A bi-level branch-and-bound algorithm for the capacitated competitive facility location problem

Vahid Mahmoodian, Hadi Charkhgard, Yu Zhang
University of South Florida, Tampa, FL

Abstract

Competitive facility location problem is a typical facility locating optimization problem but in a competitive environment. The main characteristic of this problem is the competitive nature of the market. In essence, the problem involves two competitors, i.e., a leader and a follower, who seek to attract customers by establishing new facilities to maximize their own profit. In this research, the capacitated competitive facility location problem is considered in which customers may be rejected by a facility because of capacity limitation. A branch-and-bound algorithm is proposed to solve this problem. An extensive computational study is provided to show the performance of the algorithm.

Keywords
Capacitated competitive facility location, Bi-level, Branch-and-bound algorithm, Game theory

1. Introduction
Competitive facility location (CFL) is defined as the problem of finding the best location for facilities by taking the existence of rivals and their effects into account. One of the first mathematical models for duopoly was proposed by Hotelling [7]. His model was modified later and became a basis for CFL studies (see for instance [4] and [5]). A typical competitive facility location problem involves a leader and a follower. This type of problems have been studied for a long time in the literature of Game theory. Depending on whether the leader and follower (players) make their decision sequentially or simultaneously, the model will follow Stackelberg equilibrium or Nash equilibrium, respectively. The focus of this paper is on the Stackelberg based CFL models. However, one may refer to Nash equilibrium based CFL models [6], [8] , and [11].

Although many studies have attacked CFL problem, just a few of them could tackle it with an exact solution method ([1, 3]).One of basic developments on basic competitive facility location problems is an uncapacitated model introduced first by Beresnev [1]. In this model the author has assumed that if a facility is the most attractive one for a set of customers, they will definitely provide their demand from it. This assumption as a constraint in the model makes the most attractive potential points infeasible if the built facility does not have enough capacity for all referred customers. While, in the real world, companies never miss these points and just try to manage the demands. In another study, Beresnev and Melnikov [2] generalize this model for the case that the facilities have limited capacity holding the same assumption and solve it with an effective branch-and-bound algorithm. In this model, again, the customer selects the most preferable facility, no matter what the company is. Moreover, if the facility would not have enough capacity to provide all referred demands, the additional ones will just be ignored from the model.

The recent model creatively solves the capacity issue existed in the first model. However, ignoring unsatisfied demand points raises another issue. Although in this model the customers are constrained to the closest facility and cannot be assigned to other facilities from none of companies, they will look for another facility to provide their demand in the real world when they get rejected from one facility. Allowing them to be assigned to facilities other than the closest one when the closest one does not have enough capacity, can dramatically change the solution. For example, in Figure 1 let the leader's facility (established in point 1) has the capacity to provide only 3 demand points (customers) other than the point facility itself is located on. Also, assume point 2 is an economical potential point for follower's company if more than three demand points go to it. As it is shown in the picture, in Beresnev and Melnikov's model [2], since point 3 is closer to

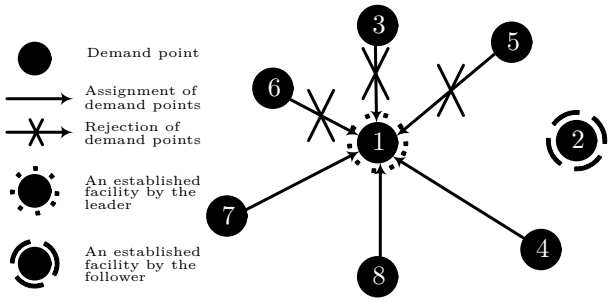

Figure 1: The issue that arises when the customers are forced to be assigned to the closest facility.

point 1, it will be just ignored in current solution. In addition, point 2 will not be economical for the follower because there is not three customers around free to go to it (because of the constraint which assigns them to the closest facility). While, if demand point 3 was free to go to the other closest facility, in presence of the follower's facility in point 2, demand point 3, 4, and 5 would refer to the facility in point 2. As a result, not only would establishing a facility in point 2 be economical for the follower, but also customer 4 which was supposed to be served by the leader's facility would be stolen.

We address this issue in a research and propose a model which solves the problem [10]. This model allows rejected customers to be served by other facilities, even though they are farther. However, it approximately solves the problem by a Genetic Algorithm. In this study, an exact branch-and-bound algorithm is developed to solve a modified of version of this model.

The structure of the paper is organized as follows: In Section 2, the model is introduced. Section 3 contains the explanation of the proposed branch-and-bound algorithm, and Section 4 includes the numerical results. Finally, in Section 5, some concluding remarks are provided.

2. Problem definition

In this section, we introduce the proposed CFL model in our previous research [10] with a fine modification. The notations are represented in Table 1

Table 1: The notation used in the model.

Sets:	
$\mathcal{I}$	set of all potential sites for establishing new facilities, $i \in \{1,2,...,m\}$
$\mathcal{I}^{\mathcal{L}}$	set of potential sites for establishing new facilities by the leader, $\mathcal{I}^{\mathcal{L}} \subset \mathcal{I}$
$\mathcal{I}^{\mathcal{F}}$	set of potential sites for establishing new facilities by the follower, $\mathcal{I}^{\mathcal{F}} \subset \mathcal{I}$
$\mathcal{J}$	set of customers, $\mathcal{J} \in \{1,2,...,n\}$
$\mathcal{S}_i$	set of facility capacities that can be established at site $i, s \in \mathcal{S}_i = \{1,2,...,k_i\}$
Parameters:	
p_{ij}	the income realized by the leader through establishing a facility at site i for satisfying the demand of customer j
q_{ij}	the income realized by the follower through establishing a facility at site i for satisfying the demand of customer j
w_j	the demand of customer j
f_{is}	the fixed cost of establishing a facility of capacity $s \in \mathcal{S}_i$ by the leader at site i
g_{is}	the fixed cost of establishing a facility of capacity $s \in \mathcal{S}_i$ by the follower at site i
lc_{is}	the maximum amount of demand can be satisfied by a facility of capacity $s \in S_i$ located at the site i by the leader
fc_{is}	the maximum amount of demand can be satisfied by a facility of capacity $s \in S_i$ located at the site i by the follower
d_{ij}	the distance beteen the customer j and site i
dc_{ij}	the transportation cost of satisfying the whole demand of customer j from facility at site j
Decision variables:	

x_{is}	if the leader establishes site i of capacity $s \in S_i$ 1, and otherwise 0,
δij	if the demand of customer j is satisfied by facility i established by the leader 1, otherwise 0,
y_{is}	if follower establishes site i of capacity $s \in S_i$
θ_{ij}	if the whole or a portion of the demand of customer j is satisfied by facility i established by the follower 1, otherwise 0,

2.1 Formulation

$$\text{L(Leader):} \max \sum_{j\in\mathcal{J}} \Big(\sum_{i\in\mathcal{I}^{\mathcal{L}}} (p_{ij} - dc_{ij})\delta_{ij}\Big)(1 - \sum_{i\in\mathcal{I}^{\mathcal{F}}} \widetilde{\theta}_{ij}) - \sum_{i\in\mathcal{I}^{\mathcal{L}}, s\in S_i} f_{is}x_{is} \tag{1}$$

$$\sum_{i\in\mathcal{I}^{\mathcal{L}}} \delta_{ij} \leq 1 \qquad \forall j \in \mathcal{J} \tag{2}$$

$$\sum_{j\in\mathcal{J}} w_j\delta_{ij} \leq \sum_{s\in S_i} lc_{is}x_{is} \qquad \forall i \in \mathcal{I}^{\mathcal{L}} \tag{3}$$

$$x_{is}, \delta_{ij} \in \{0,1\} \qquad \forall i \in \mathcal{I}^{\mathcal{L}}, \forall j \in \mathcal{J}, \forall s \in S_i \tag{4}$$

$$\text{F(Follower):} \max \sum_{i\in\mathcal{I}^{\mathcal{F}}, j\in\mathcal{J}} (q_{ij} - dc_{ij})\theta_{ij} - \sum_{i\in\mathcal{I}^{\mathcal{F}}, s\in S_i} g_{is}y_{is} \tag{5}$$

$$\sum_{i\in\mathcal{I}^{\mathcal{F}}} \theta_{ij} \leq 1 \qquad \forall j \in \mathcal{J} \tag{6}$$

$$\sum_{j\in\mathcal{J}} w_j\theta ij \leq \sum_{s\in S_i} fc_{is}y_{is} \qquad \forall i \in \mathcal{I}^{\mathcal{F}} \tag{7}$$

$$\sum_{s\in S_i} x_{is} + \sum_{s\in S_i} y_{is} \leq 1 \qquad \forall i \in (\mathcal{I}^{\mathcal{L}} \cap \mathcal{I}^{\mathcal{F}}) \tag{8}$$

$$\theta_{ij} \leq 1 - \delta_{i'j} \qquad \forall j \in \mathcal{J}, \forall i \in \mathcal{I}^{\mathcal{F}}, \forall i' \in \mathcal{I}^{\mathcal{L}}; d_{ij} \geq d_{i'j} \tag{9}$$

$$y_{is}, \theta_{ij} \in \{0,1\} \qquad \forall i \in \mathcal{I}^{\mathcal{L}}, \forall j \in \mathcal{J}, \forall s \in S_i \tag{10}$$

The proposed formulation, i.e. (1)- (10) this formulation represents a bi-level model for optimizing the profit of the leader in which (5-10) subject to constraints resulting from optimal reaction of the follower. It should be mentioned that $\widetilde{\theta}_{ij}$ denotes the optimal assignment of customers to follower's facilities given the the leader's decision. Objective (1) is the objective function of leader and maximizes the total revenue maintained by satisfying customers' demand minus the total cost associated with that. Constraint (2) guarantees that each customer at most is assigned to one established facility. Constraint (3) is forcing the model to consider capacity of facilities. Objective (5) maximizes the follower's profit and (6-7) are associated to the same constraints in leader's model respectively (2-4). Constraint (8) makes the common potential places available for only one of the competitors, and constraint (9) prevents a customer to be stolen by follower if it has already been assigned to one of the leader's closer facilities.

3. A branch-and-bound based solution method

Branch and Bound algorithm is a smart scheme for complete enumeration introduced by Land and Doig [9] for the first time and has been used to solve many mathematical optimization problems. In the following sections, the proposed branch-and-bound algorithm for proposed formulation is explained.

3.1 Bounds

In the proposed branch-and-bound algorithm, dual bound is obtained by removing the objective function of follower's model (5) and dropping the term $1 - \sum_{i\in\mathcal{I}^{\mathcal{F}}} \widetilde{\theta}_{ij}$ from the leader's objective function. In the context of bi-level models, optimality of any lower level model is basically a constraint for upper level models. Then, ignoring the follower's objective function (5) results in a relaxation for the whole model. Hence, the objective value of associated model which is referred to as Upper Bound Model (UBM) will be a dual bound for the complete model. Moreover, constraint (9) changes as follows:

$$\theta_{ij} \leq 1 - \delta_{i'j} \qquad \forall j \in \mathcal{J}, \forall i \in \mathcal{I}^{\mathcal{F}}, \forall i' \in \mathcal{I}^{\mathcal{L}}; d_{ij} \leq d_{i'j} \tag{11}$$

where the direction of the inequality in proximity condition $d_{ij} \leq d_{i'j}$ has changed. This is because of considering the effect of branching cuts on UBM. These cuts will be introduced in Section 3.2.

The follower's model while the leader's variables, i.e. (x, δ) are fixed is called Lower Bound Model (LBM). The lower bound in a node, is calculated by optimizing LBM for the obtained (x^*, δ^*) from UBM and considering its influence (explicitly by $\widetilde{\theta_{ij}}$) on the leader's objective function. For example, suppose (x^*, δ^*) is the output of UBM. The value of x_{is} and $\delta_{ij}, \forall i \in \mathcal{I}^{\mathcal{L}}, \forall s \in S_i, \forall j \in \mathcal{J}$ is fixed to (x^*, δ^*) in the follower's model and it is optimized. Then, substituting the optimal values for variable $\theta_{ij}, \forall i \in \mathcal{I}^{\mathcal{F}}, j \in \mathcal{J}$ in the leader's objective function will generate a lower bound for that node, i.e. $\sum_{j \in \mathcal{J}} \big(\sum_{i \in \mathcal{I}^{\mathcal{L}}} (p_{ij} - dc_{ij}) \delta^*_{ij} \big) (1 - \sum_{i \in \mathcal{I}^{\mathcal{F}}} \widetilde{\theta}_{ij}) - \sum_{i \in \mathcal{I}^{\mathcal{L}}, s \in S_i} f_{is} x^*_{is}$.

3.2 Branching

In developed branch-and-bound algorithm, branching is done by breaking down each node into two child nodes. The first node which will be referred to as *left-branch* is built by adding a cut to UBM, based on the stolen customers by follower. Stolen customers are the demand points which are being served (or assigned in LBM) by the follower company, despite it has already been assigned to the leader's facility (in UBM). For example, customer j^* is stolen by one of the follower's facilities in point $i^* \in \mathcal{I}^{\mathcal{F}}$ if $(\sum_{i \in \mathcal{I}^{\mathcal{L}}} \delta_{ij^*})(\theta_{i^*j^*}) = 1$. Added cut in each *left-branch* is as follows:

$$\sum_{(i,j) \in ST} \theta_{ij} \geq 1 \tag{12}$$

where ST is the set of ordered pairs which denotes the stolen customers in the parent node. Since there is a hope that feeding the follower's reaction to the leader's model can render the rival's existence and give a better upper bound, *left-branch*s are added to the front of the list of the nodes, i.e. they are in priority to be explored.

On the other side, another node is generated that will be referred to as *right-branch*. In this node, cut (13) is added to UBM. This cut changes the leader's decision by one of the binary variables.

$$\sum_{\{(i,j) | \overline{\delta}_{ij} = 0\}} \delta_{ij} + \sum_{\{(i,j) | \overline{\delta}_{ij} = 1\}} (1 - \delta_{ij}) \geq 1 \tag{13}$$

where $\overline{\delta}_{ij}, \forall i \in \mathcal{I}^{\mathcal{F}}, j \in \mathcal{J}$ indicates the value of the variable δ in a parent node. *Right-branch*s are added to the end of the list, that is they have the lowest priority to be explored.

3.3 Algorithm

In this section the steps of the branch-and-bound algorithm is described. Each node encompasses features $(LB, UB, \overline{x}, \overline{\delta}, ST, BC)$ which denote lower bound, upper bound, obtained values of variables x and δ, stolen customers, and set of branching cuts corresponding to the node's UBM, respectively. Algorithm 1 explains the steps of the proposed branch-and-bound algorithm. In this algorithm GLB and GUB represents the global lower bound and global upper bound of the algorithm.

4. Numerical results

Introduced branch-and-bound algorithm in section 4 is implemented in C++, and CPLEX 12.7 is employed to solve UBMs and LBMs. Moreover, all the experiments are done using a Dell PowerEdge R630 with two Intel Xeon E5-2650 2.2 GHz 12-Core Processors (30MB), 128GB RAM, and the RedHat Enterprise Linux 6.8 operating system, while only one threat is activated.

4.1 Instance generation

All instances used in the numerical experiments are generated randomly. d_{ij} is the Euclidean distance between points uniformly scattered over a rectangle surface with length and width 100 units. All other parameters follow a uniform distribution; $dc_{ij} \sim U(0.7, 1) \times d_{ij}, w_j \sim U(300, 500), p_{ij} \sim U(0.9, 3) \times w_j, q_{ij} \sim U(0.8, 2.5) \times w_j, f_{is} \sim U(100, 500), g_{is} \sim U(100, 500), lc_{is} \sim U(600, 1000), fc_{is} \sim U(600, 1000)$.

Algorithm 1: The proposed algorithm

Result: $BestNode, GUB, GLB$
Initialize parameters of the model, $T \leftarrow \emptyset, BC \leftarrow \emptyset$
$GLB \leftarrow 0$
$node.BC \leftarrow \emptyset$
$T \leftarrow node$
While Termination_Condition is not met
  Pop out the first node from T into $node$
  Explore($node$)
  Update GLB and GUB
  if $GLB = GUB$ return $BestNode$
  $left-branch.BC \leftarrow node.BC \cup$ cut (12) based on $node.ST$
  $rigth-branch.BC \leftarrow node.BC \cup$ cut (13) based on $node.\overline{\delta}$
  $[left-branch, right = branch] \leftarrow Branch(node)$
  $T \leftarrow left\text{-}branch \cup T$
  $T \leftarrow T \cup right\text{-}branch$
End

Algorithm 2: *Explore(node)*

Generate UBM based on the $node.BC$
Solve UBM
If UBM is infeasible prune $node$
$node.UB \leftarrow UBM$.getObjectiveValue
$node.\delta \leftarrow UBM$.getValue($\delta$)
$node.x \leftarrow UBM$.getValue(x)
Generate LBM based on the $node.x$ and $node.\delta$
Solve LBM
$node.LB \leftarrow UBM.getValue(Objective)$
Update $node.ST$
If $node.UB = node.LB$ Update $BestNode$ and fathom $node$

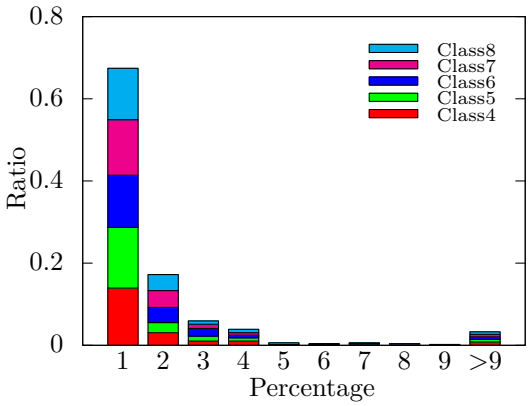

Figure 2: Distribution of improvement in lower bound for four classes

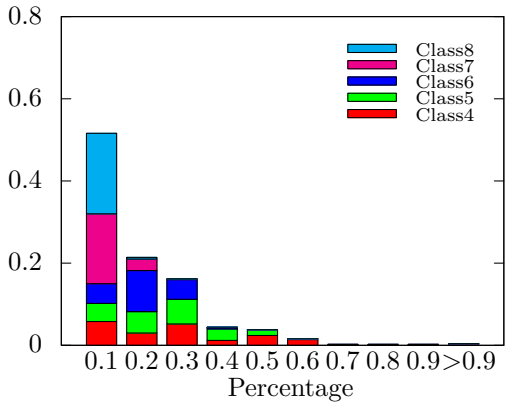

Figure 3: Distribution of improvement in upper bound for four classes

4.2 Results

Proposed algorithm is used to solve a set of small and large instances. small instances include 5 classes of size 4, 5, 6, 7, and 8 where the size shows the number of potential demand points, i.e., $\mathcal{I}^{\mathcal{L}} = \mathcal{I}^{\mathcal{F}} = \mathcal{J}$. Large instances are divided into 4 classes including sizes 10, 20, 30, and 40. It should be mentioned that each class has 100 instances, and all of the experiments are limited to 900 seconds of CPU run time.

The algorithm was able to solve 100, 94, 36, 12, and 6 instances out of 100 instances in classes 4, 5, 6, 7, and 8 to optimality, respectively. The distribution of improvements in the lower bound for small instances has been plotted in 2. This graph demonstrates that the algorithm can improve the lower bound from 1 to 9 percent (and rarely more than 10%). A similar graph for upper bound (see Figure 3) shows that a small change in upper bound. The main reason for this observation is the branching strategy of the algorithm. In specific, the algorithm steps back from an optimal solution of the relaxed problem and searches among multiple optimal solutions by adding cuts until it finds an optimal solution for the main problem. Therefore, despite the algorithm was able to solve almost half of instances up to optimality in a limited time, there is a little improvement in the upper bound. Moreover, during the course of this study we observed that the complexity of the problem dramatically increases as the size grows. Exponential decrease in the number of optimally solved problems while the size of the problem increases by just one confirms this observation. So that, in sizes greater than 10 the algorithm is rarely able to converge to an optimal solution.

Box-plot of gaps between final upper and lower bounds for four classes of large scale problems has been plotted in 4. One of the observations in this graph is the increase in stability of algorithm's performance as the size of the problem grows so that there is a negligible jump when the size of the problem has increased from 30 to 40.

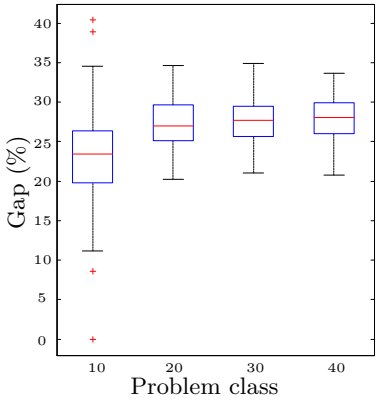

Figure 4: Box plot for gap in solved instances with initial lower bound

It is worth mentioning that the average (relative) optimality gap for small and large problems is 8.8% and 26.44%, respectively. For 82.25% of the large scale instances, the gap is below 30%. In order to show the stability of the optimality gap, we also attempted to solve 100 instances with size 100 within one hour of time limit. The results show 28.32% of optimality gap on average and a maximum gap of 33.60%.

5. Conclusion

In this research, We studied a capacitated competitive facility location problem after dropping one of the critical assumptions in the literature, which is forcing the customers to be assigned to the most attractive facility. Although the resulted problem is more difficult than the ones in the literature, we were able to formulate it as a bi-level integer program and develop a novel branch-and-bound algorithm for it. The numerical results show that the proposed algorithm's performance stays stable as the size of the instances increases, and an optimality gap around 30% is achievable even for large-sized instances.

References

[1] Beresnev, V., 2009, "Upper bounds for objective functions of discrete competitive facility location problems," Journal of Applied and Industrial Mathematics, 3(4), 419.

[2] Beresnev, V., Melnikov, A., 2018, "Exact method for the capacitated competitive facility location problem," Computers & Operations Research, 95, 73-82.

[3] Beresnev, V., 2014, "On the competitive facility location problem with a free choice of suppliers," Automation and Remote Control, 75 (4), 668-676.

[4] Eiselt, H. A., Laporte, G., Thisse, J.-F., 1993, "Competitive location models: A framework and bibliography," Transportation Science, 27 (1), 44-54.

[5] Friesz, T. L., Miller, T., Tobin, R. L., 1988, "Competitive network facility location models: A survey," Papers in Regional Science, 65 (1), 47-57.

[6] Godinho, P., Dias, J., 2013, "wo-player simultaneous location game: Preferential rights and overbidding," European Journal of Operational Research, 229 (3), 663-672.

[7] Hotelling, H., 1990, Stability in Competition, Springer, New York, 50-63.

[8] Konur, D., Geunes, J., 2012, "Competitive multi-facility location games with non-identical firms and convex traffic congestion costs," Transportation Research Part E: Logistics and Transportation Review, 48 (1), 373-385.

[9] Land, A. H., Doig, A. G., 1960, "An automatic method of solving discrete programming problems," Econometrica: Journal of the Econometric Society, 497-520.

[10] Nasiri, M. M., Mahmoodian, V., Rahbari, A., Farahmand, S., 2018, "A modified genetic algorithm for the capacitated competitive facility location problem with the partial demand satisfaction," Computers & Industrial Engineering, 124, 435-448.

[11] Shan, W., Yan, Q., Chen, C., Zhang, M., Yao, B., Fu, X., 2017, "Optimization of competitive facility location for chain stores," Annals of Operations Research, 1-19.