase (including all the cuts added so far) is converted to an IP model. Finally, during the third phase, performed for a limited number of subgradient iterations, $it^{ph3}_{max}$, the IP design subproblem is solved. The solution time of each IP problem is restricted to a time limit, $t^{Des}$. Since we care about the quality of the lower bound, we force the branch-and-bound tree to emphasize the search on finding the best lower bound. After solving the Lagrangian subproblems, a lower bound is computed for the original problem, and the best lower bound found so far, $LB^{best}$, is updated, if a better lower bound is found.

At each iteration of the subgradient method, when a new design solution is obtained from the design subproblem, the primal heuristic step is triggered using as input the solution $\widehat{y}$ of the design subproblem if: 1) the frequency of the total number of the subgradient iterations is equal to a predefined number, $fr^{prim}$, or 2) the lower bound has improved significantly since the last time the primal heuristic was called. The improvement is considered to be significant if $(LB_c - LB_l)/LB_l >$

$\delta^d$, where $\delta^d$ is a parameter (set to 1%) and $LB_c$ and $LB_l$ are, respectively, the lower bound computed at the current iteration and the lower bound obtained the last time the primal heuristic was triggered.

At the end of each execution of the primal heuristic procedure, in addition to the intensification step described in Section 3.2, a diversification step is performed if the algorithm is not able to improve the upper bound for a predefined maximum number of primal heuristic steps, $st^p_{max}$, where a *primal heuristic step* is a complete slope scaling procedure with intensification. The idea of diversification is to avoid selecting the arcs that frequently appeared in the obtained feasible solutions. The design solution of the best upper bound found so far is considered to initialize the linearized cost of formula (16) where a huge linearized cost is assigned not only to the closed arcs but also to a small set of arcs (set to 10) selected randomly from the most frequently opened ones in a percentage (set to 90%) of the obtained feasible solutions so far. Then, a new slope scaling procedure is triggered using the new initial linearized costs. The selected arcs are marked to avoid choosing them again in the next diversification step. Therefore, at each diversification step, a new set of the most frequently opened arcs are penalized. We restrict the total number of slope scaling steps in each diversification procedure and the total number of diversification steps to $it^d_{max}$ and $st^d_{max}$, respectively.

Three different termination conditions are considered to stop the Lagrangian phase: 1) if the procedure elapsed time reaches a predefined time limit $t^{Lag}$, 2) if the subgradient norm is less than $\varepsilon$, and 3) if the total number of the subgradient iterations reaches a predefined number $it^{Lag}$. At the end of the subgradient procedure, a slope scaling step is called using the design solution of the best obtained lower bound found so far in the hope of finding a better upper bound.

### 4.3. Post-Optimization Phase

At the end of the algorithm, a *post-optimization* phase is performed by solving a restricted MIP problem, $\mathcal{P}_{PO}$, for a limited time $t^{PostOpt}$. To build the restricted MIP, we use a number of elite solutions, the feasible solutions selected from the long term memory, $F$, with less than $\delta_p$% gap from the best known feasible solution. The design variables that are closed in all the elite solutions are fixed to 0. A simple iterative procedure is performed to set $\delta_p$% in such a way that at least $f$% of the design variables are set to 0 in the restricted MIP. At the beginning of the procedure, all the solutions of $F$ are selected as elite. If the percentage of the fixed variables is more than $f$%, then the desired percentage is obtained; otherwise, a new set of elite solutions is built by removing the worst solution from the elite solutions until the percentage falls into the desired range. Since there is a huge number of arcs in large-scale MSMCFND instances, in our tests, we set $f$ to a large number (96%) to make the restricted MIP problem tractable.

### 4.4. Summary

The algorithm is presented in Algorithm 2, which starts with an initialization phase where $\mathcal{P}_{LP}$ is solved, and the Lagrange multipliers are initialized to the obtained dual values. In this phase, a graph traversal algorithm reduces the size of the shortest-path subproblems. At the end of initialization phase, a slope scaling procedure is triggered using the final design solution of the cutting plane $\mathcal{P}_{LP}$ as the input design solution $\widehat{y}$.

In the Lagrangian phase, we use a subgradient method to solve the Lagrangian dual. At each iteration, Dijkstra algorithm is used to solve the shortest path subproblems. The subgradient method follows a three-phase approach when handling the design subproblem. The algorithm finds a lower bound at each iteration of the subgradient method. Using the obtained solution of the design subproblem, the slope scaling heuristic, which is empowered by the intensification and diversification procedures, finds feasible solutions. At each subgradient iteration, the best obtained upper bound is used in the subgradient procedure to define a better search direction.

At the end of the subgradient procedure, another slope scaling step is performed using the design solution of the best known lower bound. At the end of the algorithm, the post-optimization phase improves the upper bound using the elite solutions stored in a long-term memory during the algorithm.

## 5. Experimental Results

In this section, we present the set of randomly generated instances used to test the performance of our algorithm. Then, we describe how we have adjusted the parameters of the algorithm. Finally, we analyze the results of the experiments.

### 5.1. Instances

To generate the instances, we use a *time-space network* structure in which the physical nodes (terminals) are repeated for each time period to represent the time dependency. A node in such a network represents a terminal in a specific time period, and each arc represents a transfer from a terminal in a time period to either the same or a different terminal in another time period, see Zhu et al. (2014) for more details.

To generate the instances, let $S = \{1, 2, ..., |S|\}$ and $T = \{1, 2, ..., |T|\}$ be a set of terminals and a set of time periods, respectively. Then, the set of nodes is $N = \{1, 2, ..., |S| \times |T|\}$ where each node $i \in N$ corresponds to terminal $s = \lceil i/|T| \rceil$ and time period $t = ((i-1) \mod |T|) + 1$. For layer $|L|$, an arc is generated between each pair of nodes $(i \in N, j \in N)$ if the corresponding time period of $i$ is less than the corresponding time period of $j$. In other words, we generate an arc from a terminal in a time period to either the same terminal or another terminal in a future time period. The next layers (layer $|L| - 1$ to 1) are built by finding at most 50 paths between each pair of

**Algorithm 2:** Lagrangian-based matheuristic for MSMCFND

Solve $\mathcal{P}_{LP} \rightarrow \widehat{y}$
Initialize Lagrange multipliers
Run primal heuristic using $\widehat{y}$
$it = 0$
**repeat**
  **if** $it < it^{ph1}_{max}$ **then**
    solve LP relaxation of $\mathcal{P}_D \rightarrow \widehat{y}$
  **else if** $it < it^{ph2}_{max}$ **then**
    solve root node of IP $\mathcal{P}_D \rightarrow \widehat{y}$
  **else**
    solve IP $\mathcal{P}_D$ for a limited time $\rightarrow \widehat{y}$
  **end if**
  Solve $\mathcal{P}^k_{SP}$ for each commodity $k \in K$
  Compute lower bound
  Update $LB^{best} \rightarrow \underline{y}$
  **if** $\widehat{y}$ is new **and** the frequency of the number of iterations is equal to $fr^{prim}$ **then**
    Run primal heuristic using $\widehat{y}$
    **if** number of non-improvement iterations in upper bound is greater than $st^p_{max}$ **then**
      Perform diversification
    **end if**
  **end if**
  Update Lagrange multipliers
  $it = it + 1$
**until** one of the termination conditions is met
Run primal heuristic using $\underline{y}$
Run post-optimization by solving $\mathcal{P}_{PO}$

**Table 1 Instances**

| | Number of Instances | $\|L\|$ | $\|A\|$ | $\|S\|$ | $\|T\|$ | $\|N\|$ | $\|K\|$ |
|---|---|---|---|---|---|---|---|
| MLTS1 | 32 | 2 | 20612-30664 | 8,10 | 7 | 56,70 | 100,200 |
| MLTS2 | 32 | 3 | 42768-64162 | 8,10 | 7 | 56,70 | 100,200 |

nodes of the upper layer. We set flow capacity, design capacity, and fixed cost of the arcs randomly. The origins, the destinations and the demands of the commodities are also selected randomly. A random origin-destination pair ($o \in N, d \in N$) is selected as the origin and the destination of a commodity if the corresponding time period of $o$ is less than the one of $d$.

Two different sets of instances are generated randomly using the above instance generator procedure: MLTS1 (MLTS stands for MultiLayer Time-Space) and MLTS2 with two and three layers, respectively. Table 1 summarizes the characteristics of the instances. $|L|$, $|A|$, $|S|$, $|T|$, $|N|$, and $|K|$ are the number of layers, number of arcs, number of terminals, number of time periods, number of nodes, and number of commodities, respectively. Note that in our instances, the flow variable costs are the same for all commodities on a particular arc. Therefore, it is possible to aggregate the commodities whose origins are the same into a single commodity to accelerate the solution of the MMCNF problem of the primal heuristic.

### 5.2. Parameter Setting

We use a two-phase parameter setting strategy to calibrate the parameters. In the first and second phases, we set the subgradient and the primal heuristic parameters, respectively. Tables 2 and 3 display the tested values of the parameters for each phase. We choose 20% of the instances randomly for the parameter setting process.

As mentioned before, the performance of the subgradient method depends on its parameters, the most critical being $\lambda^i$, which is adjusted based on the parameters $\gamma_1$, $\gamma_2$ and $\lambda^0$. To find the best values of the important parameters (Table 2), we fix all the parameters, change one parameter at a time, and select the value that produces better results on average for all the test instances. In Table 2, the selected values are shown in bold. We set the values of the less-important parameters, as suggested in Crainic et al. (2001) to: $\varepsilon = 10^{-7}$ and $\lambda_{min} = 10^{-3}$. We set the maximum number of total subgradient iterations to a large value ($\gamma_3 = 3000$). The value of $it^{ph3}_{max}$ is then equal to $\gamma_3 - (it^{ph2}_{max} + it^{ph3}_{max})$.

**Table 2 Values of parameters tested for subgradient method**

| Name | Description | Tested Values |
|---|---|---|
| $t^{LP}$ | time limit on solving cutting plane $\mathcal{P}_{LP}$ (minutes) | 3, 4, **5**, 6, 7, 8, 9 |
| $t^{Des}$ | time limit on solving design subproblem (minutes) | 0.5, **1**, 5, 10 |
| $\gamma_1$ | parameter to change $\lambda^i$ | 1.5, **2**, 2.5, 3, 3.5 |
| $\gamma_2$ | maximum number of non improvement iterations in lower bound | 20, 40, 60, 80, **100**, 120 |
| $\lambda^0$ | initial value of subgradient scaling factor ($\lambda^i$) | 1, 1.5, **2** |
| $it^{ph1}_{max}$ | number of subgradient iterations to solve LP $\mathcal{P}_D$ | 5%, **10%**, 20%, 30% of $\gamma_3$ |
| $it^{ph2}_{max}$ | number of subgradient iterations to solve $\mathcal{P}_D$ in root node | 5%, **10%**, 20%, 30% of $\gamma_3$ |

Table 3 shows the values that are tested to set the parameters of the primal heuristic. We use the same strategy as for the subgradient procedure: we fix all the parameters, change one of them at a time, and select the best value. The table shows the selected values in bold. Some of the parameters of the primal heuristic can be set initially without testing different values. $t^{PostOpt}$ is determined based on two previous timing parameters, $t^{LP}$ and $t^{Lag}$. The big $M$ parameter has to be large enough to ensure not to select the design variables that have value zero in the design subproblem solution, and can be obtained using the following formula:

$$M = 10 \times \frac{\max_{l \in L, a \in A_l} \{f_{al}\}}{\min_{l \in L, a \in A_l} \{f_{al}\}} \tag{24}$$

### 5.3. Upper Bound Analysis

The proposed Lagrangian heuristic is evaluated using MLTS1 and MLTS2 instances. We compare the results of the proposed algorithm to those obtained by CPLEX 12.6. We add our proposed preprocessing procedure before starting the CPLEX solver. We also consider different scenarios to

**Table 3 Values of parameters tested for the primal heuristic**

| Name | Description | Tested Values |
|---|---|---|
| $fr^{prim}$ | primal heuristic frequency | 5, 10, 15, **20**, 25 |
| $t^{Lag}$ | time limit on Lagrangian procedure including subgradient and primal heuristic (hours) | 5, 6, **7**, 8 |
| $it^{s}_{max}$ | maximum number of slope scaling iterations in each primal heuristic step | 10, **15**, 20 |
| $n$ | number of intensification iterations | 1, **4**, 8 |
| $st^{p}_{max}$ | maximum number of primal heuristic steps with no improvement in upper bound | 3, **5**, 8, 10 |
| $it^{d}_{max}$ | total number of iterations in each diversification procedure | 5, **10**, 15, 20 |
| $st^{d}_{max}$ | total number of diversification steps | 1, **3**, 4, 5 |

**Table 4 CPLEX and the proposed algorithm comparison for MLTS1 or 2-layer instances.**

| | | | Optimality Gap % | | LB/UB Gap % | |
|---|---|---|---|---|---|---|
| # | (\|A1\|, \|A2\|, \|N\|, \|K\|, FCTI, DCTI, CDI) | T(sec) | $\text{Gap}^{\text{CUB}}_{\text{CLB}}$ | $\text{Gap}^{\text{LUB}}_{\text{LLB}}$ | $\text{Gap}^{\text{CLB}}_{\text{LLB}}$ | $\text{Gap}^{\text{CUB}}_{\text{LUB}}$ |
| 1 | (19952,660, 56, 100, L, L, F) | 13055.5 | 6.82 | 17.27 | 10.79 | 0.48 |
| 2 | (19952,660, 56, 100, L, L, V) | 26895.6 | 15.90 | 18.76 | 7.57 | **-4.32** |
| 3 | (19952,660, 56, 100, L, T, F) | 14321.6 | 13.93 | 20.84 | 9.53 | **-1.64** |
| 4 | (19952,660, 56, 100, L, T, V) | Limit | 11.22 | 15.86 | 4.67 | 0.59 |
| 5 | (19952,660, 56, 100, T, L, F) | Limit | 16.01 | 19.32 | 6.94 | **-3.12** |
| 6 | (19952,660, 56, 100, T, L, V) | Limit | 14.27 | 16.05 | 6.30 | **-4.31** |
| 7 | (19952,660, 56, 100, T, T, F) | Limit | 16.40 | 21.73 | 9.84 | **-3.69** |
| 8 | (19952,660, 56, 100, T, T, V) | Limit | 16.22 | 16.36 | 3.51 | **-3.35** |
| 9 | (19952,660, 56, 200, L, L, F) | 34545.7 | 26.55 | **25.02** | 9.12 | **-10.98** |
| 10 | (19952,660, 56, 200, L, L, V) | Limit | 75.52 | **19.70** | 7.44 | **-71.78** |
| 11 | (19952,660, 56, 200, L, T, F) | Limit | 83.65 | **32.25** | 13.78 | **-79.20** |
| 12 | (19952,660, 56, 200, L, T, V) | Limit | 20.32 | 21.22 | 6.74 | **-5.68** |
| 13 | (19952,660, 56, 200, T, L, F) | Limit | 22.48 | **19.34** | 4.85 | **-8.55** |
| 14 | (19952,660, 56, 200, T, L, V) | Limit | 26.34 | **18.82** | 4.91 | **-13.72** |
| 15 | (19952,660, 56, 200, T, T, F) | Limit | 73.67 | **20.22** | 3.57 | **-68.17** |
| 16 | (19952,660, 56, 200, T, T, V) | Limit | 76.87 | **23.52** | 6.21 | **-71.63** |
| 17 | (29794,870, 70, 100, L, L, F) | 31989.3 | 9.83 | 16.90 | 9.10 | **-1.36** |
| 18 | (29794,870, 70, 100, L, L, V) | 32681.7 | 6.97 | 12.46 | 7.35 | **-1.53** |
| 19 | (29794,870, 70, 100, L, T, F) | Limit | 5.94 | 13.78 | 8.29 | 0.05 |
| 20 | (29794,870, 70, 100, L, T, V) | 25351.6 | 5.97 | 10.00 | 5.56 | **-1.34** |
| 21 | (29794,870, 70, 100, T, L, F) | Limit | 8.05 | 16.22 | 8.36 | 0.57 |
| 22 | (29794,870, 70, 100, T, L, V) | Limit | 15.59 | 18.32 | 7.01 | **-3.90** |
| 23 | (29794,870, 70, 100, T, T, F) | Limit | 10.49 | 15.45 | 6.68 | **-1.21** |
| 24 | (29794,870, 70, 100, T, T, V) | Limit | 14.27 | 17.18 | 6.91 | **-3.65** |
| 25 | (29794,870, 70, 200, L, L, F) | Limit | 95.63 | **27.31** | 13.07 | **-94.78** |
| 26 | (29794,870, 70, 200, L, L, V) | Limit | 94.85 | **15.23** | 6.61 | **-94.33** |
| 27 | (29794,870, 70, 200, L, T, F) | Limit | 95.86 | **21.33** | 8.04 | **-95.17** |
| 28 | (29794,870, 70, 200, L, T, V) | Limit | 94.59 | **20.43** | 5.36 | **-93.57** |
| 29 | (29794,870, 70, 200, T, L, F) | Limit | 29.50 | **24.91** | 11.16 | **-16.59** |
| 30 | (29794,870, 70, 200, T, L, V) | Limit | 94.35 | **14.90** | 6.15 | **-93.77** |
| 31 | (29794,870, 70, 200, T, T, F) | Limit | 79.41 | **22.93** | 8.71 | **-75.61** |
| 32 | (29794,870, 70, 200, T, T, V) | Limit | 68.40 | **17.30** | 6.67 | **-64.34** |
| | | Average | 38.93 | 19.09 | 7.52 | -30.93 |
| | | Minimum | 5.94 | 10.00 | 3.51 | -95.17 |
| | | Maximum | 95.86 | 32.25 | 13.78 | 0.59 |

determine whether it is suitable to add the strong linking constraints (4) and the upward design linking constraints (6) to the model of CPLEX or not. The tested scenarios includes adding each set

**Table 5 CPLEX and the proposed algorithm comparison for MLTS2 or 3-layer instances.**

| # | (\|A1\|, \|A2\|, \|A3\|, \|N\|, \|K\|, FCTI, DCTI, CDI) | T(sec) | Optimality Gap % | | LB/UB Gap % | |
|---|---|---|---|---|---|---|
| | | | $\text{Gap}^{\text{CUB}}_{\text{CLB}}$ | $\text{Gap}^{\text{LUB}}_{\text{LLB}}$ | $\text{Gap}^{\text{CLB}}_{\text{LLB}}$ | $\text{Gap}^{\text{CUB}}_{\text{LUB}}$ |
| 33 | (22156,19952, 660, 56, 100, L, L, F) | Limit | 19.57 | 20.00 | 6.09 | **-5.58** |
| 34 | (22156,19952, 660, 56, 100, L, L, V) | Limit | 29.57 | **24.57** | 7.75 | **-13.87** |
| 35 | (22156,19952, 660, 56, 100, L, T, F) | Limit | 26.84 | **22.68** | 5.53 | **-10.62** |
| 36 | (22156,19952, 660, 56, 100, L, T, V) | Limit | 12.53 | 18.34 | 6.27 | 0.40 |
| 37 | (22156,19952, 660, 56, 100, T, L, F) | Limit | 21.73 | **15.95** | 5.49 | **-11.98** |
| 38 | (22156,19952, 660, 56, 100, T, L, V) | Limit | 20.92 | **17.17** | 4.62 | **-8.94** |
| 39 | (22156,19952, 660, 56, 100, T, T, F) | Limit | 23.52 | **16.76** | 5.27 | **-12.96** |
| 40 | (22156,19952, 660, 56, 100, T, T, V) | Limit | 21.24 | **12.87** | 4.66 | **-13.81** |
| 41 | (22156,19952, 660, 56, 200, L, L, F) | Limit | 98.32 | **22.76** | 4.30 | **-97.91** |
| 42 | (22156,19952, 660, 56, 200, L, L, V) | Limit | 97.96 | **21.03** | 2.86 | **-97.49** |
| 43 | (22156,19952, 660, 56, 200, L, T, F) | Limit | 98.13 | **22.12** | 4.65 | **-97.71** |
| 44 | (22156,19952, 660, 56, 200, L, T, V) | Limit | 97.90 | **21.92** | 3.41 | **-97.41** |
| 45 | (22156,19952, 660, 56, 200, T, L, F) | Limit | 97.66 | **18.15** | 3.64 | **-97.24** |
| 46 | (22156,19952, 660, 56, 200, T, L, V) | Limit | 72.96 | **17.85** | 3.62 | **-68.28** |
| 47 | (22156,19952, 660, 56, 200, T, T, F) | Limit | 97.60 | **20.17** | 4.81 | **-97.13** |
| 48 | (22156,19952, 660, 56, 200, T, T, V) | Limit | 66.11 | **18.94** | 4.06 | **-59.88** |
| 49 | (33498,29794, 870, 70, 100, L, L, F) | Limit | 22.30 | 26.30 | 11.40 | **-6.59** |
| 50 | (33498,29794, 870, 70, 100, L, L, V) | Limit | 15.19 | 22.67 | 11.58 | **-3.02** |
| 51 | (33498,29794, 870, 70, 100, L, T, F) | Limit | 21.70 | 31.26 | 11.66 | 0.64 |
| 52 | (33498,29794, 870, 70, 100, L, T, V) | Limit | 18.46 | 20.16 | 7.89 | **-5.93** |
| 53 | (33498,29794, 870, 70, 100, T, L, F) | Limit | 19.92 | 20.06 | 7.61 | **-7.45** |
| 54 | (33498,29794, 870, 70, 100, T, L, V) | Limit | 9.57 | 13.52 | 6.00 | **-1.71** |
| 55 | (33498,29794, 870, 70, 100, T, T, F) | Limit | 22.48 | **17.54** | 6.44 | **-12.05** |
| 56 | (33498,29794, 870, 70, 100, T, T, V) | Limit | 34.98 | **18.45** | 6.72 | **-25.63** |
| 57 | (33498,29794, 870, 70, 200, L, L, F) | Limit | 98.53 | **36.89** | 16.18 | **-98.05** |
| 58 | (33498,29794, 870, 70, 200, L, L, V) | Limit | 98.32 | **27.39** | 10.97 | **-97.94** |
| 59 | (33498,29794, 870, 70, 200, L, T, F) | Limit | 98.32 | **28.85** | 13.06 | **-97.95** |
| 60 | (33498,29794, 870, 70, 200, L, T, V) | Limit | 98.33 | **29.68** | 13.99 | **-97.95** |
| 61 | (33498,29794, 870, 70, 200, T, L, F) | Limit | 61.31 | **22.58** | 6.87 | **-53.46** |
| 62 | (33498,29794, 870, 70, 200, T, L, V) | Limit | 64.62 | **20.52** | 6.36 | **-58.32** |
| 63 | (33498,29794, 870, 70, 200, T, T, F) | Limit | 61.60 | **24.01** | 6.28 | **-52.64** |
| 64 | (33498,29794, 870, 70, 200, T, T, V) | Limit | 71.23 | **23.47** | 9.51 | **-65.98** |
| | | Average | 53.73 | 21.71 | 7.17 | -46.08 |
| | | Minimum | 9.57 | 12.87 | 2.86 | -98.05 |
| | | Maximum | 98.53 | 36.89 | 16.18 | 0.64 |

of these constraints to the model of CPLEX: 1) a priori, 2) as "user cuts", 3) as "lazy constraints", and 4) using cut callback functions of CPLEX. The tests show that the best case for CPLEX is not to add any of the upward design linking constraints but to add strong linking constraints as "user cuts" to the model.

The results are summarized in Tables 4 and 5 for MLTS1 and MLTS2 instances. We fix the total time limit to 10 hours for CPLEX and for the proposed algorithm. The second column in these tables shows the characteristics of the instances:

- $|A_l|$: number of arcs in layer $l$;
- $|N|$: number of nodes;

- $|K|$: number of commodities;
- FCTI: Flow Capacity Tightness Index defined as the flow capacity average, $(\sum_{l\in L}\sum_{a\in A_l} u_{al})/|A|$, divided by the total demand, $\sum_{k\in K} d^k$, L stands for loose flow capacity (FCTI $\geq 0.05$), while T refers to tight flow capacity (FCTI $< 0.05$);
- DCTI: Design Capacity Tightness Index defined as the design capacity average, $(\sum_{l\in L,l\geq 2}\sum_{a\in A_l} v_{al})/(|A|-|A_1|)$, divided by $B^{al}_{ave} = \sum_{l\in L,l\geq 2}\sum_{a\in A_l}|B^{al}|/(|A|-|A_1|)$, L stands for loose design capacity (DCTI $\geq 0.3$), while T refers to tight design capacity (DCTI $< 0.3$);
- CDI: Cost Dominance Index defined as the variable flow cost average, $(\sum_{k\in K}\sum_{a\in A_1} c^k_{a1})/|A_1|$, divided by the design cost average, $(\sum_{l\in L}\sum_{a\in A_l} f_{al})/|A|$, F refers to the cases where the fixed costs are predominant relatively to the variable costs (CDI $< 0.01$), and V means the reverse (CDI $\geq 0.01$).

The column T(sec) is the total run time of the proposed algorithm. CPLEX was not able to converge to optimality on any of the instances in 10 hours (time limit). Therefore, the run times of CPLEX are not reported in these tables (CPLEX run time is 10h). Let CUB and CLB be the upper and lower bounds obtained by CPLEX, respectively. Let also LUB and LLB be the upper and lower bounds obtained by the proposed algorithm, respectively. Columns $\text{Gap}^{\text{CUB}}_{\text{CLB}} = 100 \times (\text{CUB} - \text{CLB})/\text{CUB}$ and $\text{Gap}^{\text{LUB}}_{\text{LLB}} = 100 \times (\text{LUB} - \text{LLB})/\text{LUB}$ are the percentages of the optimality gaps reported by CPLEX and the proposed algorithm, respectively. Column $\text{Gap}^{\text{CLB}}_{\text{LLB}} = 100 \times (\text{CLB} - \text{LLB})/\text{LLB}$ is the percentage of the gap between the lower bound of the CPLEX and the obtained lower bound of the algorithm. Column $\text{Gap}^{\text{LUB}}_{\text{CUB}} = 100 \times (\text{LUB} - \text{CUB})/\text{CUB}$ is the percentage of the gap between the upper bound of CPLEX and the upper bound obtained by the algorithm. A negative value indicates that the Lagrangian-based matheuristic method found a better feasible solution than CPLEX.

The results show that the algorithm is better than CPLEX in almost all the instances (90%) in terms of the upper bounds (last column of Tables 4 and 5). The averages of the improvements in the upper bounds are 30.93% and 46.08% for MLTS1 and MLTS2 instances, respectively. The algorithm improves the upper bounds by more than 10% in 57% of the instances.

The proposed algorithm finds better optimality gaps than CPLEX in 60% of the instances (bold optimality gaps in column $\text{Gap}^{\text{LUB}}_{\text{LLB}}$). The goal of the proposed algorithm, as a matheuristic, is to find high-quality upper bounds. However, one of the advantages of this algorithm is that it also computes lower bounds. The obtained lower bounds enable us to evaluate the quality of the obtained upper bounds, particularly for the instances where CPLEX ends up with more than 90% optimality gap.

Table 6 shows the analysis of the performance of the algorithm according to the number of layers, number of arcs and number of commodities. The table shows that the complexity of the instances

increases when increasing the number of layers, the number of arcs, the number of nodes and, even more significantly, the number of commodities. The algorithm improves the upper bound up to 71.66% in average for the instances with 200 commodities, while the average is 5.34% for the instances with 100 commodities.

**Table 6 Performance analysis of the proposed algorithm in terms of $|K|$, $|L|$, $|A|$ and $|N|$.**

| | $\|K\|$ | | $\|L\|$ | | $\|A\|$ | | | | $\|N\|$ | |
|---|---|---|---|---|---|---|---|---|---|---|
| | 100 | 200 | 2 | 3 | 20612 | 30664 | 42768 | 64162 | 56 | 70 |
| $\text{Gap}_{\text{CUB}}^{\text{LUB}}$ Average | -5.34 | -71.66 | -30.93 | -46.08 | -21.82 | -40.03 | -49.40 | -42.75 | -35.61 | -41.39 |

### 5.4. Lower Bound Analysis

Tables 7 and 8 show the lower bound analysis of the proposed Lagrangian relaxation for MLTS1 and MLTS2 instances, respectively. Column "Lag Time" shows the elapsed time of the bounding procedure including the initialization phase and the subgradient method without the primal heuristic and the post optimization phase. Column "Strong LP" shows 1) the elapsed time of the strong LP relaxation solved by CPLEX where the the strong linking constraints (4) and the upward design linking constraints (6) are added a priori to the model, and 2) the obtained Lagrangian LB gap to the strong LP lower bound. Column "Root Node" presents 1) the elapsed time of the root node of a branch-and-bound algorithm solved by CPLEX, and 2) the obtained Lagrangian LB gap to the root node lower bound.

For the "Strong LP" case, we also tested a cutting plane approach where inequalities (4) and (6) are added iteratively. The elapsed time of the cutting plane approach is much worse than the "a priori" approach. For the "Root Node" case, we turn off the heuristic of CPLEX but allow other possible valid inequalities to be added to the model. We introduce strong inequalities as "user cuts" to the root node of CPLEX. We do not add the upward design linking constraints because it takes a long time (more than 10 hours) to add them as user cuts. When CPLEX adds some other cuts in the root note, the results can be better than the strong LP. However, since CPLEX is not adding all the strong inequalities, as well as the upward linking constraints, it is possible that the root lower bound of CPLEX is worse than the strong LP lower bound.

Although the lower bounds of the subgradient procedure are around 6% away, on average, from the lower bounds of the strong LP, its average elapsed time is much less. The average elapsed time of the subgradient process for MLTS2 instances, for example, is 1 hour and 2 minutes while it is 10 hours and 2 hours for the strong LP and the root node, respectively. The standard deviation of the elapsed time is also much less than those of the strong LP and the root node. For MLTS2 instances, the elapsed time of the subgradient procedure varies from 36 to 120 minutes, with

**Table 7 Lower bound analysis: comparing proposed Lagrangian to CPLEX root node and strong LP for MLTS1.**

| | | | Strong LP | | Root Node | |
|---|---|---|---|---|---|---|
| # | ($\|A1\|$, $\|A2\|$, $\|N\|$, $\|K\|$, FCTI, DCTI, CDI) | Lag Time | Lag gap | Time | Lag gap | Time |
| 1 | (19952,660, 56, 100, L, L, F) | 0h 14m | 9.08 | 0h 35m | 8.92 | 0h 11m |
| 2 | (19952,660, 56, 100, L, L, V) | 0h 14m | 8.38 | 0h 31m | 8.28 | 0h 30m |
| 3 | (19952,660, 56, 100, L, T, F) | 0h 14m | 8.59 | 0h 21m | 8.53 | 0h 11m |
| 4 | (19952,660, 56, 100, L, T, V) | 0h 14m | 4.00 | 0h 20m | 3.83 | 0h 21m |
| 5 | (19952,660, 56, 100, T, L, F) | 0h 14m | 5.12 | 0h 40m | 5.07 | 0h 7m |
| 6 | (19952,660, 56, 100, T, L, V) | 0h 14m | 5.61 | 0h 25m | 5.62 | 0h 8m |
| 7 | (19952,660, 56, 100, T, T, F) | 0h 20m | 7.79 | 0h 28m | 7.77 | 0h 17m |
| 8 | (19952,660, 56, 100, T, T, V) | 0h 20m | 2.82 | 0h 21m | 2.88 | 0h 11m |
| 9 | (19952,660, 56, 200, L, L, F) | 0h 32m | 6.00 | 5h 35m | 5.91 | 0h 37m |
| 10 | (19952,660, 56, 200, L, L, V) | 0h 31m | 6.48 | 3h 38m | 6.26 | 0h 37m |
| 11 | (19952,660, 56, 200, L, T, F) | 0h 29m | 6.99 | 5h 20m | 6.83 | 1h 46m |
| 12 | (19952,660, 56, 200, L, T, V) | 0h 19m | 5.58 | 2h 27m | 5.51 | 1h 5m |
| 13 | (19952,660, 56, 200, T, L, F) | 0h 19m | 4.48 | 3h 49m | 4.49 | 0h 47m |
| 14 | (19952,660, 56, 200, T, L, V) | 0h 19m | 4.50 | 2h 31m | 4.51 | 0h 43m |
| 15 | (19952,660, 56, 200, T, T, F) | 0h 30m | 3.13 | 4h 3m | 3.19 | 0h 18m |
| 16 | (19952,660, 56, 200, T, T, V) | 0h 19m | 4.47 | 2h 13m | 4.47 | 0h 25m |
| 17 | (29794,870, 70, 100, L, L, F) | 0h 18m | 7.91 | 0h 53m | 7.75 | 0h 40m |
| 18 | (29794,870, 70, 100, L, L, V) | 0h 29m | 6.71 | 0h 33m | 6.63 | 0h 47m |
| 19 | (29794,870, 70, 100, L, T, F) | 0h 28m | 5.42 | 0h 27m | 5.21 | 0h 20m |
| 20 | (29794,870, 70, 100, L, T, V) | 0h 29m | 4.65 | 0h 22m | 4.53 | 1h 24m |
| 21 | (29794,870, 70, 100, T, L, F) | 0h 19m | 7.87 | 0h 56m | 7.81 | 0h 38m |
| 22 | (29794,870, 70, 100, T, L, V) | 0h 20m | 3.42 | 0h 31m | 3.35 | 1h 12m |
| 23 | (29794,870, 70, 100, T, T, F) | 0h 19m | 5.41 | 0h 51m | 5.29 | 0h 29m |
| 24 | (29794,870, 70, 100, T, T, V) | 0h 19m | 3.76 | 0h 32m | 3.66 | 0h 38m |
| 25 | (29794,870, 70, 200, L, L, F) | 0h 28m | 8.41 | 3h 23m | 8.23 | 0h 50m |
| 26 | (29794,870, 70, 200, L, L, V) | 0h 27m | 4.78 | 3h 57m | 4.69 | 0h 48m |
| 27 | (29794,870, 70, 200, L, T, F) | 0h 28m | 5.54 | 4h 13m | 5.40 | 1h 6m |
| 28 | (29794,870, 70, 200, L, T, V) | 0h 28m | 3.14 | 2h 47m | 3.02 | 0h 59m |
| 29 | (29794,870, 70, 200, T, L, F) | 0h 29m | 8.64 | 7h 57m | 8.62 | 1h 15m |
| 30 | (29794,870, 70, 200, T, L, V) | 0h 26m | 4.96 | 1h 33m | 4.92 | 0h 56m |
| 31 | (29794,870, 70, 200, T, T, F) | 0h 26m | 7.53 | 3h 35m | 7.54 | 0h 44m |
| 32 | (29794,870, 70, 200, T, T, V) | 0h 27m | 5.30 | 2h 13m | 5.31 | 0h 57m |
| | Average | 0h 23m | 5.83 | 2h 7m | 5.75 | 0h 41m |
| | Minimum | 0h 14m | 2.82 | 0h 20m | 2.88 | 0h 7m |
| | Maximum | 0h 32m | 9.08 | 7h 57m | 8.92 | 1h 46m |
| | Standard Deviation | 0h 6m | | 1h 57m | | 0h 24m |

standard deviation equal to 26 minutes, while it varies from 1 hour to 25 hours for CPLEX to solve the strong LP, with standard deviation equal to 7 hours. Finding high-quality solutions is the primary goal of this paper, but the proposed algorithm efficiently generates effective lower bounds to evaluate the obtained feasible solutions.

## 6. Conclusions

We have proposed a general formulation for the MSMCFND as well as an efficient algorithm to solve large-scale MSMCFND instances. The algorithm has been empowered by the preprocessing

**Table 8 Lower bound analysis: comparing proposed Lagrangian to CPLEX root node and strong LP for MLTS2.**

| | | | Strong LP | | Root Node | |
|---|---|---|---|---|---|---|
| # | ($\|A1\|$, $\|A2\|$, $\|N\|$, $\|K\|$, FCTI, DCTI, CDI) | Lag Time | Lag gap | Time | Lag gap | Time |
| 33 | (19952,660, 56, 100, L, L, F) | 0h 36m | 5.23 | 6h 27m | 5.64 | 0h 32m |
| 34 | (19952,660, 56, 100, L, L, V) | 0h 37m | 7.24 | 6h 43m | 7.35 | 0h 52m |
| 35 | (19952,660, 56, 100, L, T, F) | 0h 40m | 4.69 | 3h 25m | 5.06 | 0h 58m |
| 36 | (19952,660, 56, 100, L, T, V) | 0h 37m | 7.15 | 1h 0m | 7.50 | 0h 56m |
| 37 | (19952,660, 56, 100, T, L, F) | 0h 36m | 4.65 | 3h 35m | 5.29 | 1h 30m |
| 38 | (19952,660, 56, 100, T, L, V) | 0h 38m | 3.59 | 4h 26m | 4.31 | 1h 3m |
| 39 | (19952,660, 56, 100, T, T, F) | 0h 43m | 3.57 | 2h 41m | 4.62 | 0h 26m |
| 40 | (19952,660, 56, 100, T, T, V) | 0h 39m | 3.52 | 2h 3m | 4.22 | 0h 34m |
| 41 | (19952,660, 56, 200, L, L, F) | 0h 48m | 3.96 | 21h 38m | 4.28 | 1h 19m |
| 42 | (19952,660, 56, 200, L, L, V) | 0h 46m | 2.25 | 12h 31m | 2.53 | 2h 11m |
| 43 | (19952,660, 56, 200, L, T, F) | 0h 56m | 3.41 | 15h 37m | 3.62 | 1h 34m |
| 44 | (19952,660, 56, 200, L, T, V) | 0h 45m | 2.86 | 15h 13m | 3.15 | 1h 24m |
| 45 | (19952,660, 56, 200, T, L, F) | 0h 43m | 2.56 | 16h 33m | 3.43 | 1h 18m |
| 46 | (19952,660, 56, 200, T, L, V) | 0h 43m | 2.43 | 9h 57m | 3.26 | 1h 11m |
| 47 | (19952,660, 56, 200, T, T, F) | 1h 1m | 3.42 | 21h 40m | 3.96 | 2h 8m |
| 48 | (19952,660, 56, 200, T, T, V) | 0h 45m | 3.04 | 15h 11m | 3.88 | 0h 52m |
| 49 | (29794,870, 70, 100, L, L, F) | 0h 57m | 6.90 | 8h 20m | 7.17 | 3h 15m |
| 50 | (29794,870, 70, 100, L, L, V) | 0h 58m | 4.47 | 2h 42m | 4.45 | 3h 6m |
| 51 | (29794,870, 70, 100, L, T, F) | 1h 34m | 3.89 | 6h 17m | 3.98 | 2h 52m |
| 52 | (29794,870, 70, 100, L, T, V) | 1h 35m | 4.79 | 3h 25m | 4.85 | 2h 36m |
| 53 | (29794,870, 70, 100, T, L, F) | 0h 50m | 6.49 | 2h 42m | 7.19 | 1h 55m |
| 54 | (29794,870, 70, 100, T, L, V) | 0h 47m | 4.22 | 2h 30m | 4.94 | 1h 13m |
| 55 | (29794,870, 70, 100, T, T, F) | 1h 34m | 4.49 | 1h 30m | 5.14 | 1h 59m |
| 56 | (29794,870, 70, 100, T, T, V) | 1h 29m | 5.20 | 3h 35m | 6.04 | 2h 49m |
| 57 | (29794,870, 70, 200, L, L, F) | 1h 57m | 4.91 | 15h 55m | 5.13 | 2h 38m |
| 58 | (29794,870, 70, 200, L, L, V) | 1h 9m | 4.22 | 17h 4m | 4.57 | 2h 42m |
| 59 | (29794,870, 70, 200, L, T, F) | 1h 7m | 5.21 | 25h 1m | 5.55 | 3h 37m |
| 60 | (29794,870, 70, 200, L, T, V) | 1h 59m | 3.10 | 21h 55m | 3.40 | 5h 24m |
| 61 | (29794,870, 70, 200, T, L, F) | 1h 5m | 3.04 | 18h 10m | 3.79 | 6h 40m |
| 62 | (29794,870, 70, 200, T, L, V) | 1h 12m | 3.15 | 18h 37m | 4.52 | 4h 9m |
| 63 | (29794,870, 70, 200, T, T, F) | 2h 0m | 3.15 | 19h 35m | 4.15 | 3h 3m |
| 64 | (29794,870, 70, 200, T, T, V) | 1h 9m | 4.00 | 8h 58m | 4.58 | 1h 59m |
| | Average | 1h 2m | 4.21 | 10h 28m | 4.74 | 2h 9m |
| | Minimum | 0h 36m | 2.24 | 1h 0m | 2.53 | 0h 26m |
| | Maximum | 2h 0m | 7.24 | 25h 1m | 7.50 | 6h 40m |
| | Standard Deviation | 0h 26m | | 7h 32m | | 1h 25m |

procedure, the intensification and diversification steps, and the post-optimization phase. We can summarize the obtained results as follows:

- The algorithm produces better upper bounds than CPLEX in 90% of the instances.
- The algorithm is better than CPLEX in 60% of the instances in terms of the optimality gap.
- The algorithm improves the upper bound by more than 10% in 57% of the instances.
- The complexity of the instances increases by increasing the number of layers, the number of arcs and, even more significantly, the number of commodities.

- The performance of the Lagrangian relaxation is robust in either the easy or the difficult instances in terms of computational time, i.e., the standard deviation of the elapsed time is also much less than those of the strong LP and the root node solved by CPLEX.

Four main interesting research avenues are: 1) Improving the algorithm using more efficient non-differentiable optimization approaches such as the bundle method, and embedding the algorithm into a branch-and-bound structure to derive an exact solution methodology, 2) Exploring new solution methods particularly those that have been successfully applied before to the MCFND, 3) Proposing branch-and-cut approaches by developing new valid inequalities based on the multilayer structure of the problem, and 4) Considering problems where there are commodities to be routed in all layers.

## Acknowledgments

While working on this project, the second author was Adjunct Professor with the Département d'informatique et de recherche opérationnelle, Université de Montréal. The first author gratefully acknowledges the support of the Université de Montréal, through its end-of-doctoral-studies scholarship and the excellence scholarship of the Département d'informatique et de recherche opérationnelle, as well as the support of CIRRELT through its doctoral excellence scholarship. Partial funding for this project has also been provided by the Natural Sciences and Engineering Council of Canada (NSERC), through its Discovery Grant program. We also gratefully acknowledge the support of Fonds de recherche du Québec through their infrastructure grants.

## References

Akhavan Kazemzadeh, M. R., Crainic, T. G., Gendron, B., 2018. A survey of multilayer network design. Publication CIRRELT 2018.

Camerini, P. M., Fratta, L., Maffioli, F., 1975. On improving relaxation methods by modified gradient techniques. In: Nondifferentiable optimization. Springer, pp. 26–34.

Chouman, M., Crainic, T. G., Gendron, B., 2016. Commodity representations and cut-set-based inequalities for multicommodity capacitated fixed-charge network design. Transportation Science 51 (2), 650–667.

Chouman, M., Crainic, T. G., Gendron, B., 2017. The impact of filtering in a branch-and-cut algorithm for multicommodity capacitated fixed charge network design. EURO Journal on Computational Optimization 6 (2), 1–42.

Cordeau, J.-F., Stojković, G., Soumis, F., Desrosiers, J., 2001. Benders decomposition for simultaneous aircraft routing and crew scheduling. Transportation Science 35 (4), 375–388.

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

Crainic, T. G., Gendron, B., Hernu, G., 2004. A slope scaling/lagrangean perturbation heuristic with long-term memory for multicommodity capacitated fixed-charge network design. Journal of Heuristics 10 (5), 525–545.

Crainic, T. G., Hewitt, M., Toulouse, M., Vu, D. M., 2014. Service network design with resource constraints. Transportation Science 50 (4), 1380–1393.

Dahl, G., Martin, A., Stoer, M., 1999. Routing through virtual paths in layered telecommunication networks. Operations Research 47 (5), 693–702.

Geoffrion, A. M., 1974. Lagrangean relaxation for integer programming. In: Approaches to integer programming. Springer, pp. 82–114.

Holmberg, K., Yuan, D., 2000. A Lagrangian heuristic based branch-and-bound approach for the capacitated network design problem. Operations Research 48 (3), 461–481.

Kim, D., Pardalos, P. M., 1999. A solution approach to the fixed charge network flow problem using a dynamic slope scaling procedure. Operations Research Letters 24 (4), 195–203.

Kliewer, G., Timajev, L., 2005. Relax-and-cut for capacitated network design. In: Stølting Brodal, G., Leonardi, S. (Eds.), Algorithms–ESA 2005, 13th Annual European Symposium. Springer, pp. 47–58.

Knippel, A., Lardeux, B., 2007. The multi-layered network design problem. European Journal of Operational Research 183 (1), 87–99.

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

Minoux, M., 1989. Networks synthesis and optimum network design problems: Models, solution methods and applications. Networks 19 (3), 313–360.

Padberg, M. W., 1973. On the facial structure of set packing polyhedra. Mathematical Programming 5 (1), 199–215.

Sellmann, M., Kliewe, G., et al., 2002. Lagrangian cardinality cuts and variable fixing for capacitated network design. In: Algorithms—ESA 2002. Springer, pp. 845–858.

Zhu, E., Crainic, T. G., Gendreau, M., 2014. Scheduled service network design for freight rail transportation. Operations Research 62 (2), 383 – 400.