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 Computational Comparison of Flow Formulations for the Capacitated Location-Routing Problem

Claudio Contardo
Jean-François Cordeau
Bernard Gendron

August 2011

CIRRELT-2011-47

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

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

www.cirrelt.ca

Université Laval · UQÀM · HEC Montréal · École Polytechnique Montréal · Université de Montréal

A Computational Comparison of Flow Formulations for the Capacitated Location-Routing Problem

Claudio Contardo[1,2,*], Jean-François Cordeau[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 Logistics and Operations Management, HEC Montréal, 3000 Côte-Sainte-Catherine, Montréal, Canada H3T 2A7

Abstract. In this paper we present a computational comparison of four different flow formulations for the capacitated location-routing problem. We introduce three new flow formulations for the problem, namely a two-index two-commodity flow formulation, a three-index vehicle-flow formulation and a three-index two-commodity flow formulation. We also consider the known two-index vehicle-flow formulation and extend it by considering new families of valid inequalities and separation algorithms. We introduce new branch-and-cut algorithms for each of the formulations and compare them on a wide number of instances. Our results show that compact formulations can produce tight gaps and solve many instances quickly, whereas three-index formulations scale better in time.

Keywords. Location-routing, flow formulations, branch-and-cut.

Acknowledgements. The authors would like to thank the Natural Sciences and Engineering Research Council of Canada (NSERC) and the Fonds québécois de la recherche sur la nature et les technologies (FQRNT) for their financial support.

---

* Corresponding author: Claudio.Contardo@cirrelt.ca

Dépôt légal – Bibliothèque et Archives nationales du Québec
Bibliothèque et Archives Canada, 2011

1 Introduction

In the *Capacitated Location-Routing Problem* (CLRP) we are given a set I of potential facility locations and a set J of customers. The problem consists in selecting a subset of facilities and in designing vehicle routes around these facilities so that every customer is visited exactly once. Each facility $i \in I$ has a capacity b_i and a fixed cost f_i. The fleet is unlimited and each vehicle has a capacity Q. Each customer $j \in J$ has a demand d_j. We define a graph $G = (V, E)$ where $V = I \cup J$ is the vertex set and E is the edge set. With every edge $\{i, j\} \in E$ is associated a cost c_{ij} for using edge $\{i, j\}$. Each route must start from and return to the same selected facility, and the sum of the demands of the customers served along a route cannot exceed Q. In addition, the total demand of the customers served in routes from facility i cannot exceed b_i. The objective consists in minimizing the sum of the routing costs and the fixed costs associated with the selected facilities.

A three-index mixed-integer programming formulation for the CLRP was introduced by Perl and Daskin [22] for the general case of an asymmetric network, heterogeneous vehicles and heterogeneous facilities. Its linear programming relaxation does not, however, provide lower bounds that are tight enough to be used within a branch-and-cut algorithm. Laporte et al. [15] proposed the first two-index vehicle-flow formulation for the LRP with uncapacitated facilities (ULRP). They have considered vehicle capacity cuts (CC) as well as chain barring constraints (CBC) and, by means of a branch-and-bound algorithm, were able to solve small size instances. Based on this work, Belenguer et al. [6] recently proposed a two-index integer programming formulation for the CLRP, providing strengthened versions of the CC and the CBC. They also introduced a new version of the facility capacity inequalities (FCI) and other constraints such as co-circuit constraints and depot degree constraints. The lower bounds obtained by their algorithm are very tight and suggest that by improving the separation algortithms as well as developing new families of valid inequalities, the cutting-plane approach would lead to a successful methodology for solving medium or even large size instances of the CLRP. Akca et al. [1] have introduced a mixed set partitioning / knapsack formulation by doing a Dantzig-Wolfe decomposition of the three-index formulation that is solved by means of a branch-and-price method. The pricing problem consists in finding elementary paths of minimum reduced cost under capacity constraints. The lower bounds obtained by their algorithm show a significant improvement with respect to those obtained by the algorithms based on the two-index vehicle-flow formulation. More recently, Baldacci et al. [4] have proposed a branch-and-cut-and-price algorithm. They apply two bounding procedures to compute a tight lower bound, followed by the optimal solution of a small number of multiple depot vehicle routing problems (MDVRP). They provide a strengthened version of the CC as well as clique inequalities for the set-partitioning problem. Their algorithm improves the lower bounds of the previous approaches and solves to optimality instances with up to 199 customers and 14 facilities.

The CLRP is known to be NP-hard, as it combines (and includes as particular cases) both the Capacitated Vehicle Routing Problem (CVRP) and the Capacitated Facility Location Problem (CFLP). Authors have thus focused their attention on the development of heuristic methods to find good quality solutions in reasonable computing times. Most of these heuristics are based on decomposition techniques that solve a location (design) and a routing (operational) sub-problem. Depending on whether the algorithm iterates between

the two subproblems, we distinguish between sequential algorithms [22] and iterative algorithms [13, 25, 16, 17]. Tuzun and Burke [24] decompose the problem into a location and a routing subproblem, but the location decisions at each iteration only consider the opening of new facilities or the swapping of two already open facilities, so the whole algorithm rapidly converges to a local optimum. Other heuristics include memetic algorithms [23] or Lagrangian heuristics [21].

The main contributions of this paper can be summarized as follows:

i. We introduce three new formulations based on vehicle flows and commodity flows, which are proven to dominate, in terms of the linear relaxation lower bound, the two-index vehicle-flow formulation of Belenguer et al. [6] at the expense of adding more variables.

ii. We derive two new families of multistar inequalities from the commodity-flow formulations and introduce separation algorithms for using them inside the vehicle-flow formulations.

iii. We introduce several new families of valid inequalities for the formulations introduced in this paper, and strengthen several of the existing ones.

iv. We introduce new, efficient separation algorithms for the inequalities used in our algorithms, which in many cases generalize those introduced by Belenguer et al. [6].

v. We perform a computational study comparing each of the formulations on a large number of instances and discuss their advantages and disadvantages.

The rest of the paper is organized as follows. In Section 2 we first describe the two-index formulation introduced by Belenguer et al. [6]. We then introduce the three new formulations based on vehicle flows and commodity flows. We prove that for the case of the comodity-flow formulations, some new classes of multistar inequalities are implied. In Section 3 we present both existing and new families of valid inequalities for the CLRP. In Section 4 we begin by introducing a general heuristic for generating cuts, and we then introduce the separation algorithms for each of the valid inequalities introduced in the paper. In Section 5 we describe the branch-and-cut algorithms used in our experiments by specifying the separation and branching strategies. In Section 6 we present a computational study performed after running our algorithms on several families of instances. This is followed by the conclusions in Section 7. To improve clarity, we provide in the Appendix the proofs of the lemmas and propositions introduced in Sections 2, 3 and 4.

2 Mathematical Formulations

In this section we first describe the two-index formulation introduced by Belenguer et al. [6] for the CLRP with a homogeneous fleet and symmetric costs. We then introduce three new formulations based on vehicle flows and two-commodity flows.

2.1 A two-index vehicle-flow formulation [6]

We first introduce the notation that we will use throughout the article and then present the formulation itself.

Let $G = (V, E)$ be an undirected graph, where $V = I \cup J$ and $E = \{\{v_i, v_j\} : v_i, v_j \in V\} \setminus I \times I$. For every subset $U \subseteq V$, we define $E(U) = \{\{u, w\} \in E : u, w \in U\}$, and $\delta(U) = \{\{u, w\} \in E : u \in U, w \notin U\}$. For every pair of disjoint subsets U and W, let also $(U : W) = \{\{u, w\} \in E : u \in U, w \in W\}$. With every edge $e \in \delta(I)$ are associated two binary variables: x_e equal to 1 iff edge e is used once, and y_e equal to 1 iff edge e is used twice. With every edge $e \in E(J)$ is associated a binary variable x_e equal to 1 iff edge e is used. For every facility $i \in I$, let z_i be a binary variable equal to 1 iff facility i is selected. For a given edge set $F \subseteq E$ we define $x(F) = \sum_{e \in F} x_e$ and $y(F) = \sum_{e \in F} y_e$ (if $F \subseteq \delta(I)$). For a given subset $S \subseteq J$ of customers, we define $d(S) = \sum_{j \in S} d_j$, and a constant $r(S) = \lceil d(S)/Q \rceil$ which is a lower bound on the number of vehicles required to satisfy the demand of customers in S. Finally, we define $\overline{S} = J \setminus S$. The CLRP can then be formulated as the following integer program.

$$\min \quad \sum_{i \in I} f_i z_i + \sum_{e \in E} c_e x_e + 2 \sum_{e \in \delta(I)} c_e y_e \tag{VF2}$$

subject to

$$x(\delta(j)) + 2y(I : \{j\}) = 2 \qquad j \in J \quad (1)$$

$$x(\delta(S)) + 2y(I : S) \geq 2r(S) \qquad S \subseteq J, |S| \geq 2 \quad (2)$$

$$x_{ij} + y_{ij} \leq z_i \qquad i \in I, j \in J \quad (3)$$

$$x(I : \{j\}) + y(I : \{j\}) \leq 1 \qquad j \in J \quad (4)$$

$$x((I \setminus \{i\}) \cup \overline{S} : S) + 2y(I \setminus \{i\} : S) \geq 2 \qquad i \in I, S \subseteq J, d(S) \geq b_i \quad (5)$$

$$x(\delta(S)) \geq 2(x(\{h\} : I') + x(\{j\} : I \setminus I')) \qquad S \subseteq J, |S| \geq 2, h, j \in S, I' \subset I \quad (6)$$

$$z_i \in \{0, 1\} \qquad i \in I \quad (7)$$

$$x_e \in \{0, 1\} \qquad e \in E \quad (8)$$

$$y_e \in \{0, 1\} \qquad e \in \delta(I). \quad (9)$$

Demand constraints (1) impose that every customer vertex be visited once and also act as flow conservation equations. Constraints (2) are the capacity cuts (CC) which play a dual role: they forbid tours disconnected from facilities as well as tours serving a demand larger than Q. Constraints (3) ensure that there is no outgoing flow from unselected facilities. Constraints (4) forbid single-customer routes to be linked to two different facilities. Constraints (5) are the facility capacity inequalities (FCI). They forbid the existence of a set of routes leaving from a given facility i and serving a demand higher than b_i. Constraints (6) are the path constraints (PC) that prevent the route of a vehicle from joining two different facilities. These constraints are not valid when $|S| = 1$ and they are thus complementary to constraints (4).

Unlike in traditional CVRP formulations, two sets of variables (x and y) are associated with the edges in $\delta(I)$. One can in fact check that if these variables are replaced with the

aggregated variables $\overline{x}_e = x_e + 2y_e$, single-customer routes linked to two different facilities can no longer be correctly eliminated as we do with constraints (4).

2.2 A three-index vehicle-flow formulation

Due to their large number of variables, three-index formulations for vehicle routing problems have limited practical interest. In these formulations, two indices represent a certain edge while the third index indicates which vehicle uses this edge. These formulations naturally provide tighter bounds than their two-index counterparts when augmented by all of the known valid inequalities. However, they also present a lot of symmetry that makes them of little use within a branch-and-bound framework. We introduce a three-index formulation which does not suffer from this issue. Indeed, we use the third index to specify the facility from which the edge is being visited. Symmetry is then not an issue because switching two facilities does not provide an alternate equivalent solution, either because of feasibility (facility capacities may not be the same) or costs (switching routes from one facility to another usually pruduces a change in either the routing costs or the fixed costs). Using the same notation as for the two-index vehicle-flow formulation, we define binary variables x^i_e equal to 1 iff edge e is used once by a vehicle being from facility $i \in I$ (naturally $x^i_{lj} = 0$ if $l, i \in I, l \neq i$). We also let y_{ij} be a binary variable equal to 1 iff edge $e = \{i, j\}$ is used twice (for single-customer routes) by a vehicle linked to facility i. We let u_{ij} be a binary variable equal to 1 iff customer j is served from facility i. Let us define the following notation. For an edge subset $F \subseteq E$ and a facility subset $H \subseteq I$ we let $x^H(F) = \sum_{i\in H} \sum_{e\in F} x^i_e$, and if $H = \{i\}$ is a singleton we let $x^i(F) = x^{\{i\}}(F)$. The formulation is the following,

$$\min \quad \sum_{i\in I} f_i z_i + \sum_{i\in I} \sum_{e\in E} c_e x^i_e + 2 \sum_{i\in I} \sum_{j\in J} c_{ij} y_{ij} \tag{VF3}$$

subject to

$$x^i(\delta(\{j\})) + 2y_{ij} = 2u_{ij} \qquad i \in I, j \in J \tag{10}$$

$$x^i(\delta(S)) + 2y(\{i\} : S) \geq \frac{2}{Q} \sum_{j\in S} d_j u_{ij} \qquad i \in I, S \subseteq J, |S| \geq 2 \tag{11}$$

$$\sum_{j\in J} d_j u_{ij} \leq b_i z_i \qquad i \in I \tag{12}$$

$$\sum_{i\in I} u_{ij} = 1 \qquad j \in J \tag{13}$$

$$x^i_{ij} + y_{ij} \leq u_{ij} \leq z_i \qquad i \in I, j \in J \tag{14}$$

$$z_i \in \{0, 1\} \qquad i \in I \tag{15}$$

$$x^i_e \in \{0, 1\} \qquad i \in I, e \in E \tag{16}$$

$$y_e \in \{0, 1\} \qquad e \in \delta(I) \tag{17}$$

$$u_{ij} \in \{0, 1\} \qquad i \in I, j \in J. \tag{18}$$

Constraints (10) are a disaggregated form of the degree equations (1), whereas constraints (11) are a disaggregated form of the capacity inequalities (2). Constraints (12) are the

facility capacity inequalities. Constraints (13) are the assignment constraints of customers to facilities. Constraints (14) link the assignment variables with the flow and location variables.

2.3 A two-index two-commodity flow formulation

Each facility node $i \in I$ is considered as a source of flow, to which we consider an additional sink node i'. Let us denote the set of sink facility nodes as I', and consider the augmented *undirected* graph $\overline{G} = (\overline{V}, \overline{E})$ with $\overline{V} = V \cup I'$ and $\overline{E} = E \cup \{e = \{i', j\} : i' \in I', j \in J\}$. A route starting and ending at a facility i in the original graph will be mapped to a flow in the new graph starting at i and arriving to i'. For this purpose, let us introduce the following set of continuous variables. For every edge $e = \{i, j\} \in \overline{E}$, we define an arc variable w_{ij} which denotes the amount of flow traversing edge e *if* e *is traversed from node* i *to node* j, and w_{ji} represents the remaining capacity on the vehicle traversing this edge. If the trip is done in the opposite direction the roles of w_{ij} and w_{ji} are reversed. To take into account the orientation defined by these new variables, we define for every set $U \subseteq V$, $w(\delta^+(U)) = \sum_{u \in U, v \notin U} w_{uv}$, $w(\delta^-(U)) = \sum_{u \in U, v \notin U} w_{vu}$. We keep variables y for single-customer trips, while variables w are only used for multiple-customer routes (i.e., routes serving two or more customers). The following set of constraints are thus valid for the CLRP

$$w(\delta^-(\{j\})) - w(\delta^+(\{j\})) + 2d_j y(I : \{j\}) = 2d_j \qquad j \in J \qquad (19)$$

$$w(\delta^+(\{i\})) + \sum_{j \in J} d_j y_{ij} \leq b_i z_i \qquad i \in I \qquad (20)$$

$$w(\delta^+(\{i'\})) = Qx(\delta(\{i\})) \qquad i' \in I' \qquad (21)$$

$$w_{ij} + w_{ji} = Qx_{ij} \qquad \{i, j\} \in \overline{E} \qquad (22)$$

$$w_{ij}, w_{ji} \geq 0 \qquad \{i, j\} \in \overline{E}. \qquad (23)$$

Now, vehicle capacities and facility capacities are implied by (19)-(23). A valid formulation for the CLRP is given by

$$\min \quad \sum_{i \in I} f_i z_i + \sum_{e \in E} c_e x_e + 2 \sum_{e \in \delta(I)} c_e y_e \qquad \text{(CF2)}$$

subject to (1), (3)-(4), (6)-(9), (19)-(23).

Baldacci et al. [3] proved that the following flow inequalities (FI) are valid for the two-index two-commodity flow formulation of the CVRP:

$$(Q - d_j) w_{ij} - d_j w_{ji} \geq 0 \qquad \{i, j\} \in \overline{E} \qquad (24)$$

$$(Q - d_i) w_{ji} - d_i w_{ij} \geq 0 \qquad \{i, j\} \in \overline{E}. \qquad (25)$$

It is straightforward to check that they also are for the CLRP. As stated by the following proposition, they also imply the following y-generalized large multistar inequalities (y-GLM),

Proposition 2.1. *The following y-generalized large multistar inequalities (y-GLM) are implied by formulation* (CF2) *when augmented with the flow inequalities (FI):*

$$x(\delta(S)) + 2\sum_{j\in S}\frac{d_j}{Q}y(I:\{j\}) \geq \frac{2}{Q}\left(d(S) + \sum_{\substack{h\in S\\ j\notin S}} d_j x_{hj}\right). \tag{26}$$

The generalized large multistar inequalities (GLM) which are valid for the CLRP differ from these inequalities in the coefficients d_j/Q multiplying the terms $y(I:\{j\})$ which are replaced by 1. Therefore, the y-GLM dominate the GLM.

2.4 A three-index two-commodity flow formulation

Let us consider the three-index vehicle-flow formulation (VF3). As for the previous formulation, we consider the augmented graph $\overline{G}$ and we use variables w^i_{hj}, w^i_{jh} for the flow traversing edge $\{h,j\}$ from facility i and for the remaining capacity in the vehicle, respectively. We keep variables y_{ij} for single-customer routes. For a facility $i \in I \cup I'$ and a node subset $U \subseteq \overline{V}$ we denote $w^i(\delta^+(U)) = \sum_{u\in U, v\notin U} w^i_{uv}$, $w^i(\delta^-(U)) = \sum_{u\in U, v\notin U} w_{vu}$. Formulation (VF3) can thus be augmented by adding these variables and the following set of constraints:

$$w^i(\delta^-(\{j\})) - w^i(\delta^+(\{j\})) + 2d_j y_{ij} = 2d_j u_{ij} \qquad i\in I, j\in J \tag{27}$$
$$w^i(\delta^+(\{i\})) + \sum_{j\in J} d_j y_{ij} \leq b_i z_i \qquad i\in I \tag{28}$$
$$w^i(\delta^+(\{i'\})) = Q x^i(\delta(\{i\})) \qquad i'\in I' \tag{29}$$
$$w^i_{hj} + w^i_{jh} = Q x^i_{hj} \qquad i\in I, \{h,j\}\in \overline{E} \tag{30}$$
$$w^i_{hj}, w^i_{jh} \geq 0 \qquad i\in I, \{h,j\}\in \overline{E}. \tag{31}$$

The new formulation for the CLRP is the following

$$\min \quad \sum_{i\in I} f_i z_i + \sum_{i\in I}\sum_{e\in E} c_e x^i_e + 2\sum_{i\in I}\sum_{j\in J} c_{ij} y_{ij} \tag{CF3}$$

subject to (10), (13)-(18), (27)-(31).

Note that the following disaggregated flow inequalities (DFI) are valid for this formulation

$$(Q-d_j)w^i_{hj} - d_j w^i_{jh} \geq 0 \qquad i\in I, \{h,j\}\in \overline{E} \tag{32}$$
$$(Q-d_h)w^i_{jh} - d_h w^i_{hj} \geq 0 \qquad i\in I, \{h,j\}\in \overline{E}. \tag{33}$$

As a consequence of the extra variables added with respect to the two-index two-commodity flow formulation, this one also implies the following y-location routing generalized large multistar inequalities (y-LRGLM),

Proposition 2.2. *The following y-location routing generalized large multistar inequalities (y-LRGLM) are implied by formulation* (CF3) *plus the disaggregated flow inequalities (DFI).*

$$x^{I\setminus H}(\delta(S)) + 2\sum_{j\in S}\frac{d_j}{Q}y(I\setminus H:\{j\}) \geq \frac{2}{Q}\left(\sum_{i\in I\setminus H}\sum_{j\in S}d_j u_{ij} + \sum_{\substack{h\in S\\ j\notin S}} d_j x_{hj}^{I\setminus H}\right). \tag{34}$$

Remark Note that the particular case $H = \emptyset$ corresponds to the y-GLM (26).

3 Valid Inequalities

In this section we consider several families of valid inequalities that can be used to strengthen the linear relaxation of the previous formulations. We first describe known inequalities and then introduce new families of valid inequalities.

3.1 Known valid inequalities

In this subsection we describe valid inequalities that are already known for the CLRP. These include constraints for the CVRP such as framed capacity inequalities (FrCI), strengthened comb inequalities (SCI), multistar inequalities (MSI), hypotour inequalities (HYP), y-capacity cuts (y-CC), strengthened facility capacity inequalities (SFCI), co-circuit constraints (CoCC) and facility degree constraints (FDC). For details on each of these inequalities we refer to Lysgaard et al. [18] and to Belenguer et al. [6].

3.1.1 Inequalities for the CVRP

If nodes in I are contracted into a single node, the resulting problem can be seen as a CVRP instance. If a cut valid for the CVRP is such that the coefficients of the edges joining the depot to customers do not vary with the depot (as the distance, for instance), this cut remains valid for the CLRP by considering this contracted graph. This is the case for all of the known valid inequalities, in particular, strengthened comb inequalities, multistar inequalities, generalized large multistar inequalities, framed capacity inequalities and hypotour inequalities [18]. We add them all except for the generalized large multistar inequalities which are replaced by the y-GLM (26).

3.1.2 y-Capacity cuts [6]

Let us consider constraints (2) for a given customer set S. Additionally, suppose that we are given a customer subset S' satisfying $r(S\setminus S') = r(S)$. The following constraint, called y-capacity cut or simply y-CC, is valid for the CLRP and dominates (2):

$$x(\delta(S)) + 2y(I : S\setminus S') \geq 2r(S). \tag{35}$$

For the proof that these constraints are valid, we refer to Belenguer et al. [6].

3.1.3 Strengthened facility capacity inequalities [6]

For a given facility set I', let us denote $b(I') = \sum_{i \in I'} b_i$. Belenguer et al. [6] proposed the following two strengthenings for inequalities (5). Let $S \subseteq J$ and $i \in I$ be as in inequalities (5). Let $I' \subset I$ be a subset of facilities such that $i \in I'$. If a subset $S' \subset S$ is such that $d(S \setminus S') > b(I')$ then the following strengthened facility capacity inequality (SFCI) is valid for the CLRP:

$$x((I \setminus I') \cup \bar{S} : S) + 2y(I \setminus I' : S \setminus S') \geq 2. \tag{36}$$

Let $r(S, I') = \lceil (d(S) - b_{I'})/Q \rceil$ be a lower bound on the number of vehicles needed to serve the demand of customers in S from facilities other that those in I'. Note that although $r(\cdot)$ and $r(\cdot, \cdot)$ represent different quantities, the overloaded notation satisfies $r(S, \emptyset) = r(S)$ for every $S \subseteq J$. The following inequality is valid for the CLRP:

$$x((I \setminus I') \cup \bar{S} : S) + 2y(I \setminus I' : S) \geq 2r(S, I' \setminus \{i\}) + 2z_i(r(S, I') - r(S, I' \setminus \{i\})). \tag{37}$$

We call these inequalities the effective SFCI (ESFCI). For the validity of these inequalities we refer again to Belenguer et al. [6].

3.1.4 Co-circuit constraints [6]

The co-circuit constraints (CoCC) state that the graph resulting from the deletion of the y variables must still have an even number of edges. They can be written as

$$x(\delta(S) \setminus F) \geq x(F) - |F| + 1 \tag{38}$$

for $S \subseteq J$, $F \subseteq \delta(S)$ and $|F|$ odd.

3.1.5 Facility degree constraints [6]

The facility degree constraints (FDC) are valid under the assumption that the triangle inequality holds for the edge distances. It states the sub-optimality of solutions in which two or more vehicles serve a given set of customers if these customers can be served by fewer vehicles (thus saving travel time). For single-customer routes they can be written as

$$y(i : S) \leq z_i \tag{39}$$

$\forall S \subseteq J$ such that $d_h + d_j \leq Q, \forall h \neq j \in S, \forall i \in I$. For general routes, they can be written as

$$2y(i : S) + x(i : S) + x(E(S)) \leq 2z_i + |S| - 1 \tag{40}$$

$\forall i \in I, \forall S \subseteq J, r(S) = 1$.

3.2 New valid inequalities

In this subsection we introduce new families of valid inequalities for the CLRP. These include strengthened versions of the SFCI, ESFCI, location-routing comb inequalities (LRCOMB), location-routing generalized large multistar inequalities (LRGLM) and flow-assignment inequalities (FAI), all of which are valid for the two-index formulations and by extension for the three-index formulations as well. Moreover, we strengthen some of these inequalities for the case of the three-index formulations, and add some novel classes of inequalities that cannot be derived from the former.

3.2.1 Flow-assignment inequalities

It is easy to check that the following inequalities are valid for the two-index and three index commodity-flow formulations, respectively:

$$w_{ij} + w_{ji} \leq Q \qquad \{i,j\} \in \overline{E} \tag{41}$$
$$w^l_{ij} + w^l_{ji} \leq Q \qquad l \in I, \{i,j\} \in \overline{E}. \tag{42}$$

However, they can be strengthened as a consequence of the following two observations. First, for every edge $e = \{i,j\} \in E$, at least one node i or j belongs to J. For that node, say j, it cannot happen at the same time that edge $\{i,j\}$ is used by a vehicle serving two or more customers and j is served by a single-cutomer route. Thus, the following flow-assignment inequalities (FAI) are valid for the CLRP:

$$x_{ij} + y(I:\{j\}) \leq 1 \qquad j \in J, \{i,j\} \in E. \tag{43}$$
$$w_{ij} + w_{ji} + Qy(I:\{j\}) \leq Q \qquad j \in J, \{i,j\} \in \overline{E}. \tag{44}$$
$$x^l_{ij} + y_{lj} \leq u_{lj} \qquad l \in I, j \in J, \{i,j\} \in E. \tag{45}$$
$$w^l_{ij} + w^l_{ji} + Qy_{lj} \leq Qu_{lj} \qquad l \in I, j \in J, \{i,j\} \in \overline{E}. \tag{46}$$

In the case of the three-index formulations, constraints (45)-(46) impose a strong relationship between the flow variables and the customers assignments. Indeed, if a customer is not assigned to a given facility, then all flow variables associated to the corresponding facility and linked to that customer are automatically set to 0.

3.2.2 Disaggregated co-circuit constraints

The co-circuit constraints (38) ensure that an even number of edges will traverse a given customer subset $S \subseteq J$. This is in particular valid when restricted to the edges used by some facility. Thus, for the particular case of the three-index formulations the following disaggregated co-circuit constraints (DCoCC) are valid for the CLRP:

$$x^i(\delta(S) \setminus F) \geq x^i(F) - |F| + 1 \qquad i \in I, S \subseteq J, F \subset \delta(S), |F| \text{ odd}. \tag{47}$$

Proposition 3.1. *Constraints* (47) *are valid for the CLRP.*

3.2.3 Disaggregated facility degree constraints

Using the same reasoning as for the CoCC, the facility degree constraints (40) also have their disaggregated counterpart. Indeed, if distances satisfy the triangle inequality, the following inequalities are valid for the three-index formulations of the CLRP:

$$x(i:S) + 2y(i:S) + x^i(E(S)) \leq \sum_{j \in S} u_{ij} + z_i \qquad i \in I, S \subseteq J, d(S) \leq Q. \tag{48}$$

Proposition 3.2. *Constraints* (48) *are valid for the CLRP.*

3.2.4 Strengthened facility capacity inequalities

Let us consider inequalities (36) for given $S \subseteq J$ and $I' \subseteq I$. If $S' \subset S$ is such that $r(S \setminus S', I') = r(S, I')$ let us consider the following inequality:

$$x((I \setminus I') \cup \overline{S} : S) + 2y(I \setminus I' : S \setminus S') \geq 2r(S, I'). \tag{49}$$

Proposition 3.3. *Constraints* (49) *are valid for the CLRP.*

As these constraints dominate (36), we will now refer to these inequalities as SFCI. These constraints are valid for all the formulations studied in this paper. However, for the three-index case they can be strengthened to the following constraints:

$$x^{I \setminus I'}(\delta(S)) + 2y(I \setminus I' : S \setminus S') \geq 2r(S, I'). \tag{50}$$

3.2.5 Effective strengthened facility capacity inequalities

Let us consider constraints (37) for given S, I' and $i \in I'$. Suppose that $S' \subseteq S$ is such that $r(S \setminus S', I') = r(S, I')$ and $r(S \setminus S', I' \setminus \{i\}) = r(S, I' \setminus \{i\})$. Then, the following inequality is valid for the CLRP and dominates (37):

$$x((I \setminus I') \cup \overline{S} : S) + 2y(I \setminus I' : S \setminus S') \geq 2r(S, I' \setminus \{i\}) + 2z_i(r(S, I') - r(S, I' \setminus \{i\})). \tag{51}$$

As this inequality dominates (37), we will refer to it as the ESFCI.

Proposition 3.4. *Constraints* (51) *are valid for the CLRP.*

Just as with the SFCI, for the three-index case these inequalities can be strengthened to the following set of inequalities

$$x^{I \setminus I'}(\delta(S)) + 2y(I \setminus I' : S \setminus S') \geq 2r(S, I' \setminus \{i\}) + 2z_i(r(S, I') - r(S, I' \setminus \{i\})). \tag{52}$$

Remark Constraints SFCI and ESFCI do not dominate each other. However, in practice, we have verified that for fractional values of the z variables the ESFCI have a more significant impact on the lower bound than the SFCI. Conversely, when location variables are fixed to either 0 or 1, constraints SFCI start playing an important role. Because of that, at every node of the branching tree we separate constraints ESFCI for facilities i such that $0 < z_i \leq 0.85$ and constraints SFCI for every i such that $0.75 < z_i \leq 1$. The role of every cut is complementary: constraints ESFCI help to stregthen the lower bound and hopefully to prune nodes close to the root, while constraints SFCI start dominating the ESFCI deeper in the tree.

3.2.6 Location-routing comb inequalities

Comb inequalities were developed by Chvátal [8] for the symmetric traveling salesman problem (STSP) and have since then received considerable attention in the literature [11, 14, 18]. In particular, stronger versions have been proposed for the CVRP that take advantage of the vehicle capacities. In what follows we develop a new family of inequalities that are shown to be valid for the CLRP and include some of the earlier inequalities as special cases. Let sets $H \subseteq V$ (the *handle*), $\Pi = (T_j^1)_{j=1}^{s_1} \cup (T_j^2)_{j=1}^{s_2} \subseteq \mathcal{P}(V)$ (the *teeth*) be such that

i. $|H \cap T| \geq 1 \quad T \in \Pi$

ii. $|T \setminus H| \geq 1 \quad T \in \Pi$

iii. $|T \cap U| = 0 \quad T, U \in \Pi$

iv. $|H \cap I| = 0$

v. $|T_j^1 \cap I| \geq 1 \quad 1 \leq j \leq s_1$

vi. $|T_j^2 \cap I| = 0 \quad 1 \leq j \leq s_2$

For notational simplicity, for every k, j we denote $S_j^k = T_j^k \cap J$. If $k = 1$, we also denote $I_j = T_j^1 \cap I$. Let $s_1' < s_1$ and suppose that for each $j \in \{1, \ldots, s_1'\}$ we also distinguish a special facility $i_j \in I_j$ that we call *effective*. For every set $U \subseteq V = I \cup J$ let us denote

$$\overline{x}(E(U)) = \begin{cases} x(E(U)) & \text{if } U \cap I = \emptyset \\ x(E(U \setminus I)) + x(U \cap I : U \setminus I) + 2y(U \cap I : U \setminus I) & \text{if } U \cap I \neq \emptyset. \end{cases}$$

Let $\alpha\overline{x} = \overline{x}(E(H)) + \sum_{k=1}^{2} \sum_{j=1}^{s_k} \overline{x}(E(T_j^k))$. Define the following constants:

$$\widehat{r}(H, T_j^k) = \begin{cases} r(S_j^1, I_j \setminus \{i_j\}) + r(S_j^1 \setminus H, I_j \setminus \{i_j\}) + r(S_j^1 \cap H) & \text{if } k = 1, 1 \leq j \leq s_1' \\ r(S_j^1, I_j) + r(S_j^1 \setminus H, I_j) + r(S_j^1 \cap H) & \text{if } k = 1, s_1' < j \leq s_1 \\ r(S_j^2) + r(S_j^2 \setminus H) + r(S_j^2 \cap H) & \text{if } k = 2, 1 \leq j \leq s_2 \end{cases} \quad (53)$$

$$\Lambda(H, T_j^1) = r(S_j^1, I_j \setminus \{i_j\}) + r(S_j^1 \setminus H, I_j \setminus \{i_j\}) - r(S_j^1, I_j) - r(S_j^1 \setminus H, I_j) \quad 1 \leq j \leq s_1 \quad (54)$$

$$\widehat{r}(H, \Pi) = \sum_{k=1,2} \sum_{1 \leq j \leq s_k} \widehat{r}(H, T_j^k). \quad (55)$$

If $\Lambda(H, T_j^1)$ is *even* for every $1 \leq j \leq s_1'$ and $\widehat{r}(H, \Pi)$ is *odd*, the associated location-routing comb inequality (LRCOMB) is

$$\alpha\overline{x} - \tfrac{1}{2}\Big[\sum_{1 \leq j \leq s_1} (x(I_j : J) + 2y(I_j : J)) + \sum_{1 \leq j \leq s_1'} z_{i_j} \Lambda(H, T_j^1) \Big] \leq |H| + \sum_{k=1}^{2} \sum_{j=1}^{t} |S_j^k| - \left\lceil \tfrac{1}{2} \widehat{r}(H, \Pi) \right\rceil. \quad (56)$$

Proposition 3.5. *The location-routing comb inequality* (56) *is valid for the CLRP.*

Remark 1. For the sake of clarity, we have assumed that $s_1, s_2 > 0$. Indeed, it is possible to omit this assumption and obtain the associated LRCOMB as a consequence.

Remark 2. The interest of considering $s_1' < s_1$ relies on the fact that we can relax the condition $\Lambda(H, T_j^1)$ *is even* for $s_1' < j \leq s_1$. This case becomes specially interesting when $z_j \sim 1$ for $s_1' < j \leq s_1$ because in such a case the strength of the comb inequality remains almost the same.

3.2.7 Location-routing generalized large multistar inequalities

We now introduce a new class of location-routing generalized large multistar inequalities that are valid for the two-index vehicle-flow formulation and that cannot be derived from inequalities (34). For given $I' \subset I, S \subseteq J$, and $j \notin S$, define $\eta(I', S, j) = x(S : j) + 1/2x(I' : \{j\}) + y(I' : \{j\})$. The following Location-routing generalized large multistar inequality (LRGLM) is valid for the two-index vehicle-flow formulation:

$$x((I - I') \cup \overline{S} : S) + 2y(I - I' : S) \geq \frac{2}{Q} \left(d(S) - b(I') + \sum_{j \notin S} d_j \eta(I', S, j) \right). \quad (57)$$

The validity of constraints (57) is a consequence of the following lemma and proposition.

Lemma 3.6. *Let $I' \subset I$, $S \subseteq J$. Let $W_{I'}$ be the set of customers that are served from facilities in I', and $T \subseteq \overline{S} \cap W_{I'}$. Then $x(E(S)) + 1/2x(I' : S) + y(I' : S) \leq |S| - \frac{1}{Q}(d(S \cup T) - b(I'))$.*

Proposition 3.7. *Constraint* (57) *is valid for the CLRP.*

Remark A stronger valid inequality can be obtained by replacing the right-hand side of constraint (57) by

$$\frac{2}{Q} \left(d(S) - \sum_{i \in I'} b_i z_i + \sum_{j \notin S} d_j \eta(I', S, j) \right). \quad (58)$$

3.2.8 Lifted cover inequalities

Lifted cover inequalities (LCI) can be useful when facilities have heterogeneous capacities. In such a case, the valid knapsack inequality $\sum_{i \in I} b_i z_i \geq d(J)$ can be used in order to derive LCI. For details on LCI we refer to Gu et al. [12].

4 Separation Algorithms

In this section we describe the separation algorithms that we use to identify violated valid inequalities from the families introduced in Section 3. We begin by introducing a general cut lifting heuristic that takes advantage of the particular underlying structure of some inequalities, decomposing the separation problem into two easier subproblems that are solved sequentially. Then, we present the different separation algorithms for the separation of the inequalities presented in the paper. They include some exact separation algorithms based on maximum-flow computations as well as connected components or shrinking heuristics. We make use of the CONCORDE Library [9] to solve the maximum-flow problems as well as the connected components problems, and the COMBO algorithm [19] for solving 0-1 knapsack problems.

4.1 A cut lifting heuristic

In this section we describe a general separation algorithm that takes advantage of the special structure of some families of valid inequalities. Let us consider a polytope $\mathcal{X} = \{x \in \mathbb{R}_n, Ax \leq b\}$ and denote by $\mathcal{Y} = conv(\mathcal{X} \cap \mathbb{Z}_n)$ the convex hull of the integer points of $\mathcal{X}$. Given a function $f : \mathbb{R}_n \to \mathbb{R}$ and a scalar $g \in \mathbb{R}$, we say that the tuple (f, g) is a valid inequality for $\mathcal{Y}$ if $f(x) \leq g$ for every $x \in \mathcal{Y}$. Given two functions $f : \mathbb{R}_n \to \mathbb{R}$ and $h : \mathbb{R}_n \to \mathbb{R}$ let us denote by $[f + h]$ the function $[f + h](x) = f(x) + h(x)$. Suppose that we are given a family of valid inequalities for $\mathcal{Y}$, $\mathcal{F} = \{([\alpha_j + \beta_{jk}], \gamma_j) : j = 1, \ldots \mathcal{J}, k = 1, \ldots, \mathcal{K}_j\}$ with $\beta_{jk}(\cdot) \geq 0$ for all j, k. Suppose that the family $\mathcal{F}_1 = \{(\alpha_j, \gamma_j) : j = 1, \ldots, \mathcal{J}\}$ is easy to separate, in the sense that for any $\epsilon > 0$ and $x \in \mathcal{X}$ the decision problem

$$\exists j \in \{1, \ldots, \mathcal{J}\} \text{ such that } \alpha_j(x) > \gamma_j - \epsilon \tag{P_1}$$

is easy to solve. Suppose that for given $j \in \{1, \ldots, \mathcal{J}\}$ and $x \in \mathcal{X}$, the problem

$$\begin{aligned} \max_k \quad & f(k) = \beta_{jk}(x) \\ \text{s.t.} \quad & k \in \mathcal{K}_j \end{aligned} \tag{P_2}$$

is easy to solve also, or that a good lower bound can be computed efficiently. Thus, given $x \in \mathcal{X}$, the following heuristic aims to find a valid cut $([\alpha_j + \beta_{jk}], \gamma_j) \in \mathcal{F}$ that is violated by x:

i. Fix $\epsilon > 0$ and use separation procedures for problem (P_1) in order to find one or more j's such that $\alpha_j(x) > \gamma_j - \epsilon$. We say that we find an ϵ-$\mathcal{F}_1$ cut.

ii. For every j found in (i) solve problem (P_2), obtaining k. If $\alpha_j(x) + \beta_{jk}(x) > \gamma_j$ then a violated inequality has been identified.

This procedure, although not exact, decomposes the problem into two easier subproblems and, as we will see later, can take advantage of known separation algorithms for related families of inequalities. We will see that problem (P_2) usually corresponds to solving a 0-1 knapsack problem. This problem is weakly NP-hard and efficient exact algorithms have been proposed. We have chosen to use the COMBO algorithm [19] that stands as the state-of-the-art solver for the 0-1 knapsack problem.

4.2 CVRP Inequalities

For the CVRP inequalities we make use of the separation algorithms developed by Lysgaard et al. [18] and which are available on the following website `http://www.hha.dk/~lys`.

4.3 y-Capacity constraints

We use the cut lifting heuristic described in Section 4.1 to exploit the well-known separation algorithms for the capacity constraints of the CVRP. In fact, problem (P_1) corresponds to the separation of the CC. Suppose that a set S has been found that solves the ϵ-CC separation problem. Problem (P_2) then aims to find a subset $S' \subseteq S$ such that the quantity $y(I : S')$ is

maximum while respecting the constraint $r(S \setminus S') = r(S)$. This problem can be written as the following 0-1 knapsack problem:

$$\begin{aligned} \max_{\mu} \quad & \sum_{j \in S} \mu_j y(I : j) \\ \text{s.t.} \quad & \sum_{j \in S} d_j \mu_j \leq \begin{cases} d(S) - Q(r(S) - 1) - 1 & \text{if } d(S) \not\equiv 0 \pmod{Q} \\ 0 & \text{otherwise} \end{cases} \\ & \mu \in \{0, 1\}^{|S|}. \end{aligned}$$

In our implementation, we have modified the code of Lysgaard et al. [18] to find ϵ-CC. The 0-1 knapsack problem is solved to optimality using the COMBO algorithm.

4.4 Strengthened facility capacity inequalities

We introduce separation algorithms for the separation of the SFCI (49). Note that as for the three-index case the inequalities (50) dominate (49), so the separation algorithms for the latter can in fact be safely used as heuristics. The separation for SFCI constraints (49) is done in three stages. First, we obtain candidate sets S and facilities $i \in I$ by solving the separation problem for the particular case of $|I'| = 1, |S'| = 0$. We refer to these specific constraints as the Basic FCI (BFCI). Note that these constraints are enough to ensure the feasibility of solutions. For each candidate sets S and $I' = \{i\}$, we use a greedy heuristic to enlarge the set I', and at every iteration in which I' is enlarged, we compute the set $S' \subset S$ that maximizes the quantity $y(I', S')$ and such that $r(S, I') = r(S \setminus S', I')$. This last problem corresponds to a 0-1 knapsack problem with item sizes $(d_j)_{j \in S}$, weights $(y(I', \{j\}))_{j \in S}$ and knapsack capacity of either $d(S) - b(I') - Q(r(S, I') - 1) - 1$ if $d(S) - b(I') \neq 0 \pmod{Q}$ or 0 otherwise. This procedure is an application of the cut lifting heuristic described in the subsection above, in which the subproblem corresponds to the described knapsack problem. We now describe the separation routine for generating the candidate sets S and $\{i\}$. We have implemented a safe shrinking routine, a connected component heuristic and an exact routine for the fractional case based on a series of min-cut computations, all of which are applied in the following order:

i. Start applying the shrinking routine. Every time that two customers are chosen for shrinking, the shrinking heuristic is applied to these customers.

ii. If the shrinking process is completed and the shrinking routine is not able to find a violated BFCI we run a connected component heuristic over the connected components of the shrunk graph.

iii. If none of the above procedures is able to find a violated BFCI we solve a polynomial number of min-cut problems over the shrunk graph.

We now present in detail the safe shrinking routine as well as each of the heuristic procedures mentioned above.

4.4.1 A safe shrinking routine

In what follows we denote by ω^*, ϕ^* the weight functions obtained from x^* and y^*, respectively, after successive contractions, and we keep x^*, y^* for the weights in the original unshrunk graph. We denote by d^* the aggregated demands of super-customers as well, whose set we denote by J_S. A super-customer comprises the set of all customers that have been shrunk to the same super-node. We will show that it is safe to shrink two customers $h, j \in J_S$ whenever

i. $d^*_h + d^*_j \leq Q$ and

ii. $[\omega^*_{hj} \geq 1]$ or $[\phi^*_{ih} \geq 1$ and $\phi^*_{ij} \geq 1]$.

Let us start by fixing a facility i. We will first show that for the separation of a BFCI using facility i it is safe to shrink any pair of nodes h and j in J_S satisfying only condition ii. If this is the case and $[\phi^*_{ih} \geq 1$ and $\phi^*_{ij} \geq 1]$, the new weights for the shrunk node $\{h, j\}$ are

- $\omega^*_{\{h,j\}v} = 0$ for all $v \in I \cup (J_S \setminus \{h, j\})$
- $\phi^*_{\{h,j\}v} = \begin{cases} 1 & \text{if } v = i \\ 0 & \text{otherwise} \end{cases}$

Otherwise (i.e., if $\omega^*_{hj} \geq 1$), the new weights are recalculated using the usual rule, as follows:

- $\omega^*_{\{h,j\}v} = \omega^*_{hv} + \omega^*_{jv}$ for all $v \in I \cup (J_S \setminus \{h, j\})$.

Remark For every super-customer h in the shrunk graph it is true that $\omega^*(\delta(h)) + 2\phi^*(I : h) = 2$.

Lemma 4.1. *For fixed $i \in I$, it is safe to shrink nodes $h, j \in J_S$ such that $\omega^*_{hj} \geq 1$ or $[\phi^*_{ih} \geq 1$ and $\phi^*_{ij} \geq 1]$.*

Remark If $\phi^*_{ih} = 1$ and $\phi^*_{ij} = 1$ it is not true that the shrinking of h and j is safe when considering a BFCI using a different facility, say l. In fact, in such a case, the last inequality in the proof above will be $\sigma_l(T) - \sigma_l(S) \leq 2 - 2\omega^*_{hj} - (\omega^*_{lh} + 2\phi^*_{lh})$ which is equal to 2. The next lemma proves, however, that in this case and whenever $d^*_j \leq Q$ and $d^*_h \leq Q$, h and j can be safely omitted from any BFCI containing facility l.

Lemma 4.2. *Let $h \in J_S$ be such that $\phi^*_{ih} = 1$ and $d^*_h \leq Q$. It is safe to omit node h from any BFCI containing a facility $l \neq i$.*

The following corollary follows as a consequence of Lemmas 4.1 and 4.2.

Corollary 4.3. *It is safe to shrink customers h, j such that*

i. $d^*_h + d^*_j \leq Q$ and

ii. $\omega^*_{hj} \geq 1$ or $\phi^*_{ih} = \phi^*_{ij} = 1$ for some $i \in I$.

4.4.2 Shrinking heuristic

During the execution of the shrinking routine, we can check at every iteration of the algorithm if two given super-customers $h, j \in J_S$ violate a BFCI, i.e., if $\omega^*_{hj} + \frac{1}{2}\omega^*(i : \{h, j\}) + \phi^*(i : \{h, j\}) > 2 - r(\{h, j\}, \{i\})$ for some $i \in I$. If this is the case, a violated inequality is obtained. Otherwise, we continue shrinking.

4.4.3 Connected component heuristic

Given a family of weights $(x'_e)_{e\in E}$, let $G_{x'} = (V, E_{x'})$ be the graph induced by the edges of E with strictly positive weights x'_e. The connected component heuristic works under the principle that if a violated BFCI exists associated to a facility i, then there is one contained in one of the connected components of the graph $G_{x'}$ (see Lemma 4.4 below), with x' defined as follows:

$$x'_e = \begin{cases} x^*_e + 2y^*_e & \text{if } e \in \delta(I) \\ x^*_e & \text{otherwise.} \end{cases} \tag{59}$$

Lemma 4.4. *Let $i \in I$ be a facility, and let $S \subseteq J$ be a disconnected (with respect to x') customer subset. Without loss of generality suppose that S_1, S_2 is a partition of S such that both S_1 and S_2 satisfy the CC constraints* (2)*. Then, if (i, S) defines a violated BFCI, (i, S_1) or (i, S_2) define another BFCI cut with a stronger violation as measured by the difference between the right-hand side and left-hand side of constraint* (49) *evaluated in vectors (x^*, y^*).*

The description of the algorithm is as follows. We start by looking at the connected components of the graph $G_{x'}$ (we make sure that connected components of $G_{x'}$ will satisfy constraints CC during their separation). Let S_k, I_k be the customers and facilities belonging to the k^{th} connected component, for $k = 1, \ldots, \Gamma$. Then, for every k and for every $i \in I_k$ we set $S^i_k = S_k \setminus \{h : y^*_{lh} = 1, l \neq i\}$, and we iteratively check whether the pair (i, S^i_k) violates a BFCI or not. If it does, we have identified a violated inequality. Otherwise we choose $j \in S^i_k$ such that the quantity $x^*(S^i_k \setminus \{j\} : j) + 1/2x^*_{ij} + y^*_{ij} + r(S^i_k, \{i\}) - r(S^i_k \setminus \{j\}, \{i\})$ is minimum and we remove it from S^i_k, repeating this procedure as long as we do not find a violated cut and $S^i_k \neq \emptyset$.

4.4.4 Exact separation of fractional BFCI's

The problem of finding a violated fractional BFCI can be formulated as the solution of $|I||J|$ minimum $\{s, t\}$-cut problems as follows: fix some $i \in I$ and $j \in J$. Consider the graph $G'(V', A')$ produced from $G(V, A)$ after deleting node i and contracting nodes in $I \setminus \{i\}$ in a single super node s. Define the weight of the new edges $\{h, k\} \in A', h < k$ as

$$x'_{hk} = \begin{cases} \sum_{l\in I\setminus\{i\}}(x^*_{lk} + 2y^*_{lk}) - 2d_k/Q & \text{if } h = s \\ x^*_{hk} & \text{if } h \neq s. \end{cases}$$

Although there are negative weight edges, the problem of finding a minimum $\{s, j\}$-cut can still be solved in polynomial time as pointed out by McCormick et al. [20]. Obviously there exists an $s - j$ cut in the modified graph of capacity smaller than $-2b_i/Q$ for some $i \in I, j \in J$ iff there exists a violated fractional BFCI.

4.5 Effective strengthened facility capacity inequalities

Analougously to the SFCI, note that for the three-index case, the separation procedures for constraints (51) can be safely used as heuristics for separating constraints (52). The separation of the ESFCI (51) is done in a completely analogous way to the SFCI. In a first stage, we get candidate sets S and $I' = \{i\}$ by solving the separation algorithms for the EBFCI, that correspond to the particular case of ESFCI when $|I'| = 1, |S'| = 0$. For every candidate pair S, $I' = \{i\}$, we enlarge the set I' in a greedy way and after every extension we compute the set S' that maximizes the quantity $y(I', S')$ and such that $r(S, I') = r(S \setminus S', I'), r(S, I' \setminus \{i\}) = r(S \setminus S', I' \setminus \{i\})$. Again, this problem corresponds to a 0-1 knapsack problem and is a direct application of the cut lifting heuristic. The separation algorithms for the EBFCI are completely analogous to those used for the BFCI and for the sake of brevity we will omit the remaining details.

Remark Note that the safe shrinking result for the BFCI is also safe for the separation of the EBFCI. In fact, one can take advantage of this observation and shrink the graph just once.

4.6 Co-circuit constraints

We have implemented two heuristic procedures and an exact algorithm based on the computation of a minimum-cut tree. Note that for a given set S, the computation of the set F such that the left-hand side of constraint (38) is minimum can be done in linear time by defining $F = \{e \in \delta(S) : x_e \leq 1/2\}$. If $|F|$ is even, then we either add to or remove from F the edge in $\delta(S)$ that minimizes the increase of the left-hand side of (38). The first heuristics checks, for every customer $j \in S$ if the corresponding co-circuit constraint is violated for $S = \{j\}$. If we de not find any cut, we compute the blocks (2-connected components) of the graph $G_{1/2}$ induced by the edges $\{e \in E : \epsilon \leq x_e \leq 1 - \epsilon\}$ and whose weights are taken as $w_e = \min\{x_e, 1 - x_e\}$. For this, we have taken $\epsilon = 10^{-5}$. If this procedure also fails, then we solve the separation of the blossom inequalities by computing a minimum-cut tree on graph $G_{1/2}$ using the Gomory-Hu algorithm [10]. We take as candidate handles the cuts induced by the edges of this tree. The first heuristic and the exact separation are done as suggested by Belenguer et al. [6], while the idea of considering the blocks of the graph as candidate handles has been successfully implemented into the separation of blossom inequalities in the CONCORDE solver for the TSP [2]. The separation of the DCoCC is done in a completely analogous way to the CoCC and, for the sake of brevity, we omit the details.

4.7 Facility degree constraints

Constraints (39) are not dynamically added but rather included at the beginning of the algorithm for the set J_Q built as follows. Let $J_Q = \emptyset$ and let V be the set containing the customers in J sorted by non-decreasing demands. Pick the first customer $v \in V$ and check if $d_v + d_j \leq Q$ for all $j \in J_Q$. If that is the case, then add v to J_Q, remove v from V and continue. If not, then stop. This way of constructing the set J_Q generalizes the approach of Belenguer et al. [6] in which J_Q is restricted to contain customers whose demands are $\leq Q/2$ by adding the possibility of adding one more customer.

For the separation of constraints (40) (respectively (48)) we have implemented two heuristics. First, we fix $i \in I$ and set $S = \emptyset$. Iteratively we enlarge set S by adding the customer $j \notin S$ that maximizes the quantity $2y^*_{ij} + x^*_{ij} + x^*(S : j)$ (respectively $2y^*_{ij} + x^{i*}_{ij} + x^{i*}(S : j) - u_{ij}$). The algorithm terminates if either $d(S) \geq Q$ or a violated constraint (40) (respectively (48)) has been detected. If this fails, we check the violation for every y-CC generated so far during the algorithm such that $d(S) \leq Q$, just as done by Belenguer et al. [6].

4.8 Path constraints

To separate constraints (6) we first shrink the graph using a safe shrinking routine. Once the graph has been completely shrunk we find (if one exists) a violated constraint (6) using a greedy search heuristic or, in case the first fails, a series of min-cut computations, which yields an exact separation algorithm.

4.8.1 A safe shrinking routine

Using the same notation as before, let J_S be the customer set containig the shrunk customers, and let ω^*, ϕ^* be the edge weights in the shrunk graph. The following proposition gives a safe condition for shrinking customer nodes during the separation of constraints (6).

Proposition 4.5. *For the path constraints* (6) *it is safe to shrink customers* $u, v \in J_S$ *such that* $\omega^*_{uv} \geq 1$ *and* $\omega^*(I : u) = \omega^*(I : v) = 0$.

4.8.2 Greedy search heuristic

Because solving a max-flow problem can be time-consuming, we have implemented a greedy search heuristic that aims to find all the chains of length two or three in the shrunk graph. We simply check for every pair or triplet of customers (in the shrunk graph) whether they define or not a violated PC.

4.8.3 Exact separation

The problem of finding a violated inequality (6) can be solved in polynomial time [6] in the following way. For fixed $h, j \in J$ contract in the usual way (i.e., by recalculating the edge weights properly) in the underlying graph $G(V, E)$ nodes in I in a super-node s and nodes h, j in a super-node t. Let us call $J' = (J - \{h, j\}) \cup \{t\}$. In this new graph, let us consider the following weight function:

$$x'_{uv} = \begin{cases} x^*(I : v) & u = s, v \in J' \setminus \{t\} \\ x^*(I : \{h, j\}) & u = s, v = t \\ x^*_{uv} & u, v \in J' \setminus \{t\} \\ x^*_{uh} + x^*_{uj} & u \in J' \setminus \{t\}, v = t. \end{cases}$$

Let us find a cut of minimum weight between s and t in this graph. Let S be the side of this minimum cut that contains t. Then, define $I_1 = \emptyset$. For every $i \in I$, if $x^*_{ih} > x^*_{ij}$ then

make $I_1 \leftarrow I_1 \cup \{i\}$. By construction, sets S, I_1 will violate constraint (6) iff they define a violated PC. As only a polynomial number of maximum-flow problems are solved, the method remains polynomial in $|I \cup J|$.

4.9 Location-routing comb inequalities

We present a tabu search algorithm for separating a subset of constraints LRCOMB in which $|T_j \cap I| \in \{0, 1\}$ for all j. Given a customer set H and t teeth $\Pi = (T_j)_{j=1}^t$ we call them a pseudo-comb if they satisfy conditions (iii)-(iv) of the definition of a comb, and $|T_j \cap I| \leq 1$ for all $1 \leq j \leq t$. Our separation algorithm proceeds in three stages: i) We search for ϵ-strengthened comb inequalities (SCI), getting candidate handles and teeth; ii) We use a greedy heuristic that breaks intersections (teeth can intersect in a SCI) by deleting elements that appear in two or more teeth from those that make the violation the greatest. If all the depots appear in a tooth, we delete all these depots except the one with the greatest violation. This process is repeated as many times as needed in order to obtain a pseudo-comb; iii) For every candidate pseudo-comb found after i) and ii), we proceed with the following tabu search metaheuristic.

Let us consider a pseudo-comb $C = (H, \Pi = (T_j)_{j=1}^t)$. Define $v(C)$ equal to the difference between the left-hand side of (56) and the right-hand side of (56). If C is a valid comb, then $v(C)$ represents the violation of the comb. Let us define the pseudo-violation $\mu(C)$ equal to

$$\mu(C) = v(C) - \sum_{j=1}^{t} \delta(H \cap T_j = \emptyset) - \sum_{j=1}^{t} \delta(T_j \setminus H = \emptyset) - \delta(\widehat{r}(H, \Pi) \equiv_2 0) - \sum_{j=1}^{t} \delta(\Lambda(H, T_j) \equiv_2 1).$$

The idea of considering the pseudo-violation instead of just the violation is justified by the fact that our procedure passes through pseudo-combs. Let $\mathcal{T}$ be the tabu list. A member l of $\mathcal{T}$ has two components, say $n(l)$ equal to a node and $pos(l)$ equal to a position relative to the comb. Here $pos(l)$ can take four values: $H \setminus \Pi, H \cap \Pi, \Pi \setminus H$ and $\overline{(H, \Pi)}$, where $\overline{(H, \Pi)}$ is the set containing all nodes not in the pseudo-comb (H, Π). Constructed in this way, the goal of the list $\mathcal{T}$ is to forbid the movement of a node $n(l)$ to position $pos(l)$ during a certain number of iterations.

Given a pseudo-comb $C = (H, \Pi = (T_j)_{j=1}^t)$ we consider several simple neighborhoods, all of which can be evaluated very quickly.

$\mathbf{N}_1$ Pick a customer j from $H \setminus \Pi$ and remove it from C. Add $(j, H \setminus \Pi)$ to T.

$\mathbf{N}_2$ Pick a customer j from $H \cap \Pi$ and remove it from H. Add $(j, H \cap \Pi)$ to T.

$\mathbf{N}_3$ Pick a customer j from $H \cap \Pi$ and remove it from Π. Add $(j, H \cap \Pi)$ to T.

$\mathbf{N}_4$ Pick a customer j from $\Pi \setminus H$ and remove it from C. Add $(j, \Pi \setminus H)$ to T.

$\mathbf{N}_5$ Pick a facility i from $\Pi \setminus H$ and remove it from C. Add $(i, \Pi \setminus H)$ to T.

$\mathbf{N}_6$ Pick a customer j from $\overline{C}$ and add it to $H \setminus \Pi$. Add $(j, \overline{C})$ to T.

$\mathbf{N}_7$ Pick a customer j from $\Pi \setminus H$ and add it to H. Add $(j, \Pi \setminus H)$ to T.

$\mathbf{N}_8$ Pick a customer j from $H \setminus \Pi$ and add it to Π. Add $(j, H \setminus \Pi)$ to T.

$\mathbf{N}_9$ Pick a customer j from $\overline{C}$ and add it to $\Pi \setminus H$. Add $(j, \overline{C})$ to T.

$\mathbf{N}_{10}$ Pick a facility i from $\overline{C}$ and add it to $\Pi \setminus H$. Add $(i, \overline{C})$ to T.

The neighborhoods are sorted in such a way that removal and insertion movements are alternated. If, after inspecting some neighborhood, we get a pseudo-violation of value greater than the incumbent, we update the incumbent and restart. Otherwise, we continue with the next neighborhood. We have found convenient to start the next iteration inspecting the first neighborhood not inspected during the last iteration. If we finish inspecting all the neighborhoods without finding any pseudo-comb with value greater than the incumbent, we update it with the best movement found and restart. During the process we do not consider movements of nodes to a tabu position, thus decreasing the probability of cycling. Note also that for neighborhoods N_5 and N_{10}, the contribution to the pseudo-violation depends on whether we are in the case $1 \leq j \leq s_1$ or $s_1 < j \leq s_2$ in the definition of a comb. We have chosen to make this distinction by simply considering the value of z_i in the current iteration. In fact, if $z_i < 0.75$ we consider the first case, otherwise the second. The algorithm finishes when we have found a valid comb with positive pseudo-violation, or when a maximum number of iterations has been performed without success. In our experiments we have noticed that most combs were found during the first 30 iterations. We have thus set the maximum number of iterations to 300 for the root node and 50 for the remaining nodes.

4.10 y-Generalized large multistar inequalities

The separation problem for the y-GLM (26) can be done in polynomial time by solving a maximum $\{s,t\}$-flow problem in the following graph $G' = (V', E')$. Let s and t be two dummy nodes, and let $V' = J \cup \{s,t\}$, $E' = E(J) \cup \{\{s,j\} : j \in J\} \cup \{\{j,t\} : j \in J\}$. With every edge $e \in E'$ we associate a capacity x'_e defined by

$$x'_e = \begin{cases} x^*(I:\{j\}) + 2\frac{d_j}{Q}\Big(y^*(I:\{j\}) - 1\Big) & e = \{s,j\}, j \in J \\ 0 & e = \{j,t\}, j \in J \\ x^*_e\Big(1 - 2\frac{d_j}{Q}\Big) & e = \{h,j\}, h,j \in J. \end{cases} \tag{60}$$

It is easy to check that a maximum $\{s,t\}$-flow exists in this graph with negative value iff there is a violated y-GLM. However, note that while maximum-flow algorithms assume positive edge capacities, this may not happen. Indeed, if $2d_j \leq Q$ for all $j \in J$ then the usual weight transformation on the edges joined to nodes s or t suffices. Suppose, however, that for some $j \in J$, $2d_j > Q$. The following transformation proposed by Blasum and Hochstättler [7] can be applied in order to get a non-negative weight digraph whose minimum-cut coincides with the one we are looking for. Define for every $j \in J$ the quantities $\underline{d}_j = \min\{\frac{Q}{2}, d_j\}, \overline{d}_j = d_j - \underline{d}_j$. Let us consider the following weight function:

$$x'_e = \begin{cases} x^*(I:\{j\}) + 2\frac{d_j}{Q}\Big(y^*(I:\{j\}) - 1\Big) & e = \{s,j\}, j \in J \\ -2\sum_{v \in J}\left(\frac{\underline{d}_v}{Q} - \frac{\underline{d}_j}{Q}\right) x^*_{jv} & e = \{j,t\}, j \in J \\ x^*_{hj}\left(1 - 2\left(\frac{\underline{d}_h}{Q} + \frac{\overline{d}_j}{Q}\right)\right) & e = \{h,j\}, h,j \in J. \end{cases} \tag{61}$$

It can be checked that a maximum $\{s,t\}$-flow in this modified graph is also a maximum flow in the original graph, and thus the separation algorithm of the y-GLM is polynomially solvable.

4.11 y-Location-routing generalized large multistar inequalities

The separation of constraints (34) is done in two stages. In the first stage, we separate what we call the Basic y-LRGLM (B-y-LRGLM) that corresponds to a y-LRGLM in the particular case of $|H| = 1$. For every $H = \{i\} \subset I$ we run an exact polynomial-time algorithm based on a maximum-flow computation, obtaining a candidate set S. Then, we use a greedy algorithm for enlarging the set H by inserting at each iteration the facility that makes the violation the greatest. For the separation of the B-y-LRGLM let us fix a facility $i \in I$. Let $G_i = (V_i = J \cup \{s,t\}, E_i = \delta(J) \cup (\{s\} : J) \cup (J : \{t\}))$ be the support graph, weighted as follows:

$$x'_e = \begin{cases} x^{I\setminus\{i\}}(I:j) + \frac{2d_j}{Q}\left(\sum_{l\in I\setminus\{i\}}(y_{lj} - u_{lj})\right) & e = \{s,j\}, j \in J \\ 0 & e = \{j,t\}, j \in J \\ x^{I\setminus\{i\}}_{hj}\left(1 - \frac{2d_j}{Q}\right) & h,j \in J. \end{cases} \tag{62}$$

Again, if the weights on the edges are negative, we apply the same transformation as for the separation of the y-GLM. It is easy to check that a violated B-y-LRGLM exists iff the minimum $\{s,t\}$-cut in this graph is negative.

4.12 Location-routing generalized large multistar inequalities

We have implemented the following heuristic procedure for the separation of the LRGLM (57) strengthened using as right-hand side the expression (58). First, we use an exact algorithm for finding a ϵ-LRGLM in the particular case in which $|I'| = 1$. We call these inequalities Basic LRGLM (B-LRGLM). For every pair of sets S and $I' = \{i\}$ found by this procedure, we apply a greedy heuristic that iteratively enlarges I' and checks for the violation of the corresponding LRGLM. The exact procedure used for the separation of the B-LRGLM is as follows.

Let $i \in I$, and let us consider a digraph whose vertex set is $J \cup \{i\} \cup \{s\}$, where s is the node obtained by the contraction of facilities in $I \setminus \{i\}$. The edge set is determined by the non-zero weights in the arcs, given by

$$x'_{uv} = \begin{cases} -\frac{d_j}{Q}(x^*_{iu} + 2y^*_{iu}) & u \in J, v = i \\ x^*(I\setminus\{i\} : u) + 2y^*(I\setminus\{i\} : u) - \frac{2d_u}{Q} & u = s, v \in J \\ x^*_{uv}\left(1 - 2\frac{d_v}{Q}\right) & u,v \in J. \end{cases}$$

It is easy to check that a violated LRGLM exists iff a minimum $\{s,i\}$-cut in this digraph has value less than $-\frac{2b_i}{Q}z^*_i$. In the case of negative weights, we apply the same procedure already described for the separation of the y-GLM. Thus, the problem of finding a LRGLM can be solved in polynomial time by computing a minimum $\{s,i\}$-cut in this graph.

4.13 Lifted cover inequalities

We add LCI only at the root node. We use the algorithm of Gu et al. [12] for finding violated LCI. For details on the algorithm we refer to Gu et al. [12].

5 The exact algorithms

We test the models, separation routines and valid inequalities introduced in this paper by developing four branch-and-cut algorithms. The first, named VFF2, is a branch-and-cut over the two-index vehicle-flow formulation (VF2) augmented by the valid inequalities introduced in this paper except those that are specific to the three-index formulations. The second algorithm, named VFF3, is a branch-and-cut on the three-index vehicle-flow formulation augmented by all the valid inequalities. The third algorithm, named CFF2, is branch-and-cut algorithm over the two-index two-commodity flow formulation (CF2) augmented by all the inequalities introduced in this paper except those that are specific to the three-index formulations and the y-GLM. The fourth algorithm, named CFF3, is a branch-and-cut algorithm over the three-index two-commodity flow formulation (CF3) augmented by all the inequalities except for y-GLM, LRGLM and y-LRGLM. For the two-commodity formulations, we also replace vehicle-flow variables x with their corresponding commodity-flow variables w using identities (22) and (30) and by adding (as cutting planes) inequalities (44) and (46) for the two-index and three-index formulations commodity-flow formulations, respectively. For the vehicle-flow formulations, we also add inequalities (43) and (45) dynamically as cutting planes.

5.1 The separation strategies

The separation strategies for the different formulations depend on two criteria: strength of the inequalities and need for feasibility. Inequalities that are needed to impose feasibility of integer solutions are thus separated first, while the rest are added as cutting planes, and among these two families the priority is given to inequalities that, in our experiments, have shown a bigger impact on the lower bounds. The exception are inequalities FAI that are separated immediately after the LCI. After some preliminary tests we have found the convenience of separating these inequalities before any other family of cuts. In addition, inequalities ESFCI and SFCI that seem to have an important impact in formulations VFF3, CFF2 and CFF3. Although they are not needed to impose feasibility, they are also separated first. Taking these observations into account, we have decided to divide the inequalities into two groups: those that are statically separated (i.e., separated in every node of the branching tree) and those for which we dynamically decide whether to separate them or not in a certain node of the branching tree. The criteria for selecting these dynamic cuts are explained later. In Table 1 we describe the two groups of inequalities as well as their separation order for each of the four different formulations considered in our study.

Note that in order to avoid errors due to floating point arithmetic, a certain tolerance $\epsilon > 0$ must be imposed for checking the violation of a certain cut. Moreover, if ϵ is too small, many cuts whose violations are very close to zero will be added without much impact on the lower bound. After a series of experiments, we have decided to use $\epsilon = 0.1$ for all of the

Form.	Static Cuts	Dynamic Cuts
VFF2	FAI (43), y-CC, ESFCI (51), SFCI (49), SPC	LCI, FDC (40), CoCC, FrCI, y-GLM, LRGLM, SCI, LRCOMB, MSI, HYP
VFF3	FAI (45), y-CC, ESFCI (52), SFCI (50), y-GLM, y-LRGLM, LRGLM	LCI, FDC (40) and (48), CoCC, DCoCC, SPC, FrCI, SCI, LRCOMB, MSI, HYP
CFF2	FAI (44), y-CC, ESFCI (51), SFCI (49), SPC	LCI, FDC (40), CoCC, FrCI, LRGLM SCI, LRCOMB, MSI, HYP
CFF3	FAI (46), y-CC, ESFCI (52), SFCI (50)	LCI, CoCC, DCoCC, FDC (40) and (48), SPC, FrCI, SCI, LRCOMB, MSI, HYP

Table 1: Separation order of valid inequalities

cuts except for hypotour inequalities and multistar inequalities for which the tolerance was set to $\epsilon = 0.4$. At the root node, all families of cuts are separated. Moreover, all separation algorithms are used for each family. More specifically, for the FrCI, the tree size is set to a maximum of 10,000 nodes, as for the LRCOMB the number of iterations of the Tabu Search is set to 300.

For the cutting strategy in nodes other than the root, we use the following approach. For each family of dynamic cuts (see Table 1), say for family $\mathcal{C}$, we let $n(\mathcal{C})$ be the number of times that a cut of family $\mathcal{C}$ has been found to be violated and thus added to the problem. We keep track of this quantity in the different branches of the tree and at certain depths we check whether $\mathcal{C}$ has been useful in the current branch. If $n(\mathcal{C}) = 0$ then the family $\mathcal{C}$ is not separated anymore during the current branch. For the other cuts, say those such that $n(\mathcal{C}) > 0$ the counter is reset to 0. After some testing we have decided to perform this check for the first time at depth 10 and then for multiples of 5. In practice, we have verified that no dynamic cuts are present after depth 25. Note also that the tree size in the separation of the inequalities FrCI is lowered to 200 nodes, while the maximum number of iterations of the Tabu Search algorithm for the separation of LRCOMB is lowered to 50.

Regarding the setting of the cut lifting heuristic described in Section 4.1, we have performed a series of tests in order to choose the value of ϵ that fits best with every cut family. The values that we have tested are ϵ equal to 0, 0.25, 0.50 and 1.0. For y-CC we have decided to set $\epsilon = 0.25$ at the root node and $\epsilon = 0$ for the remaining nodes (recall that cuts are not added unless they are violated by more than 0.1). For the SFCI and ESFCI we have set $\epsilon = 0.25$ during the whole computation.

5.2 The branching strategy

We use the following branching strategy. We first branch on location variables z. If no variable z is fractional, we branch on cutsets. For this, we use and idea proposed in Belenguer et al. [6]: during the root relaxation, each y-CC cut is added as an equality constraint by adding an extra slack variable to the problem. We then let CPLEX branch on these slack variables. For the three-index formulations, we then branch on the assignment variables u. Finally, we branch on the vehicle-flow variables y or x (or in their equivalent expressions using variables w in the commodity-flow formulations). We have observed that while strong

branching produces the smallest branching trees, the computational effort is too high and for hard instances it is not worthwhile. On the other hand, branching on the most fractional variables leads to much bigger branching trees. Thus, we let CPLEX branch based on pseudo costs, which we found to give the best balance between lower bound quality and CPU time.

6 Computational experience

In this section we describe the implementation of the algorithms as well as the results obtained on a series of instances from the litterature.

The algorithms have been coded in C++ using the Concert Technology framework of CPLEX 12.2. Tests were run on an Intel Xeon E5462, 3.0 Ghz processor with 16GB of memory under the Linux Operating System kernel 2.6. In order to obtain results purely related to the strength of the formulations and the cuts used in this paper, other families of cuts added by CPLEX (such as MIR, knapsack cover, GUB, clique, etc.) have been disabled. Finally, the node selection strategy has been set to best-first search.

We have run our algorithm on four datasets taken from the literature. The instances descriptions are as follows:

i. Set $\mathcal{S}_1$ contains 17 instances adapted by Barreto [5] from other problems in the literaure. Only three instances have facilities with limited capacities.

ii. Set $\mathcal{S}_2$ contains 24 randomly generated instances from the experiments of Belenguer et al. [6]. All of the instances have facilities with limited capacities. Customer loads are taken randomly in the interval $[11, 20]$ and capacities are set in such a way that: 1) the average number of customers served by a vehicle is either 5 or 10, and 2) two or three facilities are required for serving the whole demand. Note that no customer with extremely low (10 units or less) or extremely high (more than 20 units) demands is present.

iii. Set $\mathcal{S}_3$ contains 12 randomly generated instances from the experiments of Akca et al. [1]. Facilities all have limited capacities, chosen in such a way that at least two of the facilities must be open. The vehicle capacities are such that the average number of customers per route is between 4 and 7, and that the longest route serves at most 8 customers.

iv. Set $\mathcal{S}_4$ contains 6 instances with capacitated vehicles and uncapacitated facilities from the experiments of Tuzun and Burke [24]. The fixed costs of the facilities are relatively low compared to the routing costs.

Additionally, sets $\mathcal{S}_1$ and $\mathcal{S}_2$ are also subdivided into small instances (those with 50 customers or less) and large instances (those having more than 50 customers). We have used the upper bounds reported by Baldacci et al. [4] as cutoff values during the branch-and-bound search. The idea is to measure the efficiency of each of the formulations for closing the optimality gap. For a self-contained methodology, these upper bounds should be obtained by a suitable heuristic, which is beyond the scope of this paper. The data sets

can all be obtained from the website `http://www.crt.umontreal.ca/~ccontard`. We have designed and implemented five sets of experiments.

In the first set of experiments, we compute the linear relaxation lower bound for each of the four formulations, and compare their quality as well as the CPU time taken by each of them. The results are reported in Tables 2-5. In these tables, columns labeled z^* represent the upper bound for each instance. Columns labeled *gap (%)* and *t (s)* stand for the relative gap (for a given lower bound z_{lb} it is computed as $(z^* - z_{lb})/z^* \times 100$) and the CPU time in seconds. As shown by these tables, algorithms VFF3 and CFF3 normally produce the tightest lower bounds, at the expense of much larger computing times. However, algorithms VFF2 and CFF2 are the fastest to compute their respective lower bounds. There are two possible readings for these results. On the one hand, compact two-index formulations give reasonably good lower bounds in very short computing times. Therefore, much larger branching trees can be inspected during the same amount of time, with respect to formulations with more variables. On the other hand, the lower bounds obtained by three-index formulations are in some cases much tighter than the ones obtained by the two-index formulations. Therefore, the structure of the CLRP is better captured in the former case, and in some instances the differences are dramatic (like on instances of set $\mathcal{S}_4$). One could thus ask, whether it is possible or not to tighten two-index formulations with valid inequalities so to produce lower bounds that are comparable to those obtained by three-index formulations.

In the second set of experiments, we have run the four algorithms for a maximum time of two hours. The objective is to test and compare their efficiency to rapidly solve some relatively easy instances. In Tables 6-9, columns are similar to previous tables. We have added a column labeled *nodes* that reports the number of nodes inspected during the branching tree. As we can see, formulation VFF2 gives the best results on average. Indeed, it is able to solve 32 instances, four more than VFF3, three more than CFF2 and 6 more than CFF3. However, three-index formulations produce tighter gaps on instances ppw-50x5-0b and ppw-50x5-2b. This suggests that two-index formulations are not able in those cases to capture some important underlying information of the CLRP structure that is indeed beneficial to three-index formulations. Moreover, instance ppw-50x5-0b is solved to optimality only by formulation VFF3. The overall conclusion is that compact two-index formulations produce the best average results at the expense of underestimating some important information.

In the third set of experiments, we have run the algorithms for a maximum time of 12 hours. The objective is to measure the efficiency of each formulations for solving of some hard instances of the CLRP. The results are summarized in Tables 10-13. The columns are the same as for the previous experiments. Now, the number of instances solved is 32 for VFF2 and VFF3, 30 for CFF2 and 29 for CFF3. Note that this increase in the cpu time has a marginal impact on the performance of two-index formulations, whereas three-index formulations seem to scale better. The is due mainly to the fact that branching has a lower impact on trees of large size, which is typically the case with compact two-index formulations.

In the fourth set of experiments, we compare algorithm VFF2 against the branch-and-cut algorithm of Belenguer et al. [6]. In Table 14, headers *# instances*, *# solved* and *avg. gap* stand for the total number of instances, the number of instances solved to optimality and the average optimality gap on each subset of instances. Our implementation of the branch-and-cut algorithm VFF2 is able to produce tighter gaps in average than the one of Belenguer et al. [6]. Moreover, our algorithm scales to solve some large instances with up

to 100 customers (ppw-100x10-2b, P112212, P113212), while the method of Belenguer et al. [6] was able to solve instances with up to 50 customers. Several refinements in our implementation might explain these results, including the use of stronger inequalities, efficient separation algorithms, as well as the dynamic separation strategy during the branching tree that deactivates cuts that do not seem promising in a certain branch.

In the fifth and last set of experiments, we compare the results obtained by our branch-and-cut algorithms against the branch-and-cut-and-price method of Baldacci et al. [4]. In Table 15 we summarize the number of instances solved by their method (column BMW) against the instances solved by all of our methods (column FF). As shown in this table, the branch-and-cut-and-price method of Baldacci et al. [4] is able to solve much more instances than all of the flow formulations together. This is not a surprising result since column generation algorithms are based on much tighter formulations. However, it is worth noticing that their method failed to solve instance ppw-50x5-2b which has been solved by algorithm VFF3, and also solves instance ppw-50x5-0b in a much longer time than VFF3, which suggests that some of the inequalities introduced in this paper would deserve being included into set-partitioning formulations.

7 Concluding Remarks

We have introduced three new flow formulations for the CLRP that dominate, in terms of the linear relaxation lower bound, the previous two-index vehicle-flow formulation of Belenguer et al. [6]. We derive new valid inequalities for each of the formulations and strengthen some of the previously known inequalities. In addition, we are able to obtain new classes of multistar inequalities for the vehicle-flow formulations as linear combinations of the degree constraints and assignment constraints for the commodity-flow formulations. For each of the inequalities used in this paper, we introduce separation algorithms that are either new or that generalize the separation methods introduced by Belenguer et al. [6]. We have implemented suitable branch-and-cut algorithms using each of the three formulations introduced in this paper plus the original two-index vehicle-flow formulation and present computational results comparing them. The results show that, in most cases, compact formulations produce the tightest gaps in the long run due to their ability to perform more branching nodes. However, on some hard instances where facility capacities are important, three-index formulations seem to be the right choice (like on instances ppw-50x5-0b, ppw-50x5-2b, ppw-100x5-3b, ppw-100x10-3b). This is a direct consequence of an important drawback of compact two-index formulations with respect to three-index formulations, and it is the fact that it is not possible to follow the flow leaving from a facility at every single node of the graph. We also compare the algorithms used in this paper against the state-of-the-art solvers for solving the CLRP, namely the branch-and-cut method of Belenguer et al. [6] and the branch-and-cut-and-price of Baldacci et al. [4]. The results show that our implementation of the branch-and-cut on the two-index vehicle-flow formulation produces tighter gaps than the one of Belenguer et al. [6], and is able to scale and solve large instances with up to 100 customers. The branch-and-cut-and-price algorithm of Baldacci et al. [4] in general outperforms the flow-based algorithms; however, it is worth remarking that on two instances (ppw-50x5-0b, ppw-50x5-2b) the three-index formulation obtained tighter gaps, and even solved ppw-50x5-2b which no other exact

method did before. These results suggest that taking into consideration the facilities from where the flow originates has significant impact on the performance of an exact algorithm. As an avenue of future research, we believe that embedding some of the inequalities introduced in this paper into a branch-and-cut-and-price algorithm could result in a more robust exact algorithm for the CLRP.

Acknowledgements

The authors would like to thank the *Natural Sciences and Engineering Research Council of Canada* (NSERC) and *Le fonds québécois de la recherche sur la nature et les technologies* (FQRNT) for their financial support.

Instance	z^*	VFF2		VFF3		CFF2		CFF3	
		gap (%)	t (s)	gap (%)	t (s)	gap (%)	t (s)	gap (%)	t (s)
Perl83-12x2	204.00	0.61	0.01	0.00	0.03	0.48	0.08	0.00	0.03
Gas67-21x5	424.90	3.99	0.21	3.12	0.84	4.12	0.44	2.98	0.53
Gas67-22x5	585.11	0.10	0.04	0.10	0.24	0.10	0.07	0.10	0.41
Min92-27x5	3062.02	5.62	0.29	2.15	2.26	6.24	0.39	2.64	1.32
Gas67-29x5	512.10	4.89	0.46	3.26	2.11	4.76	1.90	3.64	1.74
Gas67-32x5	562.22	5.72	0.51	3.90	3.59	5.72	1.26	4.05	1.33
Gas67-32x5-2	504.33	3.27	0.80	1.86	2.17	3.24	1.25	2.19	1.29
Gas67-36x5	460.37	1.30	1.40	1.16	8.28	1.35	5.35	0.71	9.42
Chr69-50x5ba	565.62	5.62	3.74	4.41	14.32	5.63	6.40	3.83	6.76
Chr69-50x5be	565.60	8.85	3.04	7.34	16.15	8.82	15.44	5.90	5.45
Perl83-55x15	1112.06	3.42	6.17	2.44	676.90	3.42	14.29	2.06	64.25
Chr69-75x10ba	886.30	14.47	23.54	11.67	1406.12	14.63	164.87	11.47	306.63
Chr69-75x10be	848.85	10.42	15.25	7.77	1920.82	9.84	227.24	7.40	284.03
Chr69-75x10bmw	802.08	9.27	20.18	6.68	1165.53	9.69	92.79	6.58	237.99
Perl83-85x7	1622.50	2.53	18.93	2.06	966.41	2.53	199.44	1.96	153.35
Das95-88x8	355.78	5.73	13.15	4.81	1028.53	6.03	66.88	4.73	336.98
Chr69-100x10	833.43	4.89	9.71	4.07	2299.52	4.82	51.37	3.79	709.63
Average		5.34	6.91	3.93	559.64	5.38	49.97	3.77	124.77
Average in small instances		4.00	1.05	2.73	5.00	4.05	3.26	2.60	2.83
Average in large instances		7.25	15.28	5.64	1351.98	7.28	116.70	5.43	298.98

Table 2: Gaps and CPU times after linear relaxation on instances of set $\mathcal{S}_1$

Instance	z^*	VFF2		VFF3		CFF2		CFF3	
		gap (%)	t (s)	gap (%)	t (s)	gap (%)	t (s)	gap (%)	t (s)
ppw-20x5-0a	54793	4.57	0.35	3.74	1.00	4.54	0.83	3.89	0.68
ppw-20x5-0b	39104	0.00	0.02	0.00	0.10	0.00	0.04	0.00	0.14
ppw-20x5-2a	48908	2.71	0.26	2.31	0.59	2.78	0.76	2.35	0.53
ppw-20x5-2b	37542	0.00	0.01	0.00	0.06	0.00	0.02	0.00	0.10
ppw-50x5-0a	90111	10.94	27.10	5.98	100.66	10.89	51.78	5.88	26.38
ppw-50x5-0b	63242	7.50	5.05	6.64	28.54	7.76	14.70	6.67	16.58
ppw-50x5-2a	88298	7.52	5.08	5.81	36.84	7.50	17.22	5.84	11.04
ppw-50x5-2b	67308†	5.63	2.75	5.74	16.20	5.66	12.78	5.72	16.38
ppw-50x5-2a'	84055	1.95	29.50	1.89	65.33	1.97	125.81	1.93	25.09
ppw-50x5-2b'	51822	0.86	1.76	0.72	19.48	0.85	3.51	0.82	9.82
ppw-50x5-3a	86203	10.23	14.67	5.15	72.97	10.20	52.76	5.26	25.99
ppw-50x5-3b	61830	6.26	4.38	5.30	23.36	5.84	9.94	5.22	9.67
ppw-100x5-0a	274814	3.56	2509.03	2.82	4955.67	3.59	3117.51	2.86	1218.73
ppw-100x5-0b	214392	3.21	391.48	3.09	5605.32	3.18	1508.11	3.33	10298.00
ppw-100x5-2a	193671	3.77	365.93	2.17	2402.35	3.81	3127.55	2.21	460.93
ppw-100x5-2b	157173	2.34	83.27	1.91	816.75	2.32	613.30	1.96	365.99
ppw-100x5-3a	200079	8.82	108.07	2.23	2331.79	8.81	1664.74	2.39	441.84
ppw-100x5-3b	152441	5.08	27.40	2.62	792.25	5.08	148.33	2.66	163.83
ppw-100x10-0a	289018	7.88	1133.84	5.78	7281.82	7.34	4708.86	4.97	1249.72
ppw-100x10-0b	234641	4.74	147.20	4.53	4060.38	4.78	1235.21	4.45	1638.18
ppw-100x10-2a	243590	4.07	1473.84	3.28	7285.58	4.11	3823.62	3.21	1214.73
ppw-100x10-2b	203988	2.50	90.42	2.48	1928.96	2.46	612.71	2.34	1308.84
ppw-100x10-3a	252421	8.65	740.38	6.17	7228.90	8.72	3433.17	6.28	1188.19
ppw-100x10-3b	204597	5.00	112.22	4.75	4384.28	4.99	1011.74	4.60	857.56
Average		4.91	303.08	3.55	2059.97	4.88	1053.96	3.53	856.21
Average in small instances		4.85	7.58	3.61	30.43	4.83	24.18	3.63	11.87
Average in large instances		4.97	598.59	3.49	4089.50	4.93	2083.74	3.44	1700.55

† New upper bound found.

Table 3: Gaps and CPU times after linear relaxation on instances of set $\mathcal{S}_2$

Instance	z^*	VFF2		VFF3		CFF2		CFF3	
		gap (%)	t (s)	gap (%)	t (s)	gap (%)	t (s)	gap (%)	t (s)
cr30x5a-1	819.5	3.33	0.89	2.06	3.71	3.29	1.62	2.99	2.09
cr30x5a-2	821.5	5.89	0.41	5.29	2.61	5.90	0.97	4.92	1.67
cr30x5a-3	702.3	0.56	0.71	0.09	2.52	0.73	1.22	0.38	2.66
cr30x5b-1	880.0	7.39	0.52	5.91	3.00	7.35	1.55	5.61	1.35
cr30x5b-2	825.3	3.52	1.31	1.65	3.71	3.62	3.31	1.72	1.54
cr30x5b-3	884.6	3.33	1.09	2.14	4.47	3.25	2.73	2.20	2.01
cr40x5a-1	928.1	8.95	1.32	8.01	9.20	8.96	3.28	7.08	2.01
cr40x5a-2	888.4	8.83	1.04	6.17	9.91	8.92	1.95	6.09	3.63
cr40x5a-3	947.3	7.47	2.48	6.31	9.43	7.45	7.56	5.50	4.91
cr40x5b-1	1052.0	10.26	2.80	6.64	16.68	10.13	6.70	6.52	4.66
cr40x5b-2	981.5	8.57	1.26	3.70	16.18	8.42	4.55	3.79	5.41
cr40x5b-3	964.3	4.51	2.32	2.92	17.40	4.46	8.22	2.94	3.94
Average		6.05	1.35	4.24	8.23	6.04	3.64	4.14	2.99

Table 4: Gaps and CPU times after linear relaxation on instances of set $\mathcal{S}_3$

Instance	z^*	VFF2		VFF3		CFF2		CFF3	
		gap (%)	t (s)	gap (%)	t (s)	gap (%)	t (s)	gap (%)	t (s)
P111112	1467.69	12.64	16.31	7.94	2180.38	12.60	119.93	6.91	313.17
P111212	1394.8	15.92	41.04	11.37	1763.18	15.91	214.59	9.09	451.88
P112112	1167.16	11.69	42.24	3.69	3245.86	11.91	339.71	3.72	547.29
P112212	791.66	19.99	53.95	2.97	1021.34	20.01	111.13	2.94	274.73
P113112	1245.45	19.27	31.51	7.84	4987.70	19.51	130.71	7.74	637.84
P113212	902.26	16.49	83.95	1.82	4383.76	16.90	774.21	1.96	1601.76
Average		16.00	44.83	5.94	2930.37	16.14	281.71	5.39	637.78

Table 5: Gaps and CPU times after linear relaxation on instances of set $\mathcal{S}_4$

Instance	z^*	VFF2			VFF3			CFF2			CFF3		
		gap (%)	t (s)	nodes	gap (%)	t (s)	nodes	gap (%)	t (s)	nodes	gap (%)	t (s)	nodes
Perl83-12x2	204.00	0.00	0.02	0	0.00	0.05	0	0.00	0.11	3	0.00	0.08	0
Gas67-21x5	424.90	0.00	0.59	26	0.00	4.29	19	0.00	1.91	39	0.00	4.76	20
Gas67-22x5	585.11	0.00	0.07	0	0.00	0.45	1	0.00	0.19	7	0.00	1.41	9
Min92-27x5	3062.02	0.00	0.73	5	0.00	3.94	3	0.00	1.78	20	0.00	7.87	6
Gas67-29x5	512.10	0.00	1.01	10	0.00	7.28	13	0.00	3.84	20	0.00	18.06	11
Gas67-32x5	562.22	0.00	1.76	47	0.00	19.90	39	0.00	4.98	39	0.00	27.58	47
Gas67-32x5-2	504.33	0.00	1.01	2	0.00	4.72	5	0.00	2.32	7	0.00	3.96	9
Gas67-36x5	460.37	0.00	2.80	4	0.00	15.50	7	0.00	12.07	16	0.00	43.64	23
Chr69-50x5ba	565.62	0.00	44.78	224	0.00	530.77	169	0.00	212.91	384	0.00	1655.58	314
Chr69-50x5be	565.60	0.00	68.79	232	0.00	1093.98	351	0.00	200.49	258	0.00	4999.94	476
Perl83-55x15	1112.06	0.70	7496.92	2755	1.45	7269.41	30	0.85	7193.60	1090	1.14	7215.43	98
Chr69-75x10ba	886.30	9.03	7408.62	1869	10.67	7401.99	16	9.07	7193.13	1304	10.38	7315.95	22
Chr69-75x10be	848.85	3.26	7367.52	2394	6.10	7339.88	18	3.64	7193.51	1293	5.76	7472.10	28
Chr69-75x10bmw	802.08	3.37	7468.83	1766	5.25	7351.64	22	3.69	7194.08	1403	5.16	7417.08	30
Perl83-85x7	1622.50	0.68	7557.62	1404	1.29	7259.75	14	0.75	7193.95	1044	1.17	7250.88	19
Das95-88x8	355.78	0.00†	411.15	129	0.73	7267.20	8	0.00†	1308.87	289	0.32	7275.53	21
Chr69-100x10	833.43	0.51	7385.58	1777	2.83	7326.33	9	0.80	7193.80	330	2.46	7397.48	17
Average		1.03	2659.87	744	1.67	3111.59	43	1.11	2641.86	444	1.55	3418.08	68
Instances solved		11			10			11			10		
Average in small instances		0.00	12.16	55	0.00	168.09	61	0.00	44.06	79	0.00	676.29	92
Small instances solved		10			10			10			10		
Average in large instances		2.51	6442.32	1728	4.05	7316.60	17	2.69	6352.99	965	3.77	7334.92	34
Large instances solved		1			0			1			0		

† Optimality proven for the first time.

Table 6: Gaps and CPU times after 2 hours on instances of set $\mathcal{S}_1$

Instance	z^*	VFF2			VFF3			CFF2			CFF3		
		gap (%)	t (s)	nodes	gap (%)	t (s)	nodes	gap (%)	t (s)	nodes	gap (%)	t (s)	nodes
ppw-20x5-0a	54793	0.00	5.04	704	0.00	12.61	2	0.00	21.70	1009	0.00	25.59	197
ppw-20x5-0b	39104	0.00	0.03	0	0.00	0.16	0	0.00	0.11	8	0.00	0.50	6
ppw-20x5-2a	48908	0.00	1.31	137	0.00	5.93	1	0.00	4.65	191	0.00	7.44	77
ppw-20x5-2b	37542	0.00	0.02	0	0.00	0.12	0	0.00	0.08	9	0.00	0.49	3
ppw-50x5-0a	90111	1.77	7336.51	9778	2.46	7207.13	186	2.39	7193.53	5286	2.78	7210.67	949
ppw-50x5-0b	63242	1.23	7304.54	16772	0.00	1865.40	59	1.77	7193.39	6014	0.38	7199.65	1608
ppw-50x5-2a	88298	1.00	7265.57	13181	1.05	7198.80	193	1.21	7193.73	8548	1.24	7197.10	1990
ppw-50x5-2b	67308	1.17	7282.73	17630	0.68	7199.72	114	1.50	7193.41	7854	0.87	7201.81	1086
ppw-50x5-2a'	84055	0.28	7244.32	24306	0.48	7203.79	183	0.49	7193.36	11205	0.55	7203.78	1949
ppw-50x5-2b'	51822	0.00	10.64	134	0.00	136.19	7	0.00	41.93	289	0.00	125.11	91
ppw-50x5-3a	86203	1.16	7283.89	4915	2.09	7199.29	113	1.46	7194.24	2894	2.20	7210.56	632
ppw-50x5-3b	61830	0.00	71.76	596	0.00	647.18	17	0.00	378.75	1000	0.00	1368.30	348
ppw-100x5-0a	274814	2.36	7293.80	350	2.82	7454.50	185	2.48	7196.30	119	2.68	7515.34	77
ppw-100x5-0b	214392	2.19	7420.00	460	3.09	7438.56	111	2.41	7197.83	197	3.33	10298.00	0
ppw-100x5-2a	193671	1.60	7398.86	726	2.17	7413.93	118	1.65	7196.96	505	2.21	7281.76	0
ppw-100x5-2b	157173	0.78	7323.10	1777	1.40	7270.06	97	0.88	7195.16	531	1.17	7231.19	12
ppw-100x5-3a	200079	1.44	7340.95	1379	1.93	7316.05	364	1.64	7197.17	546	1.64	7270.58	290
ppw-100x5-3b	152441	0.57	7332.99	1225	0.74	7240.37	204	0.49	7196.00	329	0.85	7272.68	25
ppw-100x10-0a	289018	3.74	7394.33	59	5.78	7281.82	114	7.02	7243.67	0	4.49	7455.56	369
ppw-100x10-0b	234641	2.48	7334.20	821	4.50	7509.37	68	2.98	7194.80	32	4.17	7919.87	5
ppw-100x10-2a	243590	1.40	7351.82	764	3.28	7285.58	95	1.54	7197.52	73	2.10	7505.96	330
ppw-100x10-2b	203988	0.00	4734.83	8710	0.97	7293.31	24	0.11	7193.51	1698	0.82	7410.55	30
ppw-100x10-3a	252421	4.02	7326.18	85	6.17	7228.90	100	4.70	7195.95	16	4.98	7459.89	579
ppw-100x10-3b	204597	2.15	7338.32	1033	4.00	7412.70	55	2.55	7194.91	56	3.95	7382.64	4
Average		1.22	5391.49	4398	1.82	5284.23	100	1.55	5417.03	2017	1.68	5698.13	444
Instances solved		7			7			6			6		
Average on small instances		0.55	3650.53	7346	0.56	3223.03	73	0.73	3634.07	3692	0.67	3729.25	745
Small instances solved		6			7			6			6		
Average on large instances		1.89	7132.45	1449	3.07	7345.43	128	2.37	7199.98	342	2.70	7667.00	143
Large instances solved		1			0			0			0		

Table 7: Gaps and CPU times after 2 hours on instances of set $\mathcal{S}_2$

Instance	z^*	VFF2			VFF3			CFF2			CFF3		
		gap (%)	t (s)	nodes	gap (%)	t (s)	nodes	gap (%)	t (s)	nodes	gap (%)	t (s)	nodes
cr30x5a-1	819.5	0.00	3.23	67	0.00	27.25	11	0.00	7.65	88	0.00	35.05	46
cr30x5a-2	821.5	0.00	8.77	663	0.00	44.50	253	0.00	59.83	1244	0.00	38.20	234
cr30x5a-3	702.3	0.00	0.91	0	0.00	3.50	0	0.00	1.77	38	0.00	7.82	27
cr30x5b-1	880.0	0.00	9.05	395	0.00	86.25	235	0.00	34.66	439	0.00	90.66	184
cr30x5b-2	825.3	0.00	2.55	8	0.00	14.78	9	0.00	7.26	113	0.00	14.57	43
cr30x5b-3	884.6	0.00	3.25	40	0.00	21.85	11	0.00	11.06	137	0.00	28.52	92
cr40x5a-1	928.1	0.00	140.31	1308	0.00	1391.66	783	0.00	608.40	1342	0.14	7328.59	2143
cr40x5a-2	888.4	0.00	86.31	1655	0.00	591.68	601	0.00	203.61	935	0.00	1059.93	692
cr40x5a-3	947.3	0.00	76.63	1019	0.00	785.90	463	0.00	409.66	1286	0.00	1494.26	542
cr40x5b-1	1052.0	0.00	3115.92	21471	0.86	7197.27	2396	0.57	7193.77	9358	1.03	7196.37	2243
cr40x5b-2	981.5	0.00	7.61	25	0.00	104.38	45	0.00	46.77	1011	0.00	91.69	85
cr40x5b-3	964.3	0.00	12.33	20	0.00	50.08	31	0.00	24.30	74	0.00	54.01	49
Average		0.00	288.91	2223	0.07	859.93	403	0.05	717.39	1339	0.10	1453.31	532
Instances solved		12			11			11			10		

Table 8: Gaps and CPU times after 2 hours on instances of set $\mathcal{S}_3$

Instance	z^*	VFF2			VFF3			CFF2			CFF3		
		gap (%)	t (s)	nodes	gap (%)	t (s)	nodes	gap (%)	t (s)	nodes	gap (%)	t (s)	nodes
P111112	1467.69	2.76	7529.07	427	5.69	7389.51	9	3.17	7197.69	125	5.01	7361.46	25
P111212	1394.8	2.86	7444.58	594	8.39	7341.34	10	3.99	7193.81	147	6.42	7471.62	28
P112112	1167.16	0.49	7362.85	2205	2.22	7331.45	10	0.72	7197.28	542	2.22	7533.47	7
P112212	791.66	0.00	4980.13	8747	0.98	7326.49	10	0.20	7194.37	2763	0.97	7365.17	7
P113112	1245.45	3.64	7312.07	1954	7.83	7417.84	2	3.86	7195.71	537	4.99	7598.02	13
P113212	902.26	0.00	247.09	33	0.15	7373.92	3	0.00	1634.29	139	0.13	7689.31	5
Average		1.63	5812.63	2327	4.21	7363.43	7	1.99	6268.86	709	3.29	7503.18	14
Instances solved		2			0			1			0		

Table 9: Gaps and CPU times after 2 hours on instances of set $\mathcal{S}_4$

Instance	z^*	VFF2			VFF3			CFF2			CFF3		
		gap (%)	t (s)	nodes	gap (%)	t (s)	nodes	gap (%)	t (s)	nodes	gap (%)	t (s)	nodes
Perl83-12x2	204.00	0.00	0.02	0.00	0.00	0.05	0.00	0.00	0.11	3.00	0.00	0.08	0
Gas67-21x5	424.90	0.00	0.59	26.00	0.00	4.29	19.00	0.00	1.91	39.00	0.00	4.76	20
Gas67-22x5	585.11	0.00	0.07	0.00	0.00	0.45	1.00	0.00	0.19	7.00	0.00	1.41	9
Min92-27x5	3062.02	0.00	0.73	5.00	0.00	3.94	3.00	0.00	1.78	20.00	0.00	7.87	6
Gas67-29x5	512.10	0.00	1.01	10.00	0.00	7.28	13.00	0.00	3.84	20.00	0.00	18.06	11
Gas67-32x5	562.22	0.00	1.76	47.00	0.00	19.90	39.00	0.00	4.98	39.00	0.00	27.58	47
Gas67-32x5-2	504.33	0.00	1.01	2.00	0.00	4.72	5.00	0.00	2.32	7.00	0.00	3.96	9
Gas67-36x5	460.37	0.00	2.80	4.00	0.00	15.50	7.00	0.00	12.07	16.00	0.00	43.64	23
Chr69-50x5ba	565.62	0.00	44.78	224.00	0.00	530.77	169.00	0.00	212.91	384.00	0.00	1655.58	314
Chr69-50x5be	565.60	0.00	68.79	232.00	0.00	1093.98	351.00	0.00	200.49	258.00	0.00	4999.94	476
Perl83-55x15	1112.06	0.45	44278.60	14429.00	1.09	43228.20	85.00	0.57	43163.30	8651.00	0.77	43188.20	430
Chr69-75x10ba	886.30	8.53	43923.00	9619.00	10.31	43345.80	30.00	8.55	43162.10	7889.00	9.37	43289.40	452
Chr69-75x10be	848.85	2.69	43902.30	12090.00	5.58	43353.70	33.00	2.95	43161.20	7070.00	4.44	43355.40	102
Chr69-75x10bmw	802.08	2.91	44528.40	8496.00	4.44	43315.00	66.00	3.20	43166.10	8238.00	4.12	43409.60	189
Perl83-85x7	1622.50	0.52	44470.70	6905.00	0.84	43224.30	98.00	0.59	43164.30	6428.00	0.87	43215.60	93
Das95-88x8	355.78	0.00	411.15	129.00	0.00	26533.80	99.00	0.00	1308.87	289.00	0.00	25008.10	160
Chr69-100x10	833.43	0.26	43933.90	9206.00	1.51	43396.80	61.00	0.47	43168.30	1891.00	1.72	43376.40	52
Average		0.90	15621.74	3613	1.40	16945.79	63	0.96	15337.34	2426	1.25	17153.27	141
Instances solved		11			11			11			11		
Average in small instances		0.00	12.16	55	0.00	168.09	61	0.00	44.06	79	0.00	676.29	92
Small instances solved		10			10			10			10		
Average in large instances		2.19	37921.15	8696	3.39	40913.94	67	2.33	37184.88	5779	3.04	40691.81	211
Large instances solved		1			1			1			1		

Table 10: Gaps and CPU times after 12 hours on instances of set $\mathcal{S}_1$

Instance	z^*	VFF2			VFF3			CFF2			CFF3		
		gap (%)	t (s)	nodes	gap (%)	t (s)	nodes	gap (%)	t (s)	nodes	gap (%)	t (s)	nodes
ppw-20x5-0a	54793	0.00	5.04	704	0.00	12.61	107	0.00	21.70	1009	0.00	25.59	197
ppw-20x5-0b	39104	0.00	0.03	0	0.00	0.16	0	0.00	0.11	8	0.00	0.50	6
ppw-20x5-2a	48908	0.00	1.31	137	0.00	5.93	113	0.00	4.65	191	0.00	7.44	77
ppw-20x5-2b	37542	0.00	0.02	0	0.00	0.12	0	0.00	0.08	9	0.00	0.49	3
ppw-50x5-0a	90111	1.37	43595.50	33712	2.05	43176.50	4476	1.97	43167.70	18081	2.33	43180.30	4192
ppw-50x5-0b	63242	0.98	43595.50	58868	0.00	1865.40	2605	1.57	43167.20	20673	0.00	13194.70	4362
ppw-50x5-2a	88298	0.75	43418.50	47481	0.80	43172.10	7203	0.97	43167.60	31819	0.91	43170.40	8336
ppw-50x5-2b	67308	0.84	43574.60	61257	0.00†	42721.30	14967	1.19	43167.60	27182	0.30	43175.20	4949
ppw-50x5-2a'	84055	0.14	43350.80	95114	0.33	43177.50	11615	0.31	43167.20	50204	0.35	43175.00	10497
ppw-50x5-2b'	51822	0.00	10.64	134	0.00	136.19	155	0.00	41.93	289	0.00	125.11	91
ppw-50x5-3a	86203	0.94	43479.80	18603	1.71	43199.40	2784	1.26	43167.90	10656	1.86	43177.50	2496
ppw-50x5-3b	61830	0.00	71.76	596	0.00	647.18	247	0.00	378.75	1000	0.00	1384.13	348
ppw-100x5-0a	274814	2.19	43556.70	6805	2.39	43510.20	260	2.28	43171.20	2850	2.58	43421.20	651
ppw-100x5-0b	214392	1.99	44111.80	4588	3.07	44111.10	3	2.18	43168.90	3699	3.21	43200.00	0
ppw-100x5-2a	193671	1.43	44427.70	4532	1.78	43425.80	277	1.48	43170.50	4004	2.02	43276.10	342
ppw-100x5-2b	157173	0.63	43711.50	9989	0.69	43224.60	1365	0.74	43165.80	3986	0.72	43206.70	544
ppw-100x5-3a	200079	1.32	43846.10	6623	1.49	43356.30	933	1.46	43167.30	8919	1.46	43259.60	4015
ppw-100x5-3b	152441	0.40	43541.60	7460	0.18	43205.30	2766	0.29	43165.90	3141	0.76	43225.00	19
ppw-100x10-0a	289018	3.36	43811.60	2120	4.25	43436.60	863	3.32	43171.80	692	4.36	43477.00	4820
ppw-100x10-0b	234641	2.27	43671.20	7250	3.51	43439.10	37	2.41	43168.70	1291	3.39	43836.90	506
ppw-100x10-2a	243590	1.29	43969.70	6861	1.85	43437.00	8	1.31	43170.60	2354	1.98	43475.00	4218
ppw-100x10-2b	203988	0.00	4734.83	8710	0.65	44250.50	2598	0.01	43176.70	13392	0.62	43436.80	3613
ppw-100x10-3a	252421	3.59	44012.80	1924	4.62	43456.40	2021	2.38	43171.00	684	4.83	43413.40	5648
ppw-100x10-3b	204597	1.94	43661.60	6506	1.39	44132.90	9	2.10	43168.30	928	2.90	43357.90	1155
Average		1.06	31173.36	16249	1.28	30879.17	2309	1.14	32395.38	8628	1.44	31300.08	2545
Instances solved		7			8			6			7		
Average on small instances		0.42	21758.62	26384	0.41	18176.20	3689	0.61	21621.03	13427	0.48	19218.03	2963
Small instances solved		6			8			6			7		
Average on large instances		1.70	40588.09	6114	2.16	43582.15	928	1.66	43169.72	3828	2.40	43382.13	2128
Large instances solved		1			0			0			0		

† Optimality proven for the first time.

Table 11: Gaps and CPU times after 12 hours on instances of set $\mathcal{S}_2$

Instance	z^*	VFF2			VFF3			CFF2			CFF3		
		gap (%)	t (s)	nodes	gap (%)	t (s)	nodes	gap (%)	t (s)	nodes	gap (%)	t (s)	nodes
cr30x5a-1	819.52	0.00	3.23	67	0.00	27.25	11	0.00	7.65	88	0.00	35.05	46
cr30x5a-2	821.50	0.00	8.77	663	0.00	44.5	253	0.01	59.83	1244	0.00	38.20	234
cr30x5a-3	702.30	0.00	0.91	0	0.00	3.5	0	0.00	1.77	38	0.00	7.82	27
cr30x5b-1	880.02	0.00	9.05	395	0.00	86.25	235	0.00	34.66	439	0.00	90.66	184
cr30x5b-2	825.32	0.00	2.55	8	0.00	14.78	9	0.00	7.26	113	0.00	14.57	43
cr30x5b-3	884.60	0.00	3.25	40	0.00	21.85	11	0.00	11.06	137	0.00	28.52	92
cr40x5a-1	928.10	0.00	140.31	1308	0.00	1391.66	783	0.00	608.40	1342	0.00	8619.65	2143
cr40x5a-2	888.42	0.00	86.31	1655	0.00	591.68	601	0.00	203.61	935	0.00	1059.93	692
cr40x5a-3	947.30	0.00	76.63	1019	0.00	785.9	463	0.00	409.66	1286	0.00	1494.26	542
cr40x5b-1	1052.04	0.00	3115.92	21471	0.00	27430.7	12839	0.00	15757.00	24688	0.00	38343.40	14040
cr40x5b-2	981.54	0.00	7.61	25	0.00	104.38	45	0.00	46.77	1011	0.00	91.69	85
cr40x5b-3	964.33	0.00	12.33	20	0.00	50.08	31	0.00	24.30	74	0.00	54.01	49
Average		0.00	288.91	22223	0.00	2546.04	1273	0.00	1431.00	2616.25	0.00	4156.48	1515
Instances solved		12			12			12			12		

Table 12: Gaps and CPU times after 12 hours on instances of set $\mathcal{S}_3$

Instance	z^*	VFF2			VFF3			CFF2			CFF3		
		gap (%)	t (s)	nodes	gap (%)	t (s)	nodes	gap (%)	t (s)	nodes	gap (%)	t (s)	nodes
P111112	14676.9	2.30	44319.80	3604	4.16	43338.20	37	2.49	43177.00	1535	4.65	43290.90	34
P111212	13948	2.20	44120.90	4199	5.08	43293.90	44	2.56	43167.00	1199	4.56	43392.10	66
P112112	11671.6	0.20	43745.40	13454	1.34	43449.90	23	0.29	43168.60	3934	0.81	43532.30	121
P112212	7916.6	0.00	4980.13	8747	0.23	43269.20	1035	0.05	43169.10	16394	0.33	43339.20	363
P113112	12454.5	3.45	43574.00	8057	4.14	43415.10	19	3.69	43167.00	3276	3.38	43576.00	135
P113212	9022.6	0.00	247.09	33	0.00	8247.08	45	0.00	1634.29	139	0.00	13705.40	206
Average		1.36	30164.55	6349	2.49	37502.23	2001	1.51	36247.17	4413	2.29	38472.65	154
Instances solved		2			1			1			1		

Table 13: Gaps and CPU times after 12 hours on instances of set $\mathcal{S}_4$

Family	# instances	BBPPW		VFF2	
		# solved	avg. gap	# solved	avg. gap
$\mathcal{S}_1$ small	10	8	0.00	10	0.00[†]
$\mathcal{S}_1$ large	7	0	2.84	1	1.63[†]
$\mathcal{S}_2$ small	12	6	0.70	6	0.55
$\mathcal{S}_2$ large[‡]	12	0	–	1	1.89
$\mathcal{S}_3$ all	12	12	0.00	12	0.00
$\mathcal{S}_4$ all[‡]	6	0	–	2	1.63
Total	59	26		32	

[†] Including only instances reported also in Belenguer et al. [6].

[‡] Instances not reported in Belenguer et al. [6].

Table 14: Overall results comparison on branch-and-cut algorithms

Family	# instances	BMW	FF
		# solved	# solved
$\mathcal{S}_1$ small	10	10	10
$\mathcal{S}_1$ large	7	5	1
$\mathcal{S}_2$ small	12	12	8
$\mathcal{S}_2$ large	12	6	1
$\mathcal{S}_3$ all	12	12	12
$\mathcal{S}_4$ all	6	5	2
Total	59	50	34

Table 15: Overall results comparison against method of Baldacci et al. [4]

References

[1] Z. Akca, R. T. Berger, and T. K. Ralphs. Modeling and solving location, routing, and scheduling problems. In *Proceedings of the Eleventh INFORMS Computing Society Meeting*, pages 309–330, Charleston, South Carolina, USA, 2009.

[2] D. L. Applegate, R. E. Bixby, V. Chvátal, and W. J. Cook. *The traveling salesman problem: a computational study*. Princeton University Press, 2007.

[3] R. Baldacci, E. Hadjiconstantinou, and A. Mingozzi. An exact algorithm for the capacitated vehicle routing problem based on a two-commodity network flow formulation. *Operations Research*, 52(5):723–738, 2004.

[4] R. Baldacci, A. Mingozzi, and R. Wolfler-Calvo. An exact method for the capacitated location-routing problem. *Operations Research*, 2010. Forthcoming.

[5] S. Barreto. *Análise e Modelização de Problemas de localização-distribuição*. PhD thesis, University of Aveiro, Campus Universitário de Santiago, 3810-193 Aveiro, Portugal. In Portuguese., 2004.

[6] J. M. Belenguer, E. Benavent, C. Prins, C. Prodhon, and R. Wolfler-Calvo. A branch-and-cut algorithm for the capacitated location routing problem. *Computers & Operations Research*, 38:931–941, 2010.

[7] U. Blasum and W. Hochstättler. Application of the branch and cut method to the vehicle routing problem. Technical report, Zentrum fur Angewandte Informatik Köln, Germany, 2002.

[8] V. Chvátal. Edmonds polytopes and weakly Hamiltonian graphs. *Mathematical Programming*, 5:29–40, 1973.

[9] V. Chvátal, D. Applegate, R. Bixby, and W. Cook. Concorde: a code for solving traveling salesman problems, 1999.

[10] R. E. Gomory and T. C. Hu. Multi-terminal network flows. *Journal of SIAM*, 9(4): 551–570, 1961.

[11] M. Grötschel and M. W. Padberg. On the symmetric travelling salesman problem I: inequalities. *Mathematical Programming*, 16:265–280, 1979.

[12] Z. Gu, G. L. Nemhauser, and M. W. P. Savelsbergh. Lifted cover inequalities for 0-1 integer programs: computation. *INFORMS Journal on Computing*, 10(4):427–437, 1998.

[13] P. H. Hansen, B. Hegedahl, S. Hjortkjaer, and B. Obel. A heuristic solution to the warehouse location-routing problem. *European Journal of Operational Research*, 76: 111–127, 1994.

[14] G. Laporte and Y. Nobert. Comb inequalities for the vehicle routing problem. *Methods of Operations Research*, 51:271–276, 1984.

[15] G. Laporte, Y. Nobert, and D. Arpin. An exact algorithm for solving a capacitated location-routing problem. *Annals of Operations Research*, 6:293–310, 1986.

[16] C. K. Y. Lin, C. K. Chow, and A. Chen. A location-routing-loading problem for bill delivery services. *Computers & Operations Research*, 43:5–25, 2002.

[17] S. C. Liu and C. C. Lin. A heuristic method for the combined location routing and inventory problem. *International Journal of Advanced Manufacturing Technologies*, 26: 372–381, 2005.

[18] J. Lysgaard, A. N. Letchford, and R. W. Eglese. A new branch-and-cut algorithm for the capacitated vehicle routing problem. *Mathematical Programming*, 100:423–445, 2004.

[19] S. Martello, D. Pisinger, and P. Toth. Dynamic programming and strong bounds for the 0-1 knapsack problem. *Management Science*, 45:414–424, March 1999.

[20] S. T. McCormick, M. R. Rao, and G. Rinaldi. Easy and difficult objective functions for max cut. *Mathematical Programming Series B*, 94:459–466, 2003.

[21] Z. Ozyurt and D. Aksen. Solving multi-depot location-routing problem with time windows using lagrangian relaxation. In *ODYSSEUS 2006. Third International Workshop on Freight Transportation and Logistics*, Altea, Spain, 2006.

[22] J. Perl and M. S. Daskin. A warehouse location-routing problem. *Transportation Research B*, 19B:381–396, 1985.

[23] C. Prins, C. Prodhon, and R. Wolfer Calvo. A memetic algorithm with population management (ma|pm) for the capacitated location-routing problem. In J. Gottlieb and G. R. Raidl, editors, *EvoCOP 2006, LNCS 3906*, pages 183–194. Springer-Verlag Berlin Heidelberg 2006, 2006.

[24] D. Tuzun and L. I. Burke. A two-phase tabu search approach to the location routing problem. *European Journal of Operational Research*, 116:87–99, 1999.

[25] T.-H. Wu, C. Low, and J.-W. Bai. Heuristic solutions to multi-depot location-routing problems. *Computers & Operations Research*, 29:1393–1415, 2002.

A Proofs of lemmas and propositions

Proof of Proposition 2.1 It is direct to check that inequalities (24)-(25) imply the following inequalities

$$Qw_{ji} \leq (Q - d_j)(w_{ij} + w_{ji}) \qquad \{i,j\} \in \overline{E} \tag{63}$$

$$Qw_{ij} \geq d_j(w_{ij} + w_{ji}) \qquad \{i,j\} \in \overline{E}. \tag{64}$$

By adding identities (19) for customers $j \in S$ and after reducing we obtain

$$w(\delta^-(S)) + 2\sum_{j\in S} d_j y(I:\{j\}) = w(\delta^+(S)) + 2d(S)$$

By adding $w(\delta^+(S))$ at both sides of the identity above and after using identities (22) we obtain at the left-hand side $Qx(\delta(S)) + 2\sum_{j\in S} d_j y(I:\{j\})$. The desired right-hand side is obtained after using constraint (64) for $w(\delta^+(S))$. □

Proof of Proposition 2.2 It is easy to see that the (DFI) imply the following inequalities

$$Qw^i_{jh} \le (Q - d_j)(w^i_{hj} + w^i_{jh}) \qquad i \in I, \{h,j\} \in \overline{E} \tag{65}$$
$$Qw^i_{hj} \ge d_j(w^i_{hj} + w^i_{jh}) \qquad i \in I, \{h,j\} \in \overline{E} \tag{66}$$

By adding the flow conservation equations (27) for customers $j \in S$ and facilities $i \in I \setminus H$ we obtain

$$w^{I\setminus H}(\delta^-(S)) + 2\sum_{j\in S} d_j y(I\setminus H:\{j\}) = w^{I\setminus H}(\delta^+(S)) + 2\sum_{i\in I\setminus H}\sum_{j\in S} d_j u_{ij}$$

By adding $w^{I\setminus H}(\delta^+(S))$ at both sides of the above identity its left-hand turns to be equal to $Qx^{I\setminus H}(\delta(S)) + 2\sum_{j\in S} d_j y(I\setminus H:\{j\})$. For the right-hand size, we make use of the inequalities (65)-(66) in order to get $w^{I\setminus H}(\delta^+(S)) \ge \sum_{\substack{h\in S\\ j\notin S}} d_j x^{I\setminus H}_{hj}$. □

Proof of Proposition 3.1 if $x^i(F) < |F|$ then the constraint is trivially satisfied. If $x^i(F) = |F|$, then all the edges of F are used by vehicles linked to facility i. Since $|F|$ is odd, it follows that at least one edge, also linked to facility i, must be used in $\delta(S)\setminus F$. □

Proof of Proposition 3.2 if $\sum_{j\in S} u_{ij} = t$, then exactly t customers in S are served from facility i. For those customers, say S', given that $d(S') \le Q$, and given that the triangular inequality holds between distances, then the customers in S' must be served all by the same vehicle. Indeed, if more than one vehicle serves S', then it is always possible to serve them at lower cost by a single vehicle. □

Proof of Proposition 3.3 If $y(I\setminus I' : S') = y(I\setminus I' : S'') = |S''|$ then $x((I\setminus I')\cup\overline{S} : S) = x((I\setminus I')\cup(\overline{S\setminus S''}) : S\setminus S'')$, and then

$$\begin{aligned} x((I\setminus I')\cup\overline{S} : S) + 2y(I\setminus I' : S\setminus S') &= x((I\setminus I')\cup(\overline{S\setminus S''}) : S\setminus S'') + 2y(I\setminus I' : S\setminus S'') \\ &\ge r(S\setminus S'', I') \\ &\ge r(S\setminus S', I') \\ &= r(S, I'). \end{aligned}$$

□

Proof of Proposition 3.4 Let $S'' \subseteq S'$ such that $y(I \setminus I' : S') = y(I \setminus I' : S'') = |S''|$. This means that customer set S'' is served by single vehicles from $I \setminus I'$. Thus, $x((I \setminus I') \cup \overline{S} : S) = x((I \setminus I') \cup (\overline{S \setminus S''}) : S \setminus S'')$ so

$$\begin{aligned} x((I \setminus I') \cup \overline{S} : S) + 2y(I \setminus I' : S \setminus S') &= x((I \setminus I') \cup (\overline{S \setminus S''}) : S \setminus S'') + 2y(I \setminus I' : S \setminus S'') \\ &\geq 2r(S \setminus S'', I' \setminus \{i\}) + 2z_i(r(S \setminus S'', I') - r(S \setminus S'', I' \setminus \{i\})) \\ &\geq 2r(S \setminus S', I' \setminus \{i\}) + 2z_i(r(S \setminus S', I') - r(S \setminus S', I' \setminus \{i\})) \\ &= 2r(S, I' \setminus \{i\}) + 2z_i(r(S, I') - r(S, I' \setminus \{i\})). \end{aligned}$$

□

Proof of Proposition 3.5 Let us consider the SFCI and ESFCI in their weaker version that does not consider the subsets S'. These constraints can be written using the degree constraints as $x(E(S)) + \frac{1}{2}x(I' : S) + y(I' : S) \leq |S| - r(S, I')$ (for the SFCI) and $x(E(S)) + \frac{1}{2}x(I' : S) + y(I' : S) \leq |S| - r(S, I' \setminus \{i\}) + z_i(r(S, I' \setminus \{i\}) - r(S, I'))$ (for the ESFCI). We have

$$\begin{aligned} 2\alpha\overline{x} &\leq \sum_{u \in H} \overline{x}(\delta(u)) + \sum_{k=1}^{2} \sum_{j=1}^{s_k} (\overline{x}(E(T_j^k)) + \overline{x}(E(T_j^k \setminus H)) + \overline{x}(E(T_j^k \cap H))) \\ &\leq 2|H| + \sum_{k=1}^{2} \sum_{j=1}^{s_k} (\overline{x}(E(T_j^k)) + \overline{x}(E(T_j^k \setminus H)) + \overline{x}(E(T_j^k \cap H))). \end{aligned}$$

We now use the ESFCI in their inner form for $1 \leq j \leq s'_1$:

$$\begin{aligned} \overline{x}(E(T_j^1)) \leq & \frac{1}{2}x(I_j : S_j^1) + y(I_j : S_j^1) + |S_j^1| - r(S_j^1, I_j \setminus \{i_j\}) \\ & + z_{i_j}(r(S_j^1, I_j \setminus \{i_j\}) - r(S_j^1, I_j)) \\ \leq & \frac{1}{2}x(I_j : J) + y(I_j : J) + |S_j^1| - r(S_j^1, I_j \setminus \{i_j\}) \\ & + z_{i_j}(r(S_j^1, I_j \setminus \{i_j\}) - r(S_j^1, I_j)) \\ \overline{x}(E(T_j^1 \setminus H)) \leq & \frac{1}{2}x(I_j : S_j^1 \setminus H) + y(I_j : S_j^1 \setminus H) + |S_j^1 \setminus H| - r(S_j^1 \setminus H, I_j \setminus \{i_j\}) \\ & + z_{i_j}(r(S_j^1 \setminus H, I_j \setminus \{i_j\}) - r(S_j^1 \setminus H, I_j)) \\ \leq & \frac{1}{2}x(I_j : J) + y(I_j : J) + |S_j^1 \setminus H| - r(S_j^1 \setminus H, I_j \setminus \{i_j\}) \\ & + z_{i_j}(r(S_j^1 \setminus H, I_j \setminus \{i_j\}) - r(S_j^1 \setminus H, I_j)) \\ \overline{x}(E(T_j^1 \cap H)) \leq & |S_j^1 \cap H| - r(S_j^1 \cap H) \end{aligned}$$

and then

$$\overline{x}(E(T_j^1)) + \overline{x}(E(T_j^1 \setminus H)) + \overline{x}(E(T_j^1 \cap H)) \leq x(I_j : J) + 2y(I_j : J) + 2|S_j^1| + z_{i_j}\Lambda(H, T_j^1) - \widehat{r}(H, T_j^1).$$

For $s'_1 < j \leq s_1$ we do a similar development obtaining

$$\overline{x}(E(T_j^1)) + \overline{x}(E(T_j^1 \setminus H)) + \overline{x}(E(T_j^1 \cap H)) \leq x(I_j : J) + 2y(I_j : J) + 2|S_j^1| - \widehat{r}(H, T_j^1).$$

For the remaining teeth we have

$$\overline{x}(E(T_j^2)) + \overline{x}(E(T_j^2 \setminus H)) + \overline{x}(E(T_j^2 \cap H)) \leq 2|S_j^2| - \widehat{r}(H, T_j^2).$$

Then, adding all these terms and bounding we obtain

$$2\alpha\overline{x} \leq 2|H| + \sum_{1\leq j\leq s_1} (x(I_j : J) + 2y(I_j : J)) + \sum_{1\leq j\leq s_1'} z_{i_j}\Lambda(H, T_j^1) + 2\sum_{k=1,2}\sum_{1\leq j\leq s_k} |S_j^k| - \widehat{r}(H, \Pi).$$

As $x(I_j : J) + 2y(I_j : J)$ is *even* for $1 \leq j \leq s_1$, $\Lambda(H, T_j^1)$ is *even* for $1 \leq j \leq s_1'$ and $\widehat{r}(H, \Pi)$ is *odd*, after dividing by 2 and rounding the result follows. □

Proof of Lemma 3.6 If $S \subseteq W_{I'}$ then $d(S \cup T) \leq b(I')$ and the result is implied by the SFCI. If $S \subseteq \overline{W}_{I'}$ then $x(E(S)) + \frac{1}{2}x(I' : S) + y(I' : S) \leq |S| - \frac{1}{Q}d(S) \leq |S| - \frac{1}{Q}(d(S \cup T) - b(I'))$. If $S = S_1 \cup S_2, S_1 = S \cap W_{I'}, S_2 = S \cap \overline{W}_{I'}$, then $x(E(S)) + x(I' : S) + y(I' : S) = \sum_{i=1,2} x(E(S_i)) + \frac{1}{2}x(I' : S_i) + y(I' : S_i) \leq |S_1| + |S_2| - \frac{1}{Q}(d(S_1 \cup T) - b(I') + d(S_2))$. □

Proof of Proposition 3.7 First, note that constraint (57) can be written, using the degree constraints, in the following equivalent form:

$$x(E(S)) + \tfrac{1}{2}x(I' : S) + y(I' : S) + \tfrac{1}{Q}\sum_{j\notin S} d_j\eta(I', S, j) \leq |S| - \tfrac{1}{Q}(d(S) - b(I')). \tag{67}$$

Let us decompose the set $\overline{S}$ into three subsets $\overline{S}_0 = \{j \in \overline{S} : \eta(I', S, j) = 0\}$, $\overline{S}_{1/2} = \{j \in \overline{S} : \eta(I', S, j) = 1/2\}$ and $\overline{S}_{1+} = \{j \in \overline{S} : \eta(I', S, j) \geq 1\}$. Using this for the summation in the left-hand side of the equation (67) we have

$$\sum_{j\notin S} d_j\eta(I', S, j) = \frac{1}{2}d(\overline{S}_{1/2}) + \sum_{j\in\overline{S}_{1+}} d_j\eta(I', S, j). \tag{68}$$

But now, the second term of this last expression can be decomposed and bounded above as follows:

$$\begin{aligned}\sum_{j\in\overline{S}_{1+}} d_j\eta(I', S, j) &= \sum_{j\in\overline{S}_{1+}} (d_j - Q)\eta(I', S, j) + Q\sum_{j\in\overline{S}_{1+}} \eta(I', S, j)\\ &\leq d(\overline{S}_{1+}) - Q|\overline{S}_{1+}| + Q\sum_{j\in\overline{S}_{1+}} \eta(I', S, j).\end{aligned}$$

Thus, the left-hand side of constraint (67) can be bounded above by

$$x(E(S)) + \tfrac{1}{2}x(I', S) + y(I', S) + \tfrac{1}{2Q}d(\overline{S}_{1/2}) + \tfrac{1}{Q}d(\overline{S}_{1+}) - |\overline{S}_{1+}| + \sum_{j\in\overline{S}_{1+}} \eta(I', S, j). \tag{69}$$

But now, we have

$$x(E(S)) + \tfrac{1}{2}x(I', S) + y(I', S) + \sum_{j\in\overline{S}_{1+}} \eta(I', S, j) \leq x(E(S\cup\overline{S}_{1+})) + \tfrac{1}{2}x(I', S\cup\overline{S}_{1+}) + y(I', S\cup\overline{S}_{1+}).$$

Using this, (69) can be bounded above by

$$x(E(S \cup \overline{S}_{1+})) + \tfrac{1}{2}x(I', S \cup \overline{S}_{1+}) + y(I', S \cup \overline{S}_{1+}) + \tfrac{1}{2Q}d(\overline{S}_{1/2}) + \tfrac{1}{Q}d(\overline{S}_{1+}) - |\overline{S}_{1+}|.$$

Now, as $\overline{S}_{1/2} \subseteq W_{I'}$ we can apply the lemma and thus this last expression can be bounded above by

$$|S \cup \overline{S}_{1+}| - \tfrac{1}{Q}(d(S \cup \overline{S}_{1+} \cup \overline{S}_{1/2}) - b(I')) + \tfrac{1}{2Q}d(\overline{S}_{1/2}) + \tfrac{1}{Q}d(\overline{S}_{1+}) - |\overline{S}_{1+}| \leq |S| - \tfrac{1}{Q}(d(S) - b(I')).$$

□

Proof of Lemma 4.1 Let $h, j \in J_S$ be such that $\omega^*_{hj} \geq 1$ or $[\phi^*_{ih} \geq 1$ and $\phi^*_{ij} \geq 1]$. Let $S \subseteq J_S$ be a customer set crossing $\{h, j\}$, i.e., $S \cap \{h, j\}, S \setminus \{h, j\}$ and $\{h, j\} \setminus S \neq \emptyset$. Without loss of generality we suppose that $j \in S, h \notin S$. We will show that $T = S \cup \{h\}$ produces a violation of value at least that of S. Let us define $\sigma_i(T) = \omega^*((I \setminus \{i\}) \cup \overline{T} : T) + 2\phi^*(I \setminus \{i\} : T)$. Because $r(S, \{i\}) \leq r(T, \{i\})$ it suffices to show that $\sigma_i(T) \leq \sigma_i(S)$. In fact

$$\begin{aligned}\sigma_i(T) - \sigma_i(S) =& [\omega^*(\delta(T)) + 2\phi^*(I : T)] - [\omega^*(\delta(S)) + 2\phi^*(I : S)] \\ &+ [\omega^*(i : S) - \omega^*(i : T)] + 2[\phi^*(i : S) - \phi(i : T)] \\ =& [\omega^*(\delta(T)) + 2\phi^*(I : T)] - [\omega^*(\delta(S)) + 2\phi^*(I : S)] - [\omega^*_{ih} + 2\phi^*_{ih}].\end{aligned}$$

The submodularity of the cut function implies

$$[\omega^*(\delta(T)) + 2\phi^*(I : T)] - [\omega^*(\delta(S)) + 2\phi^*(I : S)] \leq [\omega^*(\delta(\{h, j\})) + 2\phi^*(I : \{h, j\})] - [\omega^*(\delta(j)) + 2\phi^*(I : j)]$$

and then

$$\begin{aligned}\sigma_i(T) - \sigma_i(S) &\leq \omega^*(\delta(h)) + 2\phi^*(I : h) - 2\omega^*_{hj} - (\omega^*_{ih} + 2\phi^*_{ih}) \\ &\leq 2 - 2\omega^*_{hj} - (\omega^*_{ih} + 2\phi^*_{ih}).\end{aligned}$$

The result follows by applying the shrinking hypothesis. □

Proof of Lemma 4.2 Let $T \subseteq J_S$ and $h \in T$ be such that $\phi^*_{ih} = 1$, $d^*_h \leq Q$. Let us denote $S = T \setminus \{h\}$. Because h is linked only to facility i, we have $\omega^*((I \setminus \{l\}) \cup \overline{S} : S) = \omega^*((I \setminus \{l\}) \cup \overline{T} : T)$. We also have $\phi^*(I \setminus \{l\} : S) = \phi^*(I \setminus \{l\} : T) - 1$. It follows that $\omega^*((I \setminus \{l\}) \cup \overline{S} : S) + 2\phi^*(I \setminus \{l\} : S) = \omega^*((I \setminus \{l\}) \cup \overline{T} : T) + 2\phi^*(I \setminus \{l\} : T) - 2$. If T and k violate a BFCI then $\omega^*((I \setminus \{l\}) \cup \overline{T} : T) + 2\phi^*(I \setminus \{l\} : T) < 2r(T, \{l\}) \leq 2(r(S, \{l\}) + 1)$ and the result follows. □

Proof of Lemma 4.4 S_1, S_2 are not connected between them nor with facility i, i.e., $x^*(S_1 : S_2) = x^*(i : S_2) = y^*(i : S_2) = 0$. Suppose that (i, S) defines a violated BFCI, i.e.,

$$x^*((I \setminus \{i\} : \overline{S} : S) + 2y^*(I \setminus \{i\} : S) < 2r(S, \{i\}).$$

But given that S_1 and S_2 lie in different connected components we have

$$x^*((I \setminus \{i\} : \overline{S} : S) + 2y^*(I \setminus \{i\} : S) = x^*(\delta(S_2)) + 2y^*(I : S_2) \\ + x^*((I \setminus \{i\}) \cup \overline{S}_1 : S_1) + 2y^*(I \setminus \{i\} : S_1).$$

Joining both relationships and taking into account that S_2 satisfies the CC we have

$$\begin{aligned} x^*((I \setminus \{i\}) \cup \overline{S}_1 : S_1) + 2y^*(I \setminus \{i\} : S_1) &< 2r(S, \{i\}) - [x^*(\delta(S_2)) + 2y^*(I : S_2)] \\ &\leq 2r(S, \{i\}) - 2r(S_2) \\ &\leq 2r(S_1, \{i\}) \end{aligned}$$

and the result follows. □

Proof of Proposition 4.5 Let $S \subseteq J_S$ be a customer set in the shrunk graph *crossing* the set $\{u, v\}$, i.e., $S \cap \{u, v\}, S \setminus \{u, v\}, \{u, v\} \setminus S \neq \emptyset$. Without loss of generality, we suppose that $u \in S, v \notin S$. We will show that the set $T = S \cup \{v\}$ induces a violation of value at least the same as that induced by S. First note that if u or v take the role of nodes h or j in inequality (6) then it will not be violated. As a consequence of this, nodes that can take the place of h or j are among those that have not been shrunk. Let us compute the left-hand side of inequality (6) for S and T, that we denote as $\alpha(S)$ and $\alpha(T)$, respectively, and see that they satisfy the following relationship:

$$\begin{aligned} \alpha(T) &= \alpha(S) + \omega^*(\delta(v)) - 2\omega^*(v : S) \\ &\leq \alpha(S). \end{aligned}$$

As the right hand side of the inequality is the same for both S and T, the violation incurred by set T is bigger than that of S and the result follows. □