The Capacitated Mobile Facility Location Problem

S. Raghavan

Smith School of Business and
Institute for Systems Research
University of Maryland
raghavan@umd.edu

Mustafa Sahin

Smith School of Business
University of Maryland
mustafa.sahin@rhsmith.umd.edu

F. Sibel Salman

Industrial Engineering Department
Koc University
ssalman@ku.edu.tr

Abstract

The capacitated mobile facility location problem (CMFLP) arises in logistics planning of community outreach programs delivered via mobile facilities. It adds capacity restrictions to the mobile facility location problem introduced previously by Demaine et al. [2009], thereby extending the problem to a practical setting. In the problem, one seeks to relocate (or move) a set of existing facilities and assign clients to these facilities while respecting capacities so that the weighted sum of facility movement costs and the client travel costs (each to its assigned facility) is minimized. We provide two integer programming formulations for the CMFLP. The first is on a layered graph, while the second is a set partitioning formulation. We prove that the linear relaxation of the set partitioning formulation provides a tighter lower bound to the CMFLP than the linear relaxation of the layered graph formulation. We then develop a branch-and-price algorithm on the set partitioning formulation. We find that the branch-and-price procedure is particularly effective both in terms of solution quality and running time, when the ratio of the number of clients to the number of facilities is small and the facility capacities are tight. Finally, we present two heuristic approaches for the CMFLP. One is a LP rounding heuristic, and the other is based on a natural problem decomposition on the layered graph.
Keywords: mobile facilities, capacitated facility location, column generation, branch-and-price, local search.

1 Introduction

The capacitated mobile facility location problem (CMFLP) is defined on a network where clients and facilities are initially located at vertices on the network. Associated with each client is a

demand and each facility has a specified capacity available to service demand. A destination vertex must be determined for each facility and each client should be assigned to one of the facilities so that the total demand of the clients assigned to a facility respects the capacity. The objective is to minimize the total weighted distance traveled by the facilities and the clients.

Formally, the CMFLP is set on a graph $G = (V, E)$ where V denotes the set of vertices and E denotes the set of edges. A non-negative distance d_{ij} is defined for each edge $(i, j) \in E$. We interchangeably use cost and distance henceforth to indicate d_{ij}. The initial locations of the clients are represented by the subset $C \subseteq V$. Each client $i \in C$ has demand q_i and a positive weight u_i. There are different types of facilities with differing capacities. Each facility is of a type from the set T and the subset $F = \bigcup_{t \in T} F_t \subseteq V$ of vertices denotes the initial locations of the facilities (so F_t denotes the set of initial locations of facilities of type t). Each facility $j \in F_t$ has capacity Q_t and a positive weight w_j for relocation. All facilities are equipped with the same capabilities and therefore a client can get service from any one of them as long as the capacity restrictions are satisfied. In a feasible solution to the CMFLP, each facility $j \in F$ moves to a destination vertex $v(j) \in V$ and each client $i \in C$ moves to a destination vertex $v(i) \in V$ with the condition that $v(i) = v(j)$ for some j. A facility cannot share a destination vertex with another facility and a client can only be served by a single facility, i.e. demand cannot be split. Total demand assigned to a type t facility cannot exceed Q_t, for all $t \in T$. Clients or facilities may stay put (i.e., have their destination equal to their origin). Clients and facilities are also permitted to start at the same vertex. The objective is to minimize the total weighted distance traveled by the facilities and the clients, that is, $\sum_{j \in F} w_j d_{j,v(j)} + \sum_{i \in C} u_i d_{i,v(i)}$.

By including the capacity restrictions, the CMFLP extends the mobile facility location problem (MFLP) introduced previously by Demaine et al. [2009] to a practical setting. The CMFLP finds applications in logistics planning of community outreach programs delivered via mobile facilities. Examples of community outreach programs that utilize mobile facilities include library outreach programs in rural areas, mobile daycare delivered to farm children, and mobile schools that provide basic education to street children, as well as temporary schools servicing refugee camps. The deployment of mobile healthcare facilities (e.g. cancer screening units, blood banks, eye clinics, vaccination booths in case of a disease outbreak) that serve beneficiaries residing in either urban districts or rural regions is another important application area of the CMFLP. In these applications, districts (population centers) that have patients residing in them are represented by client vertices in the CMFLP. Mobile medical facilities currently located at some of the districts are represented by facility vertices. The demand of a district shows the number of patients and their demands (i.e., visits to the medical facility) in the district and the capacity of a facility is the total number of patient visits it can handle within a time frame. Weights may be assigned to facilities and client locations according to priority, patient criticality, number of patient visits, etc. The objective of the problem is to move the mobile facilities so that every patient is served and the total weighted distance traveled by the facilities and the patients is minimized. After demand is served in an area or demand patterns have significantly changed, facilities may be relocated to a new area. The

facility destinations in the previous network will be the originating facility vertices in the current network. Then, the problem can be solved with new clients and their respective demands.

The importance of mobile facilities is noted both in the medical and the operations research communities. Geoffroy et al. [2014] discuss the benefits of mobile healthcare facilities as a complementary service to fixed clinics by expanding access to healthcare for hard-to-reach areas. It is well-known that ease of geographical access to a healthcare facility has a major impact on the likelihood of participation in preventive healthcare services [see Weiss et al., 1971]. Bingham et al. [2003] investigated factors affecting the utilization of preventive services for cervical cancer and found the screening rates to be much lower in areas where services are distant or difficult to access. They reported greater transportation cost and distance as the main reasons for low participation rates. These examples motivate the use of the weighted distance objective in the CMFLP.

Studies addressing location decisions for healthcare facilities focus mainly on fixed clinics and hospitals, and typically aim to maximize coverage of demand locations. For example, Verter and Lapierre [2002] model the preventive healthcare facility location problem as an extension of the Maximal Coverage Location Problem. Doerner et al. [2007] study a tour planning problem for a single mobile healthcare facility with criteria concerned with the number of stops and tour length, and the distance to the nearest tour stop. Ha et al. [2013] discuss applications of the multi-vehicle covering tour problem related to deployment of mobile healthcare teams and mobile library teams and the distribution of relief items after a disaster. The problem involves choosing the stops of the vehicles from a set of potential locations so that every person can reach one of these stops within an acceptable time limit. The CMFLP differs from these studies significantly as it addresses capacity limitations of the facilities while minimizing the total distances traveled by both the facilities and the clients.

Our Contributions: In this paper, we develop exact and heuristic algorithms to solve the CMFLP. We first compare two (linear) integer programming (IP) formulations for the CMFLP. The first formulation, which we call the *layered graph* formulation, extends the one given in Halper et al. [2015] for the MFLP to account for the capacity constraints. The second formulation is a *set partitioning* formulation where each variable corresponds to a type of facility to be moved to a vertex in order to serve a feasible set of clients (i.e. the total demand of the clients cannot exceed the capacity). We prove that the LP relaxation of the set partitioning formulation provides a lower bound to the CMFLP that is greater than or equal to the LP relaxation bound from the layered graph formulation and can be strictly better. Next, we provide a branch-and-price algorithm for the set partitioning formulation where a column generation procedure is used on the set partitioning formulation to obtain lower bounds. Furthermore, we present two heuristic approaches for the CMFLP. The first is an LP rounding heuristic that is also used to obtain good quality upper bounds within the branch-and-price algorithm. The second is a local search heuristic called 1-OptSwapBI that is adapted from one of the local search heuristics described in Halper et al. [2015].

To show the efficacy of the branch-and-price algorithm and the underlying column generation procedure, we conducted computational tests on instances adapted from Halper et al. [2015] (where

each vertex hosts a client). We found out the ratio of the number of clients to the number of facilities plays an important role on the performance of both the branch-and-price algorithm and the heuristics. We solved the layered graph formulation using CPLEX as a benchmark. We observe that in general the problem is harder to solve when the average number of clients per facility is relatively small (i.e., the ratio of $|C|$ to $|F|$ is small). However, in these instances the branch-and-price algorithm outperforms the CPLEX benchmark. Furthermore, the local search heuristic complements the branch-and-price algorithm by obtaining good solutions quickly when the average number of clients per facility is larger.

The rest of the paper is organized as follows. Section 2 discusses related work in the literature. Section 3 describes two integer programming formulations. Section 4 explains the column generation procedure and the branch-and-price algorithm. Section 5 discusses the heuristics, and Section 6 presents our computational results. Section 7 provides concluding remarks.

2 Related Work

To the best of our knowledge the CMFLP has not been considered previously in the literature. Its uncapacitated version, the MFLP was introduced by Demaine et al. [2009] as one of a class of movement problems. The majority of the previous work on the MFLP deals with the approximability of the problem and mainly consists of deriving theoretical bounds [e.g., Friggstad and Salavatipour, 2011, Armon et al., 2012, Anari et al., 2015]. Halper et al. [2015] introduced an IP formulation for the MFLP and developed various local search heuristics based on a decomposition of the problem. Ahmadian et al. [2013] showed that the local search heuristic n-OptSwap introduced by Halper et al. [2015] is a $3 + O\left(\sqrt{\frac{\log\log n}{\log n}}\right)$-approximation algorithm for the MFLP.

The CMFLP concerns heterogenous facilities. When all facilities have identical capacities, the special case of CMFLP with homogeneous facilities is obtained. The CMFLP with homogeneous facilities generalizes the well-studied capacitated p-median with single sourcing problem (CPMSP), in which facilities are not relocated from their initial locations, but their locations are to be determined. We can easily see that by setting the cost of moving each facility to zero in the CMFLP with homogeneous facilities, the CPMSP is obtained. A recent paper by Stefanello et al. [2015] provides a nice discussion of earlier work on this problem. They also develop a matheuristic that solves large scale CPMSP instances (with up to 4500 nodes and 1000 facilities) and obtain small optimality gaps within an hour of computation time. Their heuristic approach mainly relies on eliminating variables iteratively from the mathematical model.

In the single source capacitated facility location problem (SCFLP), an opening cost is associated with each facility instead of specifying the number of facilities. Guastaroba and Speranza [2014], Yang et al. [2012], Cortinhal and Captivo [2003], Chen and Ting [2008], Holmberg et al. [1999], and Ahuja et al. [2004], among others, propose solution methods for this problem. Guastaroba and Speranza [2014] develop a kernel search algorithm and achieve near optimal solutions for large scale instances (with up to 1500 nodes and 300 potential facility locations, as well as 1000 nodes and 1000

potential facility locations). Klose [1999] and Tragantalerngsak et al. [2000] study an extension of the SCFLP by considering two echelons of facilities. Each second-echelon facility can be supplied by only one first-echelon facility, and each customer is serviced by only one second-echelon facility. While only the locations of the second-echelon facilities are selected in Klose [1999], the locations of first-echelon facilities are also selected in Tragantalerngsak et al. [2000]. Similarly, two sets of facilities (intermediate and upper level) are located in Addis et al. [2012] and Addis et al. [2013], but the capacity of intermediate level facilities should also be determined by installing devices that provide different capacities at different costs. All upper level facilities have the same given capacity. The objective includes the cost of assigning clients to intermediate level facilities, and of intermediate level facilities to upper level facilities, in addition to the cost of locating facilities. To solve the two-level problem, Addis et al. [2012] propose a branch-and-price algorithm.

In dynamic facility location problems, facilities are relocated over a time horizon consisting of multiple periods [see Arabani and Farahani, 2012, Nickel and Saldanha da Gama, 2015, for overviews of studies on such problems]. Most of the existing multi-period location problems associate a fixed cost for opening and closing facilities or resizing the capacities that depends on the location of the facility. For instance, Torres-Soto and Uster [2011] develop exact solution methods for capacitated multi-period relocation problems with fixed relocation costs, where demand of a customer can be serviced by multiple facilities partially. On the other hand, Melo et al. [2006] include a unit variable cost of moving capacity from an existing facility to a new facility, in addition to the fixed opening and closing costs. In their model, relocation decisions are constrained by budget limitations, and the objective includes production/supply costs, transportation costs between facilities, inventory holding costs, and fixed facility operating costs. To the best of our knowledge, none of the existing dynamic facility location models consider a fixed cost of relocating a facility that depends on the initial and destination locations, as in the CMFLP.

Column generation and branch-and-price approaches have been widely used in the literature to solve the CPMSP and the SCFLP. Lorena and Senne [2004] implement a column generation approach to solve the LP relaxation of the set covering formulation of the CPMSP. The new columns are generated by solving a 0-1 knapsack problem for pricing and a Lagrangean/surrogate relaxation identified from the dual of the master problem to accelerate convergence. The relaxation also provides lower bounds. Ceselli and Righini [2005] describe a branch-and-price algorithm that uses column generation for the CPMSP. At each iteration of column generation, the current values of the dual variables are used as Lagrangian multipliers to compute a lower bound as in Lorena and Senne [2004]. The authors experiment with two branching strategies and computational experiments suggest that the performance of the branch-and-price algorithm is closely related to ratio of the number of clients ($|C|$) to the number of facilities $|F|$. Klose and Görtz [2007] describe a column generation and branch-and-price algorithm for the SCFLP. The method is based on a Lagrangean relaxation of the demand constraints and a stabilized column generation method for solving the corresponding master problem to optimality.

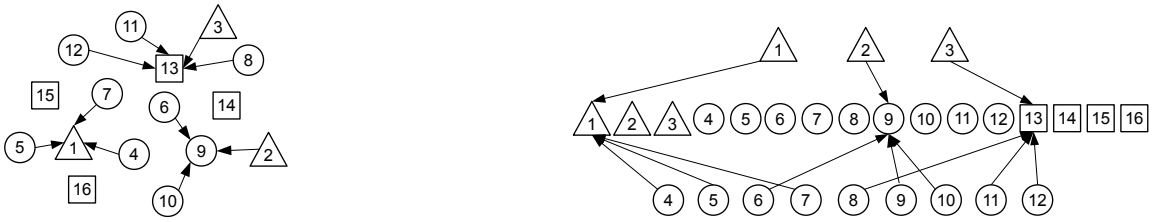

Figure 1: The original graph representation (left) and the layered graph representation (right) of a solution to an instance of the CMFLP. Triangles denote the facility vertices, circles denote the client vertices and squares denote the remaining vertices.

3 Integer Programming Formulations

We present two IP formulations for the CMFLP. The first one is the capacitated version of the formulation in Halper et al. [2015], which we refer to as the layered graph formulation. We describe a decomposition based on this formulation which also helps the development of a local search algorithm. The second formulation is a set partitioning formulation for which we describe a branch-and-price algorithm where the variables in the layered graph formulation are used for branching and the LP relaxation is solved via a column generation procedure.

3.1 Layered Graph Formulation

An instance of the CMFLP can be represented in a graph with three layers. After making copies of the client vertices C and the facility vertices F, the copies of the facility vertices make up the first layer. The vertex set V makes up the second layer and the copies of the client vertices make up the last layer. Figure 1 shows an example of the transformation from the original graph to the layered graph representation. The layered graph representation aids visualizing the formulation and the decomposition technique described next.

We define a binary variable x_{iv} for each $i \in C$ and $v \in V$, and a binary variable y_{jv} for each $j \in F$ and $v \in V$. Let $x_{iv} = 1$, if the destination of client i is vertex v; and $x_{iv} = 0$, otherwise. Similarly, let $y_{jv} = 1$, if the destination of facility j is vertex v; and $y_{jv} = 0$, otherwise. For each vertex $v \in V$ and each facility type $t \in T$, we define a binary variable z_{tv} such that $z_{tv} = 1$, if vertex v is the destination of some facility of type t; and $z_{tv} = 0$, otherwise. The CMFLP is formulated as follows:

$$\text{(IP1) Minimize} \quad \sum_{i \in C} \sum_{v \in V} u_i d_{iv} x_{iv} + \sum_{j \in F} \sum_{v \in V} w_j d_{jv} y_{jv} \tag{1}$$

$$\text{subject to} \quad \sum_{v \in V} x_{iv} = 1 \qquad \forall i \in C \tag{2}$$

$$\sum_{v \in V} y_{jv} = 1 \qquad \forall j \in F \tag{3}$$

$$\sum_{j \in F_t} y_{jv} = z_{tv} \qquad \forall v \in V,\ t \in T \tag{4}$$

$$\sum_{t \in T} z_{tv} \leq 1 \qquad \forall v \in V \tag{5}$$

$$x_{iv} \leq \sum_{t \in T} z_{tv} \qquad \forall i \in C,\ v \in V \tag{6}$$

$$\sum_{i \in C} q_i x_{iv} \leq \sum_{t \in T} Q_t z_{tv} \qquad \forall v \in V \tag{7}$$

$$z_{tv}, y_{jv}, x_{iv}, \in \{0, 1\}. \qquad \forall i \in C,\ j \in F,\ v \in V,\ t \in T \tag{8}$$

In IP1, the objective function (1) calculates the total weighted distance traveled by the facilities and clients. Constraints (2) and (3) ensure that each client and facility has a destination vertex. If $z_{tv} = 1$, constraints (4) and (5) specify that vertex v is the destination of a facility of type t and cannot host more than one facility. In the case that $z_{tv} = 0$, no facility of type t may have vertex v as its destination. Constraint (6) states that client i may travel to location v only if there is a facility moving to v. By constraint (7), total demand for a facility cannot exceed its capacity. Constraint (8) defines the binary variables. Note that by leaving the z_{tv} variables binary the y_{jv} variables can be relaxed in the interval $[0, 1]$, since constraints (3) and (4) correspond to the totally unimodular assignment constraints.

This formulation lends itself to a decomposition when the z_{tv} variables are fixed. Suppose we are given destination vertices for each facility type $t \in T$. Let Z_t denote the set of destination vertices for facilities in F_t (so $|F_t| = |Z_t|$). Let $Z = \bigcup_{t \in T} Z_t \subset V$. In other words $z_{tv} = 1$ for $v \in Z_t$, and $z_{tv} = 0$ for $V \backslash Z_t$. Then, the problem decomposes into a total of $|T| + 1$ disjoint subproblems of assigning each facility in F_t to a vertex in Z_t for every $t \in T$ (the $|T|$ facility assignment problems), and assigning each client to a vertex in Z (the client assignment problem).

In the facility assignment problems, the objective is to find a minimum cost bipartite matching between the initial facility locations in F_t and the destination locations in Z_t. For each facility type $t \in T$, if $z_{tv} = 0$, then constraint (4) implies $y_{jv} = 0$ for all $j \in F_t$ and $v \in V \setminus Z_t$. Then for $z_{tv} = 1$, constraint (4) can be rewritten as $\sum_{j \in F_t} y_{jv} = 1$ and $v \in Z_t$. Given a subset $Z_t \subset V$, the facility assignment problem (FA(Z_t, t)) can be formulated as,

$$\begin{aligned}
\text{FA}(Z_t, t) = \text{Minimize} \quad & \sum_{j \in F_t} \sum_{v \in Z_t} w_j d_{jv} y_{jv} & \\
\text{subject to} \quad & \sum_{v \in Z_t} y_{jv} = 1 & \forall j \in F_t \\
& \sum_{j \in F_t} y_{jv} = 1 & \forall v \in Z_t \\
& y_{jv} \geq 0 & \forall j \in F_t,\ v \in Z_t.
\end{aligned}$$

which models the least cost bipartite matching problem. Since the constraint matrix is totally unimodular, the integrality of y_{jv} is relaxed. The facility assignment problem can be solved in polynomial time via the Hungarian Algorithm [see Kuhn, 1955].

In the client assignment problem, the objective is to assign each client to one of the facility

destination locations in Z such that if $z_{tv} = 1$, then the facility assigned to location v serves a total demand of at most Q_t and the total weighted distance traveled by the clients is minimized. For $v \in V \setminus Z$, we have $\sum_{t \in T} z_{tv} = 0$ and constraint (6) implies that $x_{iv} = 0$ for $i \in C$ since a client cannot be assigned to a destination vertex without a facility. Therefore, constraint (6) can be rewritten as $x_{iv} \leq 1$ for $i \in C, v \in Z$, which is redundant. Given a subset $Z \subset V$, the client assignment problem (CA(Z)) is the well-studied generalized assignment problem (GAP) [see Cattrysse and Van Wassenhove, 1992] and can be formulated as,

$$
\begin{aligned}
\text{CA}(Z) = \text{Minimize} \quad & \sum_{i \in C} \sum_{v \in Z} u_i d_{iv} x_{iv} & \\
\text{Subject to} \quad & \sum_{v \in Z} x_{iv} = 1 & \forall i \in C \\
& \sum_{i \in C} q_i x_{iv} \leq Q_t & \forall t \in T,\ v \in Z_t \\
& x_{iv} \in \{0, 1\} & \forall i \in C,\ v \in Z,
\end{aligned}
$$

A similar decomposition technique was described in Halper et al. [2015] for the MFLP, which has identical facilities without capacity restrictions by design, i.e., $|T| = 1$ and $Q_t = \infty$. The decomposition in Halper et al. [2015] is extended here to the CMFLP, such that the facility assignment problem is solved separately for each facility type $t \in T$. In the client assignment problem described for the MFLP, each client is assigned to its closest facility while in the CMFLP, the client assignment problem is the GAP which is NP-Hard.

3.2 Set Partitioning Formulation

Let $\mathcal{S}_{tv}$ denote the set of all feasible client assignments to a facility of type t to be located in v. A client assignment S_{tv} is feasible if $\sum_{i \in S_{tv}} q_i \leq Q_t$. Let $a_{iS_{tv}}$ be a binary coefficient taking the value 1 if client i appears in assignment S_{tv}, and 0, otherwise. For an assignment $S_{tv} \in \mathcal{S}_{tv}$, $d_{S_{tv}}$ denotes the total weighted travel cost of clients in S_{tv}. That is, $d_{S_{tv}} = \sum_{i \in S_{tv}} u_i d_{iv}$. Furthermore, for all $S_{tv} \in \mathcal{S}_{tv}$, let $\pi_{S_{tv}}$ be a binary variable indicating if customers in S_{tv} are assigned to the facility of type t that will be located at v. Let y_{jv} be a binary variable indicating if the facility j is moved to v. The set partitioning formulation is as follows.

$$
\begin{aligned}
\text{(IP2) Minimize} \quad & \sum_{t \in T} \sum_{v \in V} \sum_{S_{tv} \in \mathcal{S}_{tv}} d_{S_{tv}} \pi_{S_{tv}} + \sum_{j \in F} \sum_{v \in V} w_j d_{jv} y_{jv} & & (9) \\
\text{subject to} \quad & \sum_{v \in V} \sum_{S_{tv} \in \mathcal{S}_{tv}} \pi_{S_{tv}} = |F_t| & \forall t \in T & (10) \\
& \sum_{t \in T} \sum_{v \in V} \sum_{S_{tv} \in \mathcal{S}_{tv}} a_{iS_{tv}} \pi_{S_{tv}} = 1 & \forall i \in C & (11) \\
& \sum_{t \in T} \sum_{S_{tv} \in \mathcal{S}_{tv}} \pi_{S_{tv}} \leq 1 & \forall v \in V & (12)
\end{aligned}
$$

$$\sum_{v \in V} y_{jv} = 1 \qquad \forall j \in F \tag{13}$$

$$\sum_{j \in F_t} y_{jv} = \sum_{S_{tv} \in \mathcal{S}_{tv}} \pi_{S_{tv}} \qquad \forall v \in V,\ t \in T \tag{14}$$

$$\pi_{S_{tv}} \in \{0,1\} \qquad \forall v \in V,\ t \in T,\ S_{tv} \in \mathcal{S}_{tv} \tag{15}$$

$$y_{jv} \in \{0,1\} \qquad \forall j \in F,\ v \in V \tag{16}$$

The objective function calculates the total weighted distance traveled by facilities and clients. Constraint (10) asserts that the total number of facilities moved for each type is equal to the total number of facilities of that type. Constraint (11) ensures that a client only appears in exactly one assignment. By constraint (12), at most one facility can be assigned to the same location. Similar to IP1, by setting $\pi_{S_{tv}}$ binary, the y_{jv} variables can be relaxed in the interval $[0,1]$ since constraints (13) and (14) correspond to the assignment constraints that are totally unimodular.

Theorem 3.1. *The optimal objective value of the LP relaxation of IP2 (namely, LP2) is greater than or equal to the optimal objective value of the LP relaxation of IP1 (namely, LP1).*

Proof. We first show that any feasible solution to LP2 can be transformed to a feasible solution of LP1 of equal cost. Let π and y be a feasible solution to LP2. Let

$$z_{tv} = \sum_{S_{tv} \in \mathcal{S}_{tv}} \pi_{S_{tv}} \qquad \forall t \in T,\ v \in V, \tag{17}$$

$$x_{iv} = \sum_{t \in T} \sum_{S_{tv} \in \mathcal{S}_{tv}} a_{iS_{tv}} \pi_{S_{tv}} \qquad \forall i \in C,\ v \in V \tag{18}$$

and y_{jv} indicates if the facility j is moved to location v in both formulations. After the transformation, constraints (2), (3), (4) and (5) are identical to constraints (11), (13), (14) and (12), respectively. From (17) and (18) we get

$$\begin{aligned} x_{iv} &= \sum_{t \in T} \sum_{S_{tv} \in \mathcal{S}_{tv}} a_{iS_{tv}} \pi_{S_{tv}}, \\ &\leq \sum_{t \in T} \sum_{S_{tv} \in \mathcal{S}_{tv}} \pi_{S_{tv}}, \\ &= \sum_{t \in T} z_{tv}, \end{aligned}$$

since $a_{iS_{tv}}$ is a 0-1 coefficient, which implies that constraint (6) holds.

By definition, all feasible assignments $S_{tv} \in \mathcal{S}_{tv}$ satisfy $\sum_{i \in S_{tv}} q_i \leq Q_t$ for all $t \in t, v \in V$, which can also be stated as $\sum_{i \in C} q_i a_{iS_{tv}} \leq Q_t$. By summing up each side for $t \in T$ and $S_{tv} \in \mathcal{S}_{tv}$ and multiplying each side by $\pi_{S_{tv}}$, a nonnegative term, we get

$$\sum_{i \in C} \sum_{t \in T} \sum_{S_{tv} \in \mathcal{S}_{tv}} q_i a_{iS_{tv}} \pi_{S_{tv}} \leq \sum_{t \in T} Q_t \sum_{S_{tv} \in \mathcal{S}_{tv}} \pi_{S_{tv}}.$$

After replacing the corresponding terms with x_{iv} and z_{tv}, we have

$$\sum_{i \in C} q_i x_{iv} \leq \sum_{t \in T} Q_t z_{tv},$$

which is constraint (7). Note that the lower and upper bound constraints for the variables are satisfied by the transformation. The second terms in the objective functions of both LP1 and LP2 are identical. Therefore, we focus on the first terms. In LP2, after replacing $d_{S_{tv}}$ with $\sum_{i \in S_{tv}} u_i d_{iv}$ in the first term, we get

$$\begin{aligned}
\sum_{t \in T} \sum_{v \in V} \sum_{S_{tv} \in \mathcal{S}_{tv}} d_{S_{tv}} \pi_{S_{tv}} &= \sum_{t \in T} \sum_{v \in V} \sum_{S_{tv} \in \mathcal{S}_{tv}} \sum_{i \in S_{tv}} u_i d_{iv} \pi_{S_{tv}}, \\
&= \sum_{t \in T} \sum_{v \in V} \sum_{S_{tv} \in \mathcal{S}_{tv}} \sum_{i \in C} u_i d_{iv} a_{iS_{tv}} \pi_{S_{tv}}, \\
&= \sum_{i \in C} \sum_{v \in V} u_i d_{iv} x_{iv},
\end{aligned}$$

which is the first term in the objective function of LP1. Therefore, a feasible solution to LP2 can be transformed into a feasible solution to LP1 of equal cost.

Now we provide an example where the objective value of the optimal solution of LP1 is strictly less than the objective value of the optimal solution of LP2. Consider the example in Figure 2 with 3 nodes, 2 identical facilities (i.e., $|T| = 1$), and 3 clients. Facilities 1 and 2 with a capacity of 5 are initially located at nodes 1 and 2, respectively. Clients 1, 2, and 3 with a demand of 1, 3, and 4 are at nodes 1, 2, and 3 respectively. Distances are given as $d_{11} = d_{22} = d_{33} = 0$, $d_{12} = d_{21} = d_{23} = d_{32} = 1$ and $d_{13} = d_{31} = 2$. Let $w_j = 1$ for all facilities and $u_i = 1$ for all clients. The optimal solution to LP1 is $y_{11} = y_{22} = 1$, $x_{11} = x_{22} = 1$ and $x_{31} = x_{32} = 0.5$ with objective value 1.5. The optimal solution to LP2 gives us $\pi_{S_{11}}, \pi_{S_{12}} = 1$, where $S_{11} = \{1, 3\}$, $S_{12} = \{2\}$. This indicates that facilities stay put and clients 1 and 3 are assigned to facility 1 and client 2 is assigned to facility 2, which is in fact the optimal solution to IP1. In this example, the lower bound obtained from LP1 is 1.5 and the lower bound obtained from LP2 is 2. The integrality gap of LP1 is 33%, whereas the integrality gap of LP2 is 0%. □

There is a more serious problem with the linear relaxation of IP1. It may be feasible when IP1 is infeasible. Consider the example in Figure 3 with 2 identical facilities and 2 clients. The distance between the two nodes is 1 and both facilities and clients have weight 1. Facilities 1 and 2 both have capacity 2, while client 1 has demand 1 and client 2 has demand 3. Clearly, this problem is infeasible. However, when we solve LP1, we obtain a feasible solution $y_{11} = y_{22} = 1$, $x_{11} = 1$, $x_{21} = 1/3$, $x_{22} = 2/3$ with objective value 1/3, in which client 1 is assigned to facility 1 and client 2 is partially assigned to both facilities. Both facilities remain at their locations and the capacity constraints are satisfied. In contrast, LP2 is infeasible.

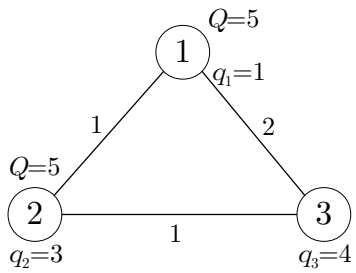

Figure 2: IP1, IP2 and LP2 have objective value of 2 (facilities and clients located at 1 and 2 stay put and client 3 is assigned to facility 1). However, LP1 has an objective value of 1.5, with a solution that splits the demand of client 3 between facilities 1 and 2.

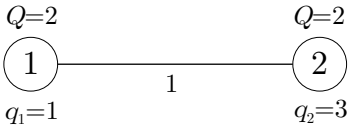

Figure 3: While LP2 is infeasible, LP1 is feasible with an objective value of 1/3.

4 Branch-and-Price Algorithm

In this section we describe a column generation procedure to solve LP2. We then apply a branch-and-price algorithm to IP2 (solving the linear relaxation using the column generation procedure), in which the variables of formulation IP1 are used for branching. We provide three branching alternatives and discuss the management of the columns.

4.1 Column Generation Procedure for LP2

Even though LP2 provides better bounds compared to LP1, there are exponentially many combinations of clients that can make up the set $\mathcal{S}_{tv}$. Instead of solving LP2 with all S_{tv} for all $t \in T, v \in V$, we describe a column generation procedure that generates columns after we solve the restricted master problem (RMP), i.e., LP2 without the complete set of $\pi_{S_{tv}}$ columns. Let us consider the dual of LP2.

$$\text{(LP2}_D\text{) Maximize} \quad \sum_{t\in T} |F_t|\alpha_t + \sum_{i\in C}\beta_i + \sum_{v\in V}\gamma_v + \sum_{j\in F}\delta_j \tag{19}$$

$$\text{subject to} \quad \alpha_t + \sum_{i\in C} a_{iS_{tv}}\beta_i + \gamma_v - \omega_{tv} \le d_{S_{tv}} \qquad \forall t \in T,\ v \in V,\ S_{tv} \in \mathcal{S}_{tv} \tag{20}$$

$$\delta_j + \sum_{t\in T}\omega_{tv} \le w_j d_{jv} \qquad \forall j \in F,\ v \in V \tag{21}$$

$$\gamma_v \le 0 \qquad \forall v \in V \tag{22}$$

For primal optimality, we need dual feasibility. Note that constraint (21) is always satisfied since all of the y_{jv} variables are in the RMP. However, for each $t \in T, v \in V$, we need to make sure there is no assignment S_{tv} such that $\alpha_t + \sum_{i\in C} a_{iS_{tv}}\beta_i + \gamma_v - \omega_{tv} > d_{S_{tv}}$. To find such assignments, we solve the following pricing problem, which is a 0-1 knapsack problem.

$$\text{(KP}(t,v)\text{) Maximize} \quad \sum_{i\in C}(\beta_i - u_i d_{iv})\xi_i \tag{23}$$

$$\text{subject to} \quad \sum_{i \in C} q_i \xi_i \leq R_{tv} \tag{24}$$

$$\xi_i \in \{0,1\} \qquad \forall i \in C \tag{25}$$

When the pricing problem is solved at some node of the branch-and-price tree, the capacity of a facility may already be partially allocated. Therefore, we denote the remaining capacity of a type t facility at v by R_{tv}. At the root node of the tree $R_{tv} = Q_t$. In order to ensure a given solution is primal optimal (dual feasible), we have to solve $\text{KP}(t, v)$ for all facility types t and vertices v (that are not fixed to zero by branching). Note that $\text{KP}(t, v)$ can be solved via dynamic programming in $O(|C|R_{tv})$ time. Let $S_{tv} = \{i \mid \xi_i = 1\}$. We check whether constraint (20) is satisfied. If the constraint is satisfied for all facility types t and vertices v, then we conclude that the solution is optimal. Otherwise, we add the column of $\pi_{S_{tv}}$ for every S_{tv} that violates (20) and resolve the RMP. The general outline of the column generation procedure is as follows.

Column Generation Procedure

Step 1: Generate an initial set of feasible columns for the RMP.

Step 2: Solve the RMP with the existing columns and calculate the values of the optimal dual variables.

Step 3: By solving the pricing problems $\text{KP}(t, v)$, find columns such that (20) is violated. If such columns exist, add them to the RMP and go to Step 2. Otherwise, terminate with the optimal solution.

4.2 Branching Scheme

The knapsack problems we solve for pricing depends on both the branching scheme we employ and the node of the branch-and-price tree. Branching on the variables of IP2 (i.e. $\pi_{S_{tv}}$) is not a viable option for the following reason. Consider branching on the variable $\pi_{S_{tv}}$, where S_{tv} is a feasible client assignment. For the branch where $\pi_{S_{tv}} = 0$, only the specific assignment S_{tv} is forbidden. Therefore, any other assignment in $\mathcal{S}_{tv}$ must still be considered. In order to do that, each client in S_{tv} must be excluded from $\text{KP}(t, v)$ one by one. As the number of forbidden assignments increases, the number of knapsack problems to be solved also increases drastically. However, branching on the variables z_{tv} and x_{iv} does not have this problem and provides a much cleaner column generation process. We describe three branching strategies that use z_{tv} and x_{iv} variables after transforming them as in (17) and (18).

4.2.1 Binary Branching

In binary branching, we fix z_{tv} and x_{iv} variables to 1 in one branch, and to 0 in the other branch. We first start by branching on the z_{tv} variables since it is not possible to branch on x_{iv} without having branched on $z_{tv} = 1$ at one of the parent nodes for some t. Consider the branch where

$z_{tv} = 1$, then some facility of type t will move to v. Since no other type of facility t' can move to v, we set $z_{t'v} = 0$ for all $t' \in T \setminus \{t\}$. In addition, the constraint corresponding to v from constraint (12) has to be set as an equality. That is, for v, the constraint is modified to $\sum_{t \in T} \sum_{S_{tv} \in \mathcal{S}_{tv}} \pi_{S_{tv}} = 1$ in the RMP of LP2. For the branch where $z_{tv} = 0$, vertex v is discarded for type t as a candidate for a facility destination and $\mathrm{KP}(t, v)$ is not solved.

For the branch where $x_{iv} = 1$, client i is assigned to vertex v. Then all $x_{iv'}$ for $v' \in V \setminus \{v\}$ can be set to zero. For all facility types $t \in T$, we adjust the residual capacity to $R_{tv} - q_i$ while solving $\mathrm{KP}(t, v)$ and exclude client i from $\mathrm{KP}(t, v)$. For the branch where $x_{iv} = 0$, client i is simply excluded from $\mathrm{KP}(t, v)$ for all $t \in T$.

Among all fractional z_{tv}, we branch on the most fractional one (i.e. closest to 0.5). If there does not exist a fractional z_{tv}, then among all i and v pairs, we branch on the most fractional x_{iv} given that z_{tv} is fixed to 1.

4.2.2 Partition Branching

Partition branching is similar to the branching strategies proposed for the GAP by Savelsbergh [1997] and for the CPMSP by Ceselli and Righini [2005]. Given a client $i \in C$, we divide the set of vertices V into two sets V^+ and V^0 such that $V^+ = \{v \mid x_{iv} > 0, v \in V\}$ and $V^0 = \{v \mid x_{iv} = 0, v \in V\}$. Then we further partition V^+ and V^0 into two sets such that $V^+ = V_1^+ \cup V_2^+$ and $V^0 = V_1^0 \cup V_2^0$. We set $V_1 = V_1^+ \cup V_1^0$ and $V_2 = V_2^+ \cup V_2^0$. A balanced partition can be achieved by sorting the vertices in V^+ in non-increasing order of x_{iv} and assigning them alternately to V_1^+ and V_2^+. We assign the vertices in V^0 to V_1^0 and V_2^0 in a similar fashion. We branch on the client i^* that satisfies $i^* = \arg\max\{|V^+|\}$, breaking ties arbitrarily. Finally, we set $x_{i^*v} = 0$ for all $v \in V_1$ in one branch and $x_{i^*v} = 0$ for all $v \in V_2$ in the other branch. In the column generation procedure, setting $x_{i^*v} = 0$ translates into removing client i^* from $\mathrm{KP}(t, v)$ for all t.

4.2.3 Hybrid Branching

Ceselli and Righini [2005] reported that partition branching performs better than the binary branching for the CPMSP, which is a special case of the CMFLP where $|T| = 1$ and $w_j = 0$ for all $j \in F$. After preliminary computational experiments performed on instances for the CMFLP with $T = \{1, 2\}$, we have observed that using partition branching by itself is inferior to the binary branching in terms of average computational time and nodes explored. This may attest to the differences between the two problems. In hybrid branching, we use binary branching on z_{tv} variables. When there is no fractional z_{tv}, we employ partition branching for the x_{iv} variables. Though not necessary, branching on z_{tv} variables before the partition branching improves the computational time according to our tests.

4.3 Columns Management

Columns management is an integral part of any column generation procedure as it significantly affects the computational effort required to complete the procedure. There are three pillars to managing columns to which every column generation procedure needs to attend. First, the initial set of columns to start the procedure. Second, the addition of new columns through the pricing problem or other approaches. Third, the management of the existing columns. In the literature, there are various schools of thought on columns management. As it is prohibitive to examine all possible approaches proposed in previous studies, we experimented on a few of the better practices in the literature with adjustments of our own.

4.3.1 Setting initial columns

At the root node of the branch-and-price tree, we generate an initial set of columns for the RMP by a greedy algorithm targeted towards obtaining feasible solutions in short time. We let the facilities stay in their original locations. Therefore, the algorithm only assigns clients to the facilities. Let R_j be the remaining capacity of facility $j \in F$. Initially $R_j = Q_t$, if $j \in F_t$. Also, initially let $F' = F$ and $C' = C$. The initial column generation algorithm is outlined as follows.

Initial Column Generation Algorithm

For each $j \in F'$, go through the following steps while $C' \neq \emptyset$ and $F' \neq \emptyset$.

Step 1: Let $i^* = \arg\min_{i \in C' | q_i \leq R_j} \{\frac{u_i d_{ij}}{q_i}\}$.

Step 2: If $i^* = \emptyset$, then set $F' = F' \setminus \{j\}$; otherwise, assign i^* to facility j and set $R_j = R_j - q_i$ and $C' = C' \setminus \{i^*\}$.

When the algorithm terminates, either $C' = \emptyset$ or $F' = \emptyset$. If $C' = \emptyset$, it means that all clients are assigned and we have a feasible solution. If $F' = \emptyset$, it means there are clients left unassigned and there is no facility with enough remaining capacity to accommodate them. In this case, we run the following *assigned-unassigned client exchange* procedure.

Assigned-Unassigned Client Exchange Procedure

Step 1: For an unassigned client $i \in C'$, assign i to its closest facility $j \in F$ such that $q_i \leq R_j$. If no such facility exists, then let $F' = F$ and go to Step 2.

Step 2: If $F' = \emptyset$, then terminate. Otherwise, consider the facility $j \in F'$ that is closest to i; find a client i' assigned to facility j such that i' satisfies the following criteria:

- $q_i > q_{i'}$
- $R_j - q_i + q_{i'} \geq 0$
- if more than one client satisfy the above criteria, pick the client with the larger $u_{i'} d_{i'j}$.

Step 3: If i' does not exist, set $F' = F' \setminus \{j\}$ and go to Step 2. Otherwise, exchange i with i', i.e., assign i to j and i' to C' and go to Step 1.

Note that even after running this procedure, we may still not have a feasible solution. In that case, we add a separate dummy variable for each constraint with a very large objective function coefficient to have a starting feasible solution.

In addition to a starting feasible solution, we generate more columns by creating a feasible assignment S_{tv} for each facility type $t \in T$ and vertex $v \in V$ according to the following procedure.

Generation of Additional Columns

Step 1: For each v, sort the list of clients with respect to $u_i d_{iv}$ in non-decreasing order. Let $\bar{d_v} = \sum_{i \in C} u_i d_{iv}/|C|$ be the average weighted distance of clients to v.

Step 2: For each facility type t, let $R_{tv} = Q_t$ be the remaining capacity.

- For each client i on the sorted list: Let $r \sim U[0,1]$. If $r \leq e^{-u_i d_{iv}/\bar{d_v}}$, and $q_i \leq R_{tv}$, then add client i to the assignment and set $R_{tv} = R_{tv} - q_i$. Otherwise, process the next client.

We run the additional column generation procedure m times resulting in m feasible assignments for each facility type t and vertex v. Note that instead of completely random assignments, we use this procedure so that clients that are closer to a given vertex v have a higher chance of being in the feasible assignment S_{tv}. After the preliminary computational experiments, we set $m = 5$, as it caused the largest decrease in the average computational time. Compared to $m = 0$, that is, the case where no additional columns are added to the initial feasible solution, setting $m = 5$ decreases the computational time required to solve the root node two to three-fold in most of the instances.

For a child node, the active columns inherited from the parent node can be used as initial columns and the optimal basis for the parent node can be used as a starting feasible basis for the child node. This can be done provided that the columns corresponding to infeasible assignments based on the branching decision have sufficiently large objective function coefficients. That way, these columns will be replaced by other columns that would yield a lower objective value. If column generation procedure terminates with one of the infeasible columns in the optimal basis, we can conclude that the node is infeasible and proceed to prune the node.

4.3.2 Adding columns through pricing

While solving the exact KP(t, v) in every RMP iteration is possible, finding columns that violate (20) does not require solving the pricing problem exactly. Instead, we prefer to use a greedy 2-approximation algorithm to speedup the computational time. The clients are sorted in non-increasing order of $(\beta_i - u_i d_{iv})/q_i$ and the knapsack is filled until no capacity is left. We check constraint (20) for violations. We only solve the exact KP(t, v) when the greedy algorithm fails to find violating columns. If the exact solution also fails to find violating columns, then we terminate with an optimal solution. However, if violating columns have been found after solving the exact KP(t, v), then we add those to the RMP and switch back to applying the greedy algorithm until it fails again.

4.3.3 Managing active columns

Even though there are exponentially many $\pi_{S_{tv}}$ variables, only $|T| + |C| + |V| + |T| \cdot |V|$ can be in the basis, hence a vast majority of them will be non-basic. The size of the problem grows every time we add a column, but the number of basic columns stays exactly the same. Clearly, the growth in the number of columns reflects badly on the computational time. To remedy this, we introduce a procedure that removes columns from the RMP. If a variable is non-basic for κ consecutive iterations, we remove that variable from the RMP. This procedure ensures that the size of the RMP stays in $O(\kappa(|T| + |C| + |V| + |T| \cdot |V|))$. Note that a removed column may be added again. This may increase the number of iterations and the total number of columns added to the RMP but the gain in computational time is well-justified based on the preliminary computational experiments.

5 Heuristics

We describe two heuristics for the CMFLP. The first is an LP rounding heuristic which is employed at all nodes of the branch-and-price tree. The second is a local search heuristic.

5.1 LP Rounding Heuristic

By rounding the optimal fractional solution at any node of the branch-and-price tree, it is possible to quickly find good quality feasible solutions to the CMFLP and generate primal bounds. After calculating the values of z_{tv} and x_{iv} variables from the optimal fractional solution to the corresponding LP2 as in (17) and (18), we run the following heuristic to obtain a feasible solution to the CMFLP.

Step 1: Sort the z_{tv} variables in non-increasing order. Then for each t, select the first $|F_t|$ vertices to a set named Z_t.

Step 2: Solve the facility assignment problems (FA(Z_t, t)) which sets the destination vertices for the facilities.

Step 3: For the client assignment problem, we run the following subroutine.

- Create a list of clients and their preferred vertices. Pair any client i in the list with the vertex v such that x_{iv} is closest to 1.
- Sort the list in non-increasing order of x_{iv}. Starting from the top of the list, assign each client to the facility that has its preferred vertex as the destination. Adjust its remaining capacity. If there is not enough remaining capacity, then go to the next client in the list.
- At the end of the list, if there are some clients left unassigned because there was not enough remaining capacity, assign them to the nearest facility with enough remaining capacity.

Step 4: Finally, run the following improvement heuristic.

- Evaluate all possible client shifts, i.e., removing the client from its current facility and assigning it to a different facility. Implement the shift that would best improve the total cost. If no such shift is found, go to the next step.
- Evaluate all possible client swaps, i.e., exchanging clients that are assigned to different facilities. Implement the swap that would best improve the total cost. If there is no improving swap, then the heuristic is terminated.

The LP rounding heuristic is run every time a feasible LP solution is obtained in the branch-and-price tree. Note that the LP rounding heuristic is not guaranteed to terminate with an integer feasible solution. In fact, determining whether or not an instance to the CMFLP has a feasible solution is NP-Complete.

5.2 Local Search Heuristic

In Halper et al. [2015], the authors describe several heuristics for the MFLP based on the decomposition of the MFLP to facility and client assignment problems for a given set of facility destination vertices. Even though the facility and client assignment problems are different for the CMFLP, the general framework of the local search heuristics still applies. Here, instead of having a single facility assignment problem, we have $|T|$ facility assignment problems and instead of a polynomially solvable client assignment problem, we have the NP-hard generalized assignment problem.

In n-OptSwapBI (where BI stands for best improvement), we are given a set of facility destination vertices $Z \subset V$. For each type t, a subset of k_t facility destinations in Z_t are replaced by a subset of k_t destinations in $V \setminus Z_t$. Every possible combination of replacements across all types are considered such that $1 \leq \sum_{t \in T} k_t \leq n$. For each replacement, the corresponding facility assignment problems are solved optimally by the Hungarian Algorithm. Unlike the MFLP, the CMFLP has multiple facility types. Therefore, in n-OptSwapBI for the CMFLP, a set of facility destinations from one type of facility may be replaced by destinations currently occupied by other facilities of different facility types. In that case, the facility assignment problem has to be solved for every type of facility involved. For the facility assignment problem $\text{FA}(Z_t, t)$, the Hungarian algorithm requires $O(|Z_t|^3)$ from scratch. However, Halper et al. [2015] describe a procedure to update the facility assignments in $O(k_t|Z_t|^2)$, given the previous optimal assignments. We also employ this update procedure in our computations.

To solve the client assignment problem, we use the same greedy algorithm we have used to generate feasible solutions in the column generation procedure outlined in Section 4.3.1, albeit with one caveat. Instead of choosing i^* according to $i^* = \arg\min_{i \in C' | q_i \leq R_j} \{\frac{u_i d_{ij}}{q_i}\}$ in Step 1, we use $i^* = \arg\min_{i \in C' | q_i \leq R_j} \{u_i d_{ij}\}$ in order to target solution quality rather than feasibility. In the case that the algorithm terminates with unassigned clients, we run the same assigned-unassigned client exchange procedure.

In Halper et al. [2015], the computational results indicate that setting $n > 1$ is not viable computationally, even for the MFLP where there is a single facility assignment problem and the client assignment problem is solvable in polynomial time. Hence, we focus on the case where $n = 1$. That is, we consider replacing each facility destination in Z_t with every other destination in $V \setminus Z_t$ for each type t by solving corresponding facility and client assignment problems, and select the replacement that yields the largest decrease in the objective value. If a facility destination is replaced by a destination currently occupied by another type of facility, then the facility assignment problem is solved for both facility types. Note that the neighborhood of 1-OptSwapBI for the CMFLP is populated by at most $\sum_{t \in T} |Z_t|(|V| - |Z_t|)$ possible replacements.

6 Computational Results

In order to assess the solution quality and computational efficiency of the branch-and-price algorithm and the underlying column generation procedure, we coded the branch-and-price algorithm to solve IP2 as described in Section 4. We used the hybrid branching scheme in the results reported since it performed the best during the preliminary computational experiments. We evaluate the nodes in the branch-and-price tree according to breadth-first search. We used CPLEX to solve IP1 as a benchmark to the branch-and-price algorithm. In this section we first provide results on the root node LP relaxations for IP1 and IP2, namely, LP1 and LP2 to compare the strength of the formulations. We also compare the solutions obtained from the LP rounding heuristic based on LP2 (i.e., at the root node of the branch-and-price tree for IP2), namely LP2RH, and the local search heuristic LSH with those obtained from the branch-and-price algorithm. We provide results for both the homogeneous facilities case and the heterogenous facilities case with two facility types.

6.1 Test Instances

The computational experiments are performed on instances titled `p-med` adapted from Halper et al. [2015]. Originally, the `p-med` instances were generated for the p-median problem and adapted to the MFLP by Halper et al. [2015]. We further adapted the instances to the CMFLP to make them capacitated. The computational studies performed on these instances provide insights into the relationships between the solution quality and the computational efficiency of the proposed algorithms, as well as the structural properties of the instances.

The instances are adapted to the CMFLP by generating a demand value q_i for all $i \in C$. We draw q_i randomly from a Gamma distribution with $\alpha = 5$ and $\beta = 2$. If q_i exceeds $\frac{0.8 \cdot E[q_i] \cdot |C|}{|F|}$, we set it to $\frac{0.8 \cdot E[q_i] \cdot |C|}{|F|}$ so that the demand can be served by a single facility with some slack. We experimented with homogeneous facilities ($|T| = 1$) and heterogeneous facilities ($|T| = 2$). For homogeneous facilities, the capacity Q is set to $\frac{\sum_{i \in C} q_i}{0.9 \cdot |F|}$. When we have two types of facilities, the facilities are alternately assigned to F_1 and F_2, and their capacities Q_1 and Q_2 are set to $\frac{0.65 \sum_{i \in C} q_i}{0.9 \cdot |F_1|}$ and $\frac{0.35 \sum_{i \in C} q_i}{0.9 \cdot |F_2|}$. We provide some slack to the total capacity by scaling so that the problem is feasible with very high probability. In fact, we have not encountered an infeasible instance.

6.2 Computational Settings

The CPLEX MIP solver is used to solve IP1 and LP1. We also solved IP1 after disabling the default CPLEX cuts, which we denote as IP1*, to assess the performance of IP1 in a plain branch-and-bound framework. LP2 is solved using the column generation procedure described in Section 4. IP2 is solved using the branch-and-price algorithm proposed in Section 4.2. Within the algorithm, LP2 is used to obtain lower bounds and the LP rounding heuristic is used to obtain upper bounds. Recall that we remove columns staying nonbasic for κ iterations. After preliminary analysis, we concluded that setting κ to $\lceil 0.15 \cdot |V| \rceil$ and $\lceil 0.1 \cdot |V| \rceil$ provides the most average decrease in run time for $|V| \leq 500$ and $|V| > 500$, respectively. We have implemented the branch-and-price algorithm using C++ where the RMP is solved with CPLEX. We used CPLEX version 12.5 coded in C++ in all computational experiments and ran the instances on a computer with Intel Core i7-2600 CPU @ 3.40 GHz and 16 GB of RAM running 64-bit Windows 7. Further, each instance was limited to a three hour (10800 seconds) run time.

In general when there is significant slack capacity in the facilities (i.e., the ratio of the total demand to the total available capacity is significantly less than 1) IP1 and its relaxation LP1 work well. In these cases the capacity constraints do not play much of a role and given that IP1 works well for the uncapacitated version of the problem, i.e., the MFLP, it is not a surprise that IP1 works well in these cases. Similarly, the local search heuristic, which performs well for the MFLP as reported in Halper et al. [2015], also performs well for the CMFLP in these cases.

On the other hand, the practical setting of the CMFLP is in an environment where capacity constraints are tight (i.e., there is not too much unused capacity in the problem instance) as budgets are tight and organizations are keen on efficiently utilizing their resources to the fullest extent. In these tightly capacitated scenarios (which our simulated instances are) we observed a relationship between the ratio of the number of clients to the number of facilities, $|C|/|F|$, and the quality of solutions obtained from IP1 and its relaxation LP1. In 17 of the 40 `p-med` instances where the ratio of $|C|/|F| > 10$ (i.e., client demands are small compared to the facility capacity), the packing problem (i.e., GAP) is easy to solve and LP1 provides tight bounds (for instances with homogenous facilities the average gap between the lower bound provided by LP1 and the upper bound provided by IP1 is 0.63% and for instances with heterogenous facilities this average gap is 0.72%). For these problem instances where IP1 works well, we would recommend IP1 over IP2 since it is a compact formulation and is very easy to implement in a commercial solver like CPLEX (as compared to IP2 which requires special purpose column generation code to be written). In 23 of the 40 `p-med` instances where the ratio of $|C|/|F| \leq 10$ (i.e., client demands are somewhat larger compared to the facility capacity), the quality of LP1 significantly deteriorates and IP1 becomes computationally challenging to solve. Our focus in our computational experiments will be on these tightly capacitated instances (the ratio of the total demand to the total capacity is 0.9) where $|C|/|F| \leq 10$ to see whether IP2 and the column generation approach provide a viable alternative to IP1.

In the `p-med` instances adapted from Halper et al. [2015] problem instances range from 100 to

900 nodes. To ensure we have instances where $|C|/|F| = 3$, $|C|/|F| = 5$, and $|C|/|F| = 10$ for each problem size, we generated 5 additional instances to the 23 where the ratio of $|C|/|F| \leq 10$ (for a total of 28 instances). We generated one 700 node instance (named `p-med34-1` using the shortest path distances, client and facility weights and client demands of instance `p-med34` only generating more facility locations to conform to the structure observed in the first 30 instances. In a similar fashion, two instances each were generated for 800 and 900 node instances (named `p-med37-1` and `p-med37-2`, and `p-med40-1` and `p-med40-2`).

6.3 Homogeneous Facilities Case

Tables 1, 2, and 3 describe our results when $|C|/|F| \leq 10$. If an instance of IP1 or IP2 was terminated at three hours, we report the objective value of the best integer solution found, namely the best upper bound, and the best lower bound found. In all tables, if a value cannot be calculated due to the time limit, we denote the corresponding cell with '-'. The running times exceeding three hours are also denoted with '-'.

6.3.1 Comparison of the LP Relaxations

Table 1 presents the bounds obtained from LP1 and LP2 with respect to the best IP lower and upper bounds. The first column specifies the names of the instances. Generically, let X(L) denote the best lower bound obtained from model X after 3 hours of computation, where X can be either IP1, IP1* or IP2. Similarly, X(U) denotes the objective value of the best feasible integer solution obtained from model X after 3 hours of computation. In the first group of columns, we report the best lower bound (BL) from either IP1(L), IP1*(L), or IP2(L) and the best upper bound (BU) from either IP1(U), IP1*(U), IP2(U) for each instance, along with the source of the bound. If the same bounds are found by IP1, IP1*, and IP2, we specify the source as 'ALL'. Note that if IP1, IP1* or IP2 terminated with the optimal solution, then BU=BL. For example, in Table 1, the source of the best known upper bound for the `p-med15` instance is IP1(U). That is, the integer feasible solution with the lowest objective function value for `p-med15` is obtained from IP1. On the other hand, the source of the best known lower bound for `p-med15` is IP2(L), meaning that it is found while solving IP2 by the branch-and-price algorithm. In the second group of columns labeled Gap (%), we provide the percentage gaps between various formulations. For all reported gaps denoted as X-Y, the gaps are calculated as $(X - Y)/X$. The column labeled BU-BL provides the gap between the best upper bound and the best lower bound, i.e. the best known optimality gap. The columns labeled BU-LP1 and BU-LP2 denote the gap between the best upper bound and the lower bounds obtained by solving LP1 and LP2, respectively. Similarly, the columns labeled BL-LP1 and BL-LP2 denote the gap between the best lower bound and lower bounds of LP1 and LP2. The group of columns labeled 'Running time (s)' provides the CPU times in seconds. Finally, the size of the instances are given in the last group of columns.

In Table 1 we observe that as $|C|$ gets larger, especially for $|C| \geq 600$, IP2 starts to overtake IP1 at finding the best upper bound. Furthermore, the best lower bound is found by IP2 when

$|C| \geq 300$. LP2 provides significantly smaller gaps than LP1 on the average with 1.80% versus 5.62% compared to the best upper bound. In addition, we observe that the quality of LP2 compared to LP1 gets progressively better as $|C|/|F|$ gets smaller. Compared to the upper bound, LP2 has an average gap of 0.80% versus 1.27% of LP1 for $|C|/|F| = 10$. The gaps become 2.09% versus 4.92% for $|C|/|F| = 5$ and 2.62% versus 11.17% for $|C|/|F| = 3$. We attribute this difference to the packing constraints (7) in LP1 which lead to more fractional variables as $|F|$ gets larger. These computational results confirm the theoretical finding that IP2 is a stronger formulation than IP1. In general, as expected, LP1 runs faster than LP2 with an average of 128.2 versus 519.7 seconds. As a result, we observe an apparent trade-off between obtaining smaller gaps and having longer run times.

6.3.2 Comparison of the Lower Bounds

Table 2 presents the computational results for IP1(L), IP1*(L), and IP2(L) and the running times of IP1, IP1*, and IP2. The columns labeled BL-IP1(L), BL-IP1*(L) and BL-IP2(L) report the gap between the best lower bound and IP1(L), IP1*(L) and IP2(L). On average, IP2(L) is better than IP1(L) with an average value of 0.04% compared to 0.47%, which is somewhat expected given the quality of LP2 versus LP1 when $|C|/|F| \leq 10$. In fact, CPLEX does a pretty good job closing the initial gap of LP1, which is 4.27% on average with respect to the best lower bound. On the other hand, deprived of its state-of-the-art cuts, we observe that the poor quality of LP1 hinders the ability of IP1* at closing the gap, which is 3.53% on average but gets as large as 9.71%. In terms of running time, IP1, IP1* and IP2 all hit the three hour limit when $|C| \geq 300$ (except `p-med13` for IP1) with IP1 faring slightly better.

6.3.3 Comparison of the Upper Bounds

Table 3 presents the computational results for IP1(U), IP1*(U), IP2(U), the LP rounding heuristic from LP2 (LP2RH), and the local search heuristic (LSH). The columns labeled IP1(U)-BL, IP1*(U)-BL and IP2(U)-BL report the gap between IP1(U), IP1*(U) and IP2(U), and the best lower bound, i.e., the optimality gap. In the column LP2RH-BL, the gap between the best integer feasible solution found by LP2RH and the best lower bound is given. In the next column labeled LSH-BL, we provide the gap between the feasible solution found by LSH and the best lower bound.

We observe that on average, IP2 performs better than IP1 in finding a good feasible solution. IP1 gives an average gap of 4.02% while IP2 yields 1.60% gap on the average with respect to the best lower bound. When we discard instance `p-med40-2` for which IP1 terminated with a very poor quality upper bound, IP2 still performs better compared to the lower bound with gaps 1.54% vs. 1.68%. Especially when $|C| \geq 600$, excluding instance `p-med40-2`, the average IP1(U)-BL gap is 3.12% versus 2.56% of IP2(U)-BL, which signals that the relative quality of IP2 solutions get better in larger instances.

Table 1: Comparison of the quality of LP1 and LP2 on homogenous instances.

Instance	Objective Value				Gap (%)					Runtime (s)		$V = C$	
	BL	Source	BU	Source	BU-BL	BU-LP1	BU-LP2	BL-LP1	BL-LP2	LP1	LP2	$\|C\|$	$\|C\|/\|F\|$
pmed2	5275.27	ALL	5275.27	ALL	0.00%	1.47%	0.60%	1.47%	0.60%	0.42	0.98	100	10
pmed3	6023.05	ALL	6023.05	ALL	0.00%	0.61%	0.27%	0.61%	0.27%	0.24	0.89	100	10
pmed4	5374.39	ALL	5374.39	ALL	0.00%	4.19%	0.78%	4.19%	0.78%	0.26	0.45	100	5
pmed5	3320.49	ALL	3320.49	ALL	0.00%	9.86%	2.22%	9.86%	2.22%	0.30	0.34	100	3
pmed8	6658.71	ALL	6658.71	ALL	0.00%	1.31%	0.70%	1.31%	0.70%	1.94	16.66	200	10
pmed9	5046.37	IP1(L)	5061.78	IP1(U)	0.30%	4.31%	1.21%	4.02%	0.90%	2.81	4.17	200	5
pmed10	3128.94	IP1(L)	3128.94	IP1(U)	0.00%	8.56%	0.92%	8.56%	0.92%	1.62	2.90	200	3
pmed13	5690.55	IP1(L)	5690.55	IP1(U)	0.00%	0.73%	0.32%	0.73%	0.32%	7.75	45.52	300	10
pmed14	4427.26	**IP2(L)**	4465.27	IP1*(U)	0.85%	4.15%	1.16%	3.32%	0.31%	7.77	10.76	300	5
pmed15	3416.81	**IP2(L)**	3459.56	IP1(U)	1.24%	9.65%	1.47%	8.52%	0.24%	5.50	9.92	300	3
pmed18	6010.19	**IP2(L)**	6038.28	IP1(U)	0.47%	1.04%	0.70%	0.58%	0.23%	19.95	93.34	400	10
pmed19	4674.05	**IP2(L)**	4749.88	IP1*(U)	1.60%	4.80%	2.03%	3.25%	0.44%	14.99	36.60	400	5
pmed20	3928.93	**IP2(L)**	3966.80	IP1(U)	0.95%	10.25%	1.13%	9.39%	0.18%	14.81	23.65	400	3
pmed23	6996.73	**IP2(L)**	7023.91	**IP2(U)**	0.39%	1.04%	0.59%	0.66%	0.20%	62.15	252.86	500	10
pmed24	5320.38	**IP2(L)**	5397.27	IP1(U)	1.42%	4.25%	1.62%	2.87%	0.20%	34.57	124.43	500	5
pmed25	3852.84	**IP2(L)**	3972.88	IP1*(U)	3.02%	12.11%	3.19%	9.38%	0.17%	27.83	85.02	500	3
pmed28	6367.63	**IP2(L)**	6390.13	IP1(U)	0.35%	0.90%	0.43%	0.55%	0.08%	101.12	386.88	600	10
pmed29	5286.17	**IP2(L)**	5405.70	**IP2(U)**	2.21%	5.34%	2.35%	3.20%	0.14%	64.94	193.69	600	5
pmed30	4031.32	**IP2(L)**	4187.17	**IP2(U)**	3.72%	13.41%	4.04%	10.06%	0.33%	40.11	91.39	600	3
pmed33	6981.21	**IP2(L)**	7007.81	IP1(U)	0.38%	0.87%	0.51%	0.49%	0.13%	227.11	793.56	700	10
pmed34	5075.42	**IP2(L)**	5254.71	**IP2(U)**	3.41%	6.07%	3.55%	2.75%	0.14%	163.35	222.80	700	5
pmed34-1	4079.74	**IP2(L)**	4241.66	**IP2(U)**	3.82%	12.10%	3.93%	8.61%	0.12%	59.85	129.40	700	3
pmed37	6536.65	**IP2(L)**	6646.71	IP1(U)	1.66%	2.21%	1.81%	0.57%	0.16%	384.68	1055.25	800	10
pmed37-1	5081.89	**IP2(L)**	5218.90	**IP2(U)**	2.63%	5.49%	2.71%	2.94%	0.09%	279.35	592.47	800	5
pmed37-2	4190.18	**IP2(L)**	4337.52	IP1(U)	3.40%	12.44%	3.48%	9.36%	0.08%	111.42	6332.21	800	3
pmed40	7397.00	**IP2(L)**	7543.70	IP1(U)	1.94%	2.53%	2.09%	0.60%	0.15%	1127.80	2115.01	900	10
pmed40-1	5790.58	**IP2(L)**	5984.67	**IP2(U)**	3.24%	5.64%	3.37%	2.47%	0.13%	478.20	1164.40	900	5
pmed40-2	4642.74	**IP2(L)**	4790.29	**IP2(U)**	3.08%	12.12%	3.19%	9.33%	0.11%	220.26	246.65	900	3
				Min	0.00%	0.61%	0.27%	0.49%	0.08%	0.24	0.34		
				Max	3.82%	13.41%	4.04%	10.06%	2.22%	1127.80	6332.21		
				Avg	1.43%	5.62%	1.80%	4.27%	0.37%	123.61	501.15		

Table 2: Comparison of the quality of lower bounds obtained from IP1, IP1* and IP2 on homogenous instances.

Instance	Gap (%)			Runtime (s)			$V = C$	
	BL-IP1(L)	BL-IP1*(L)	BL-IP2(L)	IP1	IP1*	IP2	$\|C\|$	$\|C\|/\|F\|$
pmed2	0.00%	0.00%	0.00%	9.45	34.45	13.26	100	10
pmed3	0.00%	0.00%	0.00%	1.05	1.15	10.13	100	10
pmed4	0.00%	0.00%	0.00%	32.67	973.69	41.24	100	5
pmed5	0.00%	4.01%	0.00%	112.29	-	3868.58	100	3
pmed8	0.00%	0.47%	0.00%	1227.54	-	7949.99	200	10
pmed9	0.00%	2.92%	0.32%	-	-	-	200	5
pmed10	0.00%	6.52%	0.62%	1591.44	-	-	200	3
pmed13	0.00%	0.32%	0.06%	1048.31	-	-	300	10
pmed14	0.39%	2.69%	0.00%	-	-	-	300	5
pmed15	0.44%	7.88%	0.00%	-	-	-	300	3
pmed18	0.07%	0.24%	0.00%	-	-	-	400	10
pmed19	0.84%	2.93%	0.00%	-	-	-	400	5
pmed20	0.42%	8.86%	0.00%	-	-	-	400	3
pmed23	0.29%	0.57%	0.00%	-	-	-	500	10
pmed24	0.66%	2.55%	0.00%	-	-	-	500	5
pmed25	0.43%	8.88%	0.00%	-	-	-	500	3
pmed28	0.15%	0.47%	0.00%	-	-	-	600	10
pmed29	0.71%	3.13%	0.00%	-	-	-	600	5
pmed30	0.79%	9.71%	0.00%	-	-	-	600	3
pmed33	0.25%	0.45%	0.00%	-	-	-	700	10
pmed34	0.89%	2.69%	0.00%	-	-	-	700	5
pmed34-1	1.47%	8.47%	0.00%	-	-	-	700	3
pmed37	0.36%	0.53%	0.00%	-	-	-	800	10
pmed37-1	1.00%	2.93%	0.00%	-	-	-	800	5
pmed37-2	1.28%	9.31%	0.00%	-	-	-	800	3
pmed40	0.36%	0.57%	0.00%	-	-	-	900	10
pmed40-1	1.00%	2.46%	0.00%	-	-	-	900	5
pmed40-2	1.40%	9.20%	0.00%	-	-	-	900	3
Min	0.00%	0.00%	0.00%	1.05	1.15	10.13		
Max	1.47%	9.71%	0.62%	-	-	-		
Avg	0.47%	3.53%	0.04%	8273.57	9822.32	9297.10		

Table 3: Comparison of the quality of upper bounds obtained from IP1, IP1*, IP2, LP2RH, and LSH on homogenous instances.

Instance	Gap (%) IP1(U)-BL	IP1*(U)-BL	IP2(U)-BL	LP2RH-BL	LSH-BL	Runtime (s) LSH	Nodes Explored	$V = C$ $\|C\|$	$\|C\|/\|F\|$
pmed2	0.00%	0.00%	0.00%	0.90%	3.54%	0.32	33	100	10
pmed3	0.00%	0.00%	0.00%	0.24%	3.10%	0.29	7	100	10
pmed4	0.00%	0.00%	0.00%	3.80%	6.39%	1.48	366	100	5
pmed5	0.00%	0.00%	0.00%	7.10%	12.84%	4.43	49973	100	3
pmed8	0.00%	0.06%	0.00%	2.57%	5.32%	6.48	944	200	10
pmed9	0.30%	0.59%	0.30%	2.15%	10.41%	34.77	31539	200	5
pmed10	0.00%	0.40%	0.68%	6.04%	13.78%	68.33	38760	200	3
pmed13	0.00%	0.00%	0.01%	0.70%	2.80%	40.73	1125	300	10
pmed14	0.95%	0.85%	1.11%	4.92%	8.20%	191.19	14590	300	5
pmed15	1.24%	1.58%	1.93%	10.04%	16.68%	401.13	12410	300	3
pmed18	0.47%	0.51%	0.69%	4.50%	3.33%	188.12	2489	400	10
pmed19	2.06%	1.60%	1.80%	4.90%	9.61%	863.69	6742	400	5
pmed20	0.95%	1.93%	1.79%	7.28%	12.97%	2100.53	6317	400	3
pmed23	0.44%	0.64%	0.39%	2.55%	4.19%	559.10	1521	500	10
pmed24	1.42%	1.61%	1.79%	6.00%	8.45%	3126.61	4128	500	5
pmed25	3.13%	3.02%	3.06%	6.70%	15.73%	5201.88	3552	500	3
pmed28	0.35%	0.39%	0.51%	3.92%	5.89%	1184.15	1311	600	10
pmed29	2.87%	3.26%	2.21%	4.14%	8.77%	4679.60	3785	600	5
pmed30	5.83%	5.80%	3.72%	5.86%	16.23%	-	3003	600	3
pmed33	0.38%	0.48%	0.58%	3.83%	4.10%	2384.93	907	700	10
pmed34	3.99%	4.72%	3.41%	5.47%	8.73%	-	2855	700	5
pmed34-1	5.03%	5.62%	3.82%	5.62%	20.20%	-	2062	700	3
pmed37	1.66%	1.92%	1.72%	3.13%	5.06%	3905.30	620	800	10
pmed37-1	3.21%	4.39%	2.63%	4.01%	12.98%	-	1931	800	5
pmed37-2	3.40%	9.39%	4.18%	6.80%	22.16%	-	586	800	3
pmed40	1.94%	2.59%	2.12%	5.64%	4.32%	7157.25	519	900	10
pmed40-1	5.61%	5.08%	3.24%	5.47%	18.71%	-	1060	900	5
pmed40-2	67.21%	9.57%	3.08%	5.47%	26.04%	-	876	900	3
Min	0.00%	0.00%	0.00%	0.24%	2.80%	0.29	7		
Max	67.21%	9.57%	4.18%	10.04%	26.04%	-	49973		
Avg	4.02%	2.36%	1.60%	4.63%	10.38%	3846.47	6928.96		

The LP rounding heuristic performs fairly well, given the hardness of this subset of instances (i.e., where $|C|/|F| \leq 10$) and the poor quality of the local search heuristic. On average, the gap is 4.63%, which means by just solving LP2 at the root node and using the LP rounding heuristic, we get an integer solution which is on average at most 4.63% away from the optimal. However, the quality of the local search heuristic is extremely poor with an average gap of 10.38%. This is somewhat expected since the greedy procedure for the GAP tends to work better when the number of items assigned to a single bin gets larger. The running time of the LP rounding heuristic nearly equals the running time of LP2 since the steps after LP2 solution take negligible time. Therefore, we see that the LP rounding heuristic is quite fast for $|C|/|F| \leq 10$ and gets even faster as $|C|/|F|$ gets smaller. In contrast, the local search heuristic runs quite slowly since the neighborhood size is larger for $|C|/|F| \leq 10$. We observe that the speed of LP2 does not translate into IP2 since the number of nodes explored increases as $|C|/|F|$ gets smaller. We see two reasons for the increased number of nodes. The first one is simply the increased number of variables as $|F|$ gets larger. The second reason is more subtle. After fixing a z_{tv} variable, we naturally expect the lower bound obtained at that node to increase. However, as $|F|$ increases, the marginal increase in the lower bound caused by fixing a single facility decreases. This in turn makes the algorithm explore more nodes as we observe in the column labeled 'Nodes Explored', which provides the total number of nodes explored by the branch-and-price algorithm that solves IP2.

6.4 Heterogeneous Facilities Case

Tables 4 through 6 present the results using the same layout and format as for homogeneous facilities. We did not implement the Local Search heuristic for the heterogeneous facilities since it is significantly outperformed by every other method when $|C|/|F| \leq 10$. Furthermore, its neighborhood structure and size make its performance highly predictable in terms of time and quality.

In Table 4, the BU-BL gap values show that the presence of heterogeneous facilities increases the difficulty of the problem as expected. The average BU-BL gaps are observed to be 1.43% and 2.86%, respectively for homogeneous and heterogeneous facilities. In general, the insights we gain from homogeneous facilities are still valid and more pronounced in heterogeneous facilities. In homogeneous facilities, we observe that IP2 starts to produce better upper bounds compared to IP1 when $|C| \geq 600$. Not only the trend holds up even stronger for heterogeneous facilities, but the quality of the upper bounds obtained from IP1 rapidly declines after $|C| \geq 700$. After disabling default CPLEX cuts, IP1* failed to find a feasible solution in three hours for six out of nine instances for $|C| \geq 700$. The quality of the LP rounding heuristic slightly declines for heterogeneous facilities. However, compared to IP1, the LP rounding heuristic provides good quality solutions in reasonable time. Especially for $|C| \geq 700$, the LP rounding heuristic generally outperforms IP1. Similarly, the lower bound IP1(L) obtained from IP1 after three hours is worse than the lower bound obtained from LP2 for the same instances. Interestingly after solving the root node, the branch-and-price algorithm has better lower and upper bounds than IP1 has at its termination after three hours.

Table 4: Comparison of the quality of LP1 and LP2 on heterogenous instances.

Instance	Objective Value				Gap (%)					Runtime (s)		$V = C$	
	BL	Source	BU	Source	BU-BL	BU-LP1	BU-LP2	BL-LP1	BL-LP2	LP1	LP2	$\|C\|$	$\|C\|/\|F\|$
pmed2	5064.83	ALL	5064.83	ALL	0.00%	1.25%	0.48%	1.25%	0.48%	0.24	7.47	100	10
pmed3	6162.93	ALL	6162.93	ALL	0.00%	1.99%	0.88%	1.99%	0.88%	0.19	1.98	100	10
pmed4	5172.33	ALL	5172.33	ALL	0.00%	3.16%	1.30%	3.16%	1.30%	0.17	0.73	100	5
pmed5	3302.98	ALL	3302.98	ALL	0.00%	9.10%	1.19%	9.10%	1.19%	0.22	0.53	100	3
pmed8	6484.06	IP1(L)	6484.06	IP1(U)	0.00%	2.18%	0.48%	2.18%	0.48%	2.62	32.15	200	10
pmed9	5043.84	**IP2(L)**	5115.69	IP1(U)	1.40%	6.02%	1.73%	4.68%	0.33%	2.40	6.24	200	5
pmed10	3199.40	IP1(L)	3240.95	IP1(U)	1.28%	11.36%	1.87%	10.21%	0.60%	2.62	9.92	200	3
pmed13	5673.47	IP1(L)	5673.47	IP1(U)	0.00%	1.09%	0.49%	1.09%	0.49%	10.90	87.39	300	10
pmed14	4372.98	**IP2(L)**	4414.65	IP1(U)	0.94%	4.40%	1.12%	3.49%	0.18%	6.43	17.36	300	5
pmed15	3364.28	**IP2(L)**	3447.41	IP1(U)	2.41%	10.63%	2.53%	8.42%	0.13%	6.38	13.31	300	3
pmed18	5997.19	**IP2(L)**	6038.59	IP1*(U)	0.69%	1.81%	0.80%	1.13%	0.12%	15.51	133.51	400	10
pmed19	4701.85	**IP2(L)**	4830.43	IP1*(U)	2.66%	6.28%	2.74%	3.72%	0.08%	16.68	34.05	400	5
pmed20	3973.66	**IP2(L)**	4162.68	IP1(U)	4.54%	14.31%	4.66%	10.23%	0.13%	15.07	30.55	400	3
pmed23	6997.44	**IP2(L)**	7039.70	IP1*(U)	0.60%	1.50%	0.67%	0.91%	0.07%	58.06	355.03	500	10
pmed24	5247.22	**IP2(L)**	5406.64	**IP2(U)**	2.95%	6.08%	3.02%	3.23%	0.07%	46.93	82.17	500	5
pmed25	3780.82	**IP2(L)**	3944.54	IP1(U)	4.15%	12.79%	4.21%	9.02%	0.06%	29.16	44.69	500	3
pmed28	6352.88	**IP2(L)**	6429.52	IP1(U)	1.19%	2.21%	1.26%	1.03%	0.07%	107.78	609.07	600	10
pmed29	5184.69	**IP2(L)**	5332.38	**IP2(U)**	2.77%	5.61%	2.84%	2.92%	0.07%	79.42	128.86	600	5
pmed30	3931.89	**IP2(L)**	4213.84	**IP2(U)**	6.69%	15.90%	6.77%	9.87%	0.08%	43.32	80.81	600	3
pmed33	6970.24	**IP2(L)**	7064.82	IP1*(U)	1.34%	2.26%	1.39%	0.93%	0.05%	260.57	861.42	700	10
pmed34	5056.57	**IP2(L)**	5267.82	**IP2(U)**	4.01%	6.76%	4.06%	2.87%	0.05%	204.86	202.46	700	5
pmed34-1	4079.69	**IP2(L)**	4358.50	**IP2(U)**	6.40%	15.31%	6.44%	9.52%	0.05%	129.03	124.13	700	3
pmed37	6507.84	**IP2(L)**	6655.00	**IP2(U)**	2.21%	3.14%	2.26%	0.95%	0.05%	500.87	969.15	800	10
pmed37-1	5092.52	**IP2(L)**	5352.55	**IP2(U)**	4.86%	8.55%	4.91%	3.89%	0.05%	446.82	341.45	800	5
pmed37-2	4134.17	**IP2(L)**	4558.39	**IP2(U)**	9.31%	18.41%	9.36%	10.04%	0.05%	159.37	187.87	800	3
pmed40	7351.84	**IP2(L)**	7510.01	**IP2(U)**	2.11%	2.98%	2.15%	0.89%	0.05%	1102.80	2620.10	900	10
pmed40-1	5742.88	**IP2(L)**	5918.55	**IP2(U)**	2.97%	6.12%	3.00%	3.25%	0.03%	575.02	524.82	900	5
pmed40-2	4731.60	**IP2(L)**	5192.53	**IP2(U)**	8.88%	17.77%	8.91%	9.76%	0.03%	313.58	238.67	900	3
				Min	0.00%	1.09%	0.48%	0.89%	0.03%	0.17	0.53		
				Max	9.31%	18.41%	9.36%	10.23%	1.30%	1102.80	2620.10		
				Avg	2.66%	7.11%	2.91%	4.63%	0.26%	147.75	276.64		

Table 5: Comparison of the quality of lower bounds obtained from IP1, IP1* and IP2 on heterogenous instances.

Instance	Gap (%)			Runtime (s)			$V = C$	
	BL-IP1(L)	BL-IP1*(L)	BL-IP2(L)	IP1	IP1*	IP2	$\|C\|$	$\|C\|/\|F\|$
pmed2	0.00%	0.00%	0.00%	5.78	5.48	439.52	100	10
pmed3	0.00%	0.00%	0.00%	17.43	185.24	2113.54	100	10
pmed4	0.00%	0.00%	0.00%	26.18	205.45	277.40	100	5
pmed5	0.00%	0.00%	0.00%	194.19	-	1642.23	100	3
pmed8	0.00%	0.44%	0.31%	2240.12	-	-	200	10
pmed9	0.31%	3.19%	0.00%	-	-	-	200	5
pmed10	0.00%	8.54%	0.37%	-	-	-	200	3
pmed13	0.00%	0.38%	0.36%	8570.64	-	-	300	10
pmed14	0.60%	2.88%	0.00%	-	-	-	300	5
pmed15	0.75%	7.33%	0.00%	-	-	-	300	3
pmed18	0.40%	0.61%	0.00%	-	-	-	400	10
pmed19	1.28%	3.37%	0.00%	-	-	-	400	5
pmed20	1.54%	9.62%	0.00%	-	-	-	400	3
pmed23	0.48%	0.61%	0.00%	-	-	-	500	10
pmed24	1.33%	3.12%	0.00%	-	-	-	500	5
pmed25	1.66%	8.92%	0.00%	-	-	-	500	3
pmed28	0.58%	0.85%	0.00%	-	-	-	600	10
pmed29	1.24%	2.84%	0.00%	-	-	-	600	5
pmed30	1.51%	9.57%	0.00%	-	-	-	600	3
pmed33	0.63%	0.90%	0.00%	-	-	-	700	10
pmed34	1.41%	2.80%	0.00%	-	-	-	700	5
pmed34-1	2.23%	9.49%	0.00%	-	-	-	700	3
pmed37	0.71%	0.93%	0.00%	-	-	-	800	10
pmed37-1	2.11%	3.85%	0.00%	-	-	-	800	5
pmed37-2	3.06%	10.02%	0.00%	-	-	-	800	3
pmed40	0.81%	0.87%	0.00%	-	-	-	900	10
pmed40-1	1.81%	3.25%	0.00%	-	-	-	900	5
pmed40-2	3.16%	9.75%	0.00%	-	-	-	900	3
Min	0.00%	0.00%	0.00%	5.78	5.48	277.40		
Max	3.16%	10.02%	0.37%	-	-	-		
Avg	0.99%	3.72%	0.04%	8926.10	9716.26	9417.65		

Table 6: Comparison of the quality of upper bounds obtained from IP1, IP1*, IP2, and LP2RH on heterogenous instances.

Instance	Gap (%) IP1(U)-BL	IP1*(U)-BL	IP2(U)-BL	LP2RH-BL	Nodes Explored	$V = C$ $\|C\|$	$\|C\|/\|F\|$
pmed2	0.00%	0.00%	0.00%	1.78%	223	100	10
pmed3	0.00%	0.00%	0.00%	5.37%	2632	100	10
pmed4	0.00%	0.00%	0.00%	3.72%	1745	100	5
pmed5	0.00%	0.00%	0.00%	6.64%	18422	100	3
pmed8	0.00%	0.08%	0.21%	0.89%	932	200	10
pmed9	1.40%	2.23%	2.66%	6.32%	21716	200	5
pmed10	1.28%	1.64%	3.26%	7.89%	29006	200	3
pmed13	0.00%	0.12%	0.28%	1.51%	1032	300	10
pmed14	0.94%	1.88%	1.91%	4.85%	11439	300	5
pmed15	2.41%	2.60%	5.31%	13.47%	14975	300	3
pmed18	0.73%	0.69%	0.86%	2.17%	3221	400	10
pmed19	2.74%	2.66%	3.54%	10.15%	10408	400	5
pmed20	4.54%	7.66%	6.95%	12.05%	9592	400	3
pmed23	0.70%	0.60%	1.12%	3.08%	2484	500	10
pmed24	3.45%	3.05%	2.95%	7.19%	6493	500	5
pmed25	4.15%	5.92%	6.68%	13.09%	5447	500	3
pmed28	1.19%	1.25%	1.25%	5.71%	1334	600	10
pmed29	6.82%	6.58%	2.77%	3.95%	3781	600	5
pmed30	10.81%	18.81%	6.69%	10.71%	3964	600	3
pmed33	1.58%	1.34%	1.61%	1.65%	960	700	10
pmed34	4.98%	7.69%	4.01%	7.06%	3021	700	5
pmed34-1	71.41%	-	6.40%	8.48%	2529	700	3
pmed37	2.44%	-	2.21%	6.13%	601	800	10
pmed37-1	62.34%	-	4.86%	5.51%	1983	800	5
pmed37-2	11.66%	-	9.31%	11.31%	1746	800	3
pmed40	52.19%	50.73%	2.11%	5.14%	436	900	10
pmed40-1	60.86%	-	2.97%	4.42%	1375	900	5
pmed40-2	70.75%	-	8.88%	12.05%	1304	900	3
Min	0.00%	0.00%	0.00%	0.89%	223		
Max	71.41%	50.73%	9.31%	13.47%	29006		
Avg	13.55%	5.25%	3.17%	6.51%	5814.32		

7 Conclusions

In this paper we introduced the CMFLP, a problem that arises in the logistics planning of community outreach programs delivered via mobile facilities. We provided two integer programming formulations for the CMFLP. The first (IP1) is a layered graph formulation adapted from the MFLP formulation in Halper et al. [2015] to account for the capacity restrictions. The second (IP2) is a set partitioning formulation. We show that the LP relaxation of IP2 (LP2) is stronger than the LP relaxation of IP1 (LP1) and propose an efficient column generation procedure to solve LP2, which is used within a branch-and-price algorithm to solve IP2. Within the branch-and-price algorithm, we use a greedy 2-approximation algorithm to solve the pricing problem and only solve the pricing problem exactly when the heuristic algorithm fails to find a column to be added to the RMP of LP2. We also keep track of variables that have been nonbasic for a certain number of iterations and remove them from the problem to maintain tractability. Since branching on the variables of IP2 was not a viable option, we implicitly branch on the variables of IP1 in the branch-and-bound tree while using IP2 as the relaxation. These strategies result in a computationally effective column generation and branch-and-price procedure. We propose and test two heuristics for the CMFLP. The first is an LP rounding heuristic that uses the fractional variables from the column generation procedure. The second is a local search heuristic called 1-OptSwapBI, originally proposed for the MFLP, that uses the decomposition of IP1 into client and facility subproblems.

The computational results provide some interesting conclusions. The increase in the total number of vertices makes the problem harder to tackle. However, IP1 has more trouble handling larger problems than IP2, especially when the average number of clients assigned to a facility is small. When $|C|/|F| \leq 10$, IP2 and LP2 dominate IP1 and LP1 respectively. The disparity between IP1 and IP2 are accentuated when we consider heterogenous facilities as opposed to homogenous facilities. A similar behavior is observed with the heuristics. When the average number of clients assigned to a facility is small ($|C|/|F| \leq 10$) the local search heuristic performs poorly. On the other hand, under these conditions the LP rounding heuristic runs faster and provides high quality upper bounds.

We suggest IP1 as the go-to formulation in loosely capacitated problems as well as problems where the average number of clients assigned to a facility is large. However, for problems where capacity constraints play a more significant role (i.e., where the ratio of total demand to total capacity is closer to 1 and the ratio of clients to number of facilities is less than or equal to 10) IP2 becomes the better formulation.

References

B. Addis, G. Carello, and A. Ceselli. Exactly solving a two-level location problem with modular node capacities. *Networks*, 59(1):161–180, 2012.

B. Addis, G. Carello, and A. Ceselli. Combining very large scale and {ILP} based neighborhoods

for a two-level location problem. *European Journal of Operational Research*, 231(3):535–546, 2013.

S. Ahmadian, Z. Friggstad, and C. Swamy. Local-search based approximation algorithms for mobile facility location problems. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1607–1621, 2013.

R. K. Ahuja, J. B. Orlin, S. Pallottino, M. P. Scaparra, and M. G. Scutellà. A multi-exchange heuristic for the single-source capacitated facility location problem. *Management Science*, 50(6): 749–760, 2004.

N. Anari, M. Fazli, M. Ghodsi, and M. Safari. Euclidean movement minimization. *Journal of Combinatorial Optimization*, 2015. to appear.

A. B. Arabani and R. Z. Farahani. Facility location dynamics: An overview of classifications and applications. *Computers & Industrial Engineering*, 62(1):408–420, 2012.

A. Armon, I. Gamzu, and D. Segev. Mobile facility location: combinatorial filtering via weighted occupancy. *Journal of Combinatorial Optimization*, 28(2):358–375, 2012.

A. Bingham, A. Bishop, P. Coffey, J. Winkler, J. Bradley, I. Dzuba, and I. Agurto. Factors affecting utilization of cervical cancer prevention services in low-resource settings. *Salud publica de Mexico*, 45:408–416, 2003.

D. G. Cattrysse and L. N. Van Wassenhove. A survey of algorithms for the generalized assignment problem. *European Journal of Operational Research*, 60(3):260–272, 1992.

A. Ceselli and G. Righini. A branch-and-price algorithm for the capacitated p-median problem. *Networks*, 45(3):125–142, 2005.

C.-H. Chen and C.-J. Ting. Combining lagrangian heuristic and ant colony system to solve the single source capacitated facility location problem. *Transportation Research Part E: Logistics and Transportation Review*, 44(6):1099–1122, 2008.

M. J. Cortinhal and M. E. Captivo. Upper and lower bounds for the single source capacitated location problem. *European Journal of Operational Research*, 151(2):333–351, 2003.

E. Demaine, M. Hajiaghayi, H. Mahini, A. S. Sayedi-Roshkhar, S. Oveisgharan, and M. Zadimoghaddam. Minimizing movement. *ACM Transactions on Algorithms*, 5(3):30, 2009.

K. Doerner, A. Focke, and W. J. Gutjahr. Multicriteria tour planning for mobile healthcare facilities in a developing country. *European Journal of Operational Research*, 179(3):1078–1096, 2007.

Z. Friggstad and M. R. Salavatipour. Minimizing movement in mobile facility location problems. *ACM Transactions on Algorithms*, 7(3):28, 2011.

E. Geoffroy, A. D. Harries, K. Bissell, E. Schell, A. Bvumbwe, K. Tayler-Smith, and W. Kizito. Bringing care to the community: expanding access to health care in rural malawi through mobile health clinics. *Public Health Action*, 4(4):252–258, 2014.

G. Guastaroba and M. G. Speranza. A heuristic for BILP problems: The single source capacitated facility location problem. *European Journal of Operational Research*, 238(2):438–450, 2014.

M. H. Ha, L.-A. Bostel, N., and L.-M. Rousseau. An exact algorithm and a metaheuristic for the multi-vehicle covering tour problem with a constraint on the number of vertices. *European Journal of Operational Research*, 226(2):211–220, 2013.

R. Halper, S. Raghavan, and M. Sahin. Local search heuristics for the mobile facility location problem. *Computers & Operations Research*, 62:210–223, 2015.

K. Holmberg, M. Rönnqvist, and D. Yuan. An exact algorithm for the capacitated facility location problems with single sourcing. *European Journal of Operational Research*, 113(3):544–559, 1999.

A. Klose. An LP-based heuristic for two-stage capacitated facility location problems. *The Journal of the Operational Research Society*, 50(2):pp. 157–166, 1999.

A. Klose and S. Görtz. A branch-and-price algorithm for the capacitated facility location problem. *European Journal of Operational Research*, 179(3):1109–1125, 2007.

H. W. Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2:83–97, 1955.

L. A. N. Lorena and E. L. F. Senne. A column generation approach to capacitated p-median problems. *Computers & Operations Research*, 31(6):863–876, 2004.

M. T. Melo, S. Nickel, and F. Saldanha da Gama. Dynamic multi-commodity capacitated facility location: a mathematical modeling framework for strategic supply chain planning. *Computers & Operations Research*, 33(1):181–208, 2006.

S. Nickel and F. Saldanha da Gama. *Location Science*, chapter Multi-Period Facility Location, pages 289–310. Springer, 2015.

M. Savelsbergh. A branch-and-price algorithm for the generalized assignment problem. *Operations Research*, 45(6):831–841, 1997.

F. Stefanello, O. C. B. de Araújo, and F. M. Müller. Matheuristics for the capacitated p-median problem. *International Transactions in Operational Research*, 22(1):149–167, 2015.

J. E. Torres-Soto and H. Uster. Dynamic-demand capacitated facility location problems with and without relocation. *International Journal of Production Research*, 49(13):3979–4005, 2011.

S. Tragantalerngsak, J. Holt, and M. Rönnqvist. An exact method for the two-echelon, single-source, capacitated facility location problem. *European Journal of Operational Research*, 123(3): 473–489, 2000.

V. Verter and S. D. Lapierre. Location of preventive health care facilities. *Annals of Operations Research*, 110(1-4):123–132, 2002.

J. E. Weiss, M. R. Greenlick, and J. F. Jones. Determinants of medical care utilization: The impact of spatial factors. *Inquiry*, 8(4):50–57, 1971.

Z. Yang, F. Chu, and H. Chen. A cut-and-solve based algorithm for the single-source capacitated facility location problem. *European Journal of Operational Research*, 221(3):521–532, 2012.