Development of a Branch and Bound Algorithm for One Competitive Facility Location Problem with Elastic Demand

T Levanova, A Gnusarev
Sobolev Institute of Mathematics, Omsk Branch, Omsk, Russia
E-mail: levanova@ofim.oscsbras.ru, alexander.gnussarev@gmail.com

Abstract. We consider a variant of the Competitive Location Problem, in which it is necessary to choose not only location, but also the design of facilities in order to maximize the share of customer demand served. It is described as an integer programming model with a nonlinear objective function. Commercial software is not suitable for finding the optimal solution to this problem in a reasonable time. Previously, Aboolian R. et al. proposed an adapted weighted greedy heuristic for this problem. In our earlier study, we developed several variants of local search algorithm. This article is devoted to development of the branch-and-bound algorithm. The scheme of branching is described, the results of a computational experiment are presented, future development possibility is discussed.

1. Introduction
The competitive facility location and design problem with elastic demand seeks to locate new facilities in a competitive market so that the captured share of customer demand is maximized. It is assumed that each client selects among all available facilities in accordance with the so-called gravity rule. Thus, we can talk about elastic demand, meaning the changing structure of the served demand for companies. Aboolian, Berman, Krass [2] applied an exponential function to describe elastic demand. This nonlinear objective function of the mathematical model complicates the search for its solution. Earlier numerical experiments have shown that the known software does not guarantee finding the optimal solution, and on problems of medium dimension, it does not always find even a feasible solution [2]. In our previous papers, we developed Variable Neighborhood Search algorithms, Threshold Algorithms and Ant Colony Optimization Algorithm for this problem [7, 8, 9]. Here we introduce a new branch-and-bound algorithm to find the optimal solution to the problem under consideration.

The paper is organized as follows. In Section 2, the notation and mathematical model for the competitive facility location and design problem with elastic demand are presented. In Section 3, a new branch-and-bound algorithm is introduced. The computational results are described in Section 4. Finally, some conclusions are drawn and opportunities for future research are discussed. An example of how the algorithm works is described in the Appendix.
2. Problem Formulation

Let us describe the problem in the interpretation of [2]. The situation in the competitive market is considered when a new company plans to open its own businesses, choosing locations and design options for its facilities. Each client chooses among all available facilities according to random rule depending on the attractiveness of the facilities and the distance to them. The company’s goal is to serve the largest share of total customer demand for the highest profit.

Let us construct a mathematical model for the competitive facility location and design problem with elastic demand according to [2]. The following sets are setting:

- \( N \) is a discrete set of demand points, where a facility can be located;
- \( C \subset N \) are the competitor’s points;
- \( S = N \setminus C \) are the points of possible location of the new company;
- \( R \) is a set of facility types.

Introduce the following notation:

- \( w_i \) is the demand weight at point \( i \in N \);
- \( c_{jr} \) is the opening cost of a facility in case \( r \in R \) at point \( j \in S \).

The problem variable \( x_{jr} \) is equal to 1, if a facility of the type \( r \in R \) is located in point \( j \in S \), and \( x_{jr} = 0 \) otherwise. The auxiliary coefficients \( k_{ijr} = a_{jr}(d_{ij} + 1)^{-\beta} \) are constructed, they depend on the sensitivity \( \beta \) of customers to distance to facility and attractiveness \( a_{jr} \). The utility for the customers \( j \in S \) in point \( i \in N \) is calculated by the formula \( u_{ij} = \sum_{r=1}^{R} k_{ijr} x_{jr} \).

The total utility \( U_i(S) \) from the facilities opened by the company in points \( i \in N \) is calculated as follows: \( U_i(S) = \sum_{j \in S} u_{ij} \). The notation for the competitors are similar. The total utility for the customers in point \( i \in N \) from the facilities controlled by the competitors is \( U_i(C) = \sum_{j \in C} u_{ij} \).

The demand function is \( g(U_i) = 1 - \exp\left(-\lambda_i U_i\right) \), where \( U_i = U_i(S) + U_i(C) \) is the total utility from all the company’s and competitor’s facilities for a customer at \( i \in N \), \( \lambda_i > 0 \) is the characteristic of the elastic demand in point \( i \). The company’s total share of facility \( i \in N \) is measured by:

\[
MS_i = \frac{U_i(S)}{U_i(S) + U_i(C)} = \frac{\sum_{j \in S} \sum_{r \in R} k_{ijr} x_{jr}}{\sum_{j \in S} \sum_{r \in R} k_{ijr} x_{jr} + U_i(C)}.
\]

According to the notations above, the mathematical model looks like:

\[
\max \sum_{i \in N} w_i \cdot \left(1 - \exp\left(-\lambda_i \left(\sum_{j \in S, r \in R} k_{ijr} x_{jr} + U_i(C)\right)\right)\right) \cdot MS_i
\]

\[
\sum_{j \in S, r \in R} c_{jr} x_{jr} \leq B,
\]

\[
\sum_{r \in R} x_{jr} \leq 1, j \in S,
\]

\[
x_{jr} \in \{0, 1\}, \quad j \in S, r \in R.
\]

Function (1) represents the goal of the new company: To capture the largest share of the demand of all the customers. Condition (2) says that every customer spends the budget proportionally to the utility either from the Company’s facilities or from a competitor. Inequality (3) shows that only one type of facilities’ design can be selected for location at \( j \in S \). Condition (4) ensures that the problem variables are integer. It is known that the location problem considered in this paper is NP-hard [3].
3. Branch and Bound Algorithm

The branch-and-bound method is a well-known approach to solving a wide range of difficult problems, including facility location problems (see for example [1, 4, 5, 10]). Depending on the implementation, it allows to find both exact and approximate solutions. In this paper we present an exact branch-and-bound algorithm for the competitive facility location and design problem with elastic demand. Traditionally, when developing such algorithms, the main efforts are focused on building bounds and choosing a branching scheme. Let us describe these two main characteristics.

3.1. Upper and Lower Bounds

In this problem, the choice of customers is influenced not only by the location, but also by the facility design. In the article [2], a study was conducted on how the values of λ affect the decision making of clients. It is noted that with 0 < λ ≤ 1, customers are more sensitive to what projects are selected for facilities, and demand is elastic. Earlier, we proposed a way to construct an upper bound (UB) of the objective function of the problem (1)-(4) with elastic demand [7] which is also considered in this paper. The main idea is to reformulate a nonlinear problem into a model of mixed integer linear programming that can be solved using well-known software. Let us recall this technique here.

Let us consider the objective function (1) of a location and design problem. When λ is close to 1, the multiplier of the objective function behaves as a constant:

$$1 - \exp\left(-\lambda_i \left( \sum_{j \in S} \sum_{r \in R} k_{ijr} x_{jr} + U_i(C) \right) \right) \approx 1.$$  

Then the initial problem is equivalent to the following one:

$$\max \sum_{i \in N} w_i \cdot \left( \frac{\sum_{j \in S} \sum_{r \in R} k_{ijr} x_{jr}}{\sum_{j \in S} \sum_{r \in R} k_{ijr} x_{jr} + U_i(C)} \right),$$  

(5)

$$\sum_{j \in S} \sum_{r \in R} c_{jr} x_{jr} \leq B,$$  

(6)

$$\sum_{r \in R} x_{jr} \leq 1, \quad j \in S,$$  

(7)

$$x_{jr} \in \{0, 1\}, \quad r \in R, j \in S.$$  

(8)

Now it can be reduced to the mixed integer linear problem. To do this, we represent the objective function (5) as follows:

$$\sum_{i \in N} w_i \cdot \left( \frac{U_i(S)}{\sum_{j \in S} U_i(S) + U_i(C)} \right) =$$

$$= \sum_{i \in N} \sum_{j \in S} \sum_{r \in R} \frac{w_i k_{ijr} x_{jr}}{\sum_{j \in S} \sum_{r \in R} k_{ijr} x_{jr} + U_i(C)}.$$  

Let us use the known method of linearization. To do this, define

$$z_{ijr} = \frac{w_i k_{ijr} x_{jr}}{\sum_{j \in S} \sum_{r \in R} k_{ijr} x_{jr} + U_i(C)}, i \in N, j \in S, r \in R.$$  

Introduce supplementary variables:

$$y_i = \frac{w_i}{\sum_{j \in S} \sum_{r \in R} k_{ijr} x_{jr} + U_i(C)}, i \in N.$$  

$$3$$
Then $z_{ij}$ must satisfy the inequalities:

\begin{align*}
  k_{ijr}y_i + m(x_{jr} - 1) & \leq z_{ijr} \leq k_{ijr}y_i, \\
  z_{ijr} & \leq x_{jr}w_i,
\end{align*}

where $m = \max\{\frac{w_i k_{ijr}}{U_i(C)}\}$, $i \in N$, $j \in S$, $r \in R$. \hfill (9)

The nonlinear model (1)-(4) is reduced to the following linear model:

\begin{align*}
  \max \sum_{i \in N} \sum_{j \in S} \sum_{r \in R} z_{ijr}, \hfill (10) \\
  k_{ijr}y_i + m(x_{jr} - 1) & \leq z_{ijr} \leq k_{ijr}y_i, \quad i \in N, j \in S, r \in R, \hfill (11) \\
  z_{ijr} & \leq x_{jr}w_i, \quad i \in N, j \in S, r \in R, \hfill (12) \\
  \sum_{r \in R} \sum_{j \in S} z_{ijr} + y_i \cdot U_i(C) & = w_i, \quad i \in N, \hfill (13) \\
  \sum_{j \in S} \sum_{r \in R} c_{jr}x_{jr} & \leq B, \hfill (14) \\
  x_{jr} & \in \{0, 1\}, \quad j \in S, r \in R. \hfill (15)
\end{align*}

Problem (10)-(15) can be solved exactly, for example, using the CPLEX (GAMS) solver. Optimal value of function (10) can be used as an upper bound for goal function (1) of problem (1)-(4).

Lower bounds are also involved in the implementation of the branch and bound algorithm. They are usually used to reduce the number of vertexes being checked. A lower bound (LB) is the objective value of a feasible solution of the original problem obtained by the simulated annealing algorithm. This algorithm was developed and studied in our earlier paper [8].

### 3.2. The Scheme of Branching

In the proposed variant of branch-and-bound algorithm the best-bound search is used. The algorithm looks through the possible locations of the company’s facilities in some order. During one iteration, in the current point, the algorithm looks through all possible variants for locating the facilities, taking into account the available budget. For each variant, a branch is formed in the branching tree, the upper bound for a partial solution is calculated. Further on the next iteration, the leaf vertex with the highest value of the upper estimate of the branching tree is selected. Then $\tilde{B}$ units of new branches are formed at this vertex, where $\tilde{B}$ is a non-negative integer value of the current available budget. The process stops when there are no more vertices to view, this situation occurs if:

**a)** the upper bound is less than the lower one;

**b)** the entire budget is spent;

**c)** the entire branch is passed;

**d)** there are no more vertices for branching;

**e)** the values of the upper bound and the lower bound are the same.

In the case (a), the vertex is not promising for branching. In the cases (b) and (c), a new feasible solution is obtained. If it is greater than the lower bound, the value of the lower bound is updated, a new iteration begins. In the cases (d) and (e), the found solution is optimal, the branch-and-bound algorithm stops; the optimal value of variables $x_{jr}$ of the problem (1)-(4) is found, that is, the optimal locations and type of company’s facilities are determined and the maximum share of the demand of all customers is measured. An illustration of the algorithm steps is provided in the Appendix.
Table 1. Results of numerical experiments

| №  | GAP   | BS   | V    | TBB  | TSA   | №  | GAP   | BS   | V    | TBB  | TSA   |
|----|-------|------|------|------|-------|----|-------|------|------|------|-------|
| 1  | 1.382 | 11   | 4129 | 15539| 14236 | 11 | 1.516 | 9    | 8584 | 31704| 36974 |
| 2  | 1.267 | 7    | 3306 | 9288 | 12482 | 12 | 1.409 | 12   | 4805 | 13211| 15283 |
| 3  | 1.302 | 7    | 4268 | 12842| 13407 | 13 | 1.206 | 15   | 1076 | 3980 | 3884  |
| 4  | 1.228 | 6    | 2896 | 9150 | 8590  | 14 | 1.245 | 10   | 1897 | 4723 | 5966  |
| 5  | 1.297 | 11   | 4647 | 12311| 17029 | 15 | 1.332 | 11   | 3824 | 12608| 12608 |
| 6  | 1.234 | 7    | 1092 | 3392 | 3405  | 16 | 1.309 | 11   | 4946 | 16986| 19130 |
| 7  | 1.236 | 14   | 3358 | 13632| 13764 | 17 | 1.184 | 11   | 1133 | 3586 | 3711  |
| 8  | 1.192 | 9    | 1365 | 4692 | 5204  | 18 | 1.227 | 9    | 1206 | 3393 | 3474  |
| 9  | 1.301 | 9    | 3713 | 13723| 14261 | 19 | 1.296 | 10   | 2529 | 7830 | 8142  |
| 10 | 1.223 | 12   | 2552 | 6108 | 8470  | 20 | 1.330 | 8    | 4071 | 11800| 13222 |

4. Computational Experiments

The computational experiments were carried out on a computer with CPU Intel® Xeon® X5675 @ 3.07 GHz, 32 GB RAM. Algorithm implemented in C++, the Gurobi 7.0.2 solver was used to solve the linear model.

A series of test cases was built using real data from [2]. Distances between points are measured in the Euclidean metric. The series consists of the set of 20 instances with the dimension |N| = 60 with three possible designs and budget limits of 3. The parameter λ = 1, which means that demand is elastic. The sensitivity of consumers to the distance to facilities is inversely proportional to the value of this distance in the second degree. The sensitivity of consumers to the distance to enterprises is inversely proportional to the value of this distance in the second degree, that is, β = 2.

Two variants of numerical experiments were performed. In the first of them, at the beginning of the branch-and-bounds algorithm, only the upper bound of the entire set of solutions was calculated. In the second variant, the lower bound was obtained by the simulated annealing algorithm before the main process starts. Table 1 contains information about the results of numerical experiments. The table columns show:

- the number of the test instance in order, №;
- the number of vertices of the branching tree that were viewed, V;
- the number of updates to the value of the best solution found, BS;
- the differs of the initial upper bound from the optimal solution, GAP;
- the CPU time of branch-and-bound algorithm, TBB, sec.
- the CPU time of branch-and-bound algorithm with initial solution obtained by simulated annealing algorithm, TSA, sec.

The GAP value indicate how much the initial upper bound differs from the optimal solution and is calculated by the rule \(GAP = UB(\text{Root})/F_{opt}\). As the results of the numerical experiment show, the GAP value was similar for all the test instances of the series and varied from 1.184 to 1.516. On average, the upper bound \(UB(\text{Root})\) exceeds the value of \(F_{opt}\) by 1.210 times.

It is interesting to note that the number of vertexes viewed in both experiments was the same. We think this can be explained by the fact that in both cases the algorithm quickly gets to the same solution, and then the solution process occurs in the same way. The number of visited vertexes is changed in the range from 1076 to 8584, the average number of viewed vertexes is 3270 (see the column V). The number of updates to the value of the best solution found (BS) in the first experiment varies from 7 to 12 and is significantly less than the number of vertexes viewed.
In the second experiment, the branch and bound algorithm did not make any improvements to the initial record. This means that the solution found by the simulated annealing algorithm turned out to be optimal, and the branch and bound algorithm only proved its optimality.

We hoped that using the lower bound obtained by the simulated annealing algorithm at the beginning would speed up obtaining the optimal solution. However, in the series of test tasks under consideration, it is impossible to speak unambiguously about such an influence. We can say that the processor time in both experiments is comparable. It decreases on some tasks, and increases on others. The minimum CPU time for one example is 3392 sec., and the maximum is 36974 sec., and the average time for all 20 examples was 10516 sec. in the first experiment and 11662 sec. in the second one. Obviously, with increase in the number of visited vertices, the CPU time for a unit of test instances increases. It is interesting to note that the ratio of the CPU time to the number of viewed vertices is approximately the same and on average is 3.7. In test instances number 11 with the highest value of GAP, the maximum number of vertices was viewed and, as a result, the maximum processor time was spent. The quantity of viewed vertices in the other examples ranged from 12% to 57% relative to test instance № 11, and CPU time consists of 8% to 49%.

During the experimental studies, the behavior of the CoinBonmin solver was also observed. CoinBonmin does not guarantee finding the optimal solution, but it was useful in developing a simulated annealing algorithm when optimal solutions were not known. It’s interesting to know how much the results of CoinBonmin differ from the optimal solution. It turned out that its values are even greater than the upper bound. Current research has shown that for test instances number 5, 9, 13, 19 the GAP value between initial upper bound and CoinBonmin results is 1.306, 1.322, 1.232, and 1.340, respectively.

5. Conclusion
In this article, the competitive facility location and design problem with elastic demand is considered. The mathematical model of this problem is nonlinear, so the well-known software does not guarantee to find an optimum. In this work, a variant of branch-and-bound algorithm and a problem-oriented branching scheme are proposed. The constructed implementation of the algorithm can be used in the case of elastic demand, i.e. when $\lambda_i$ is close to 1. The results of the experimental study indicate that it is possible to find the optimal solution in a suitable time.

It should be noted that the research has led to new issues for discussion and testing. Some of them concern the construction of the upper bound. It turned out that the accuracy of the constructed estimate depends on the type of distance function between points. Two types of problems were studied in which distances are measured on a graph and in a Euclidean metric. Numerical experiments have shown that when using the first type, the value of the estimate is closer to the optimal solution of the problem (1)-(4) compared to the second [8]. Earlier for the case when demand is highly elastic ($\lambda_i$ is small, $\lambda_i << 1$), a different upper bound was constructed [6]. This bound is more accurate for the instances with euclidean distance. It is interesting to investigate the branch and bound algorithm for such examples with corresponding upper bounds. Besides the constant $m$ in (9) can be selected in various ways. This will also affect the accuracy of the upper bounds. Further research involves analyzing the application of the algorithm to test instances with different GAP values. It is useful to compare the implemented algorithm with its modification using a depth-first search. It would be interesting to continue the research in these directions.

Acknowledgments
The authors thank the reviewers for their attention and useful comments.

The work was supported by the program of fundamental scientific researches of the SB RAS I.5.1, project 0314-2019-0019.
Figure 1. Illustration of the solving process using the proposed branch and bound algorithm. Branching tree

References

[1] Aboolian, R., Berman, O., Krass, D. 2007 *Competitive Facility Location model with Concave Demand* (Eur. J. Oper. Res. 181) pp. 598–619

[2] Aboolian, R., Berman, O., Krass, D. 2007 *Competitive facility location and design problem* (Eur. J. Oper. Res. 182(1)) pp. 40–62

[3] Aboolian, R., Berman, O., Krass, D. 2013 *Capturing Market Share: Facility Location and Design Problem* (Int. Conf. on Discrete Optimization and Operations Research) pp. 7–11

[4] Beresnev V.L., Melnikov A.A. 2014 *The branch-and-bound algorithm for a competitive facility location problem with the prescribed choice of suppliers* (Journal of Applied and Industrial Mathematics 8(2)) pp. 177–189

[5] Freire A.S., Moreno E., Yushimito W.F. 2016 *A branch-and-bound algorithm for the maximum capture problem with random utilities* (Eur. J. Oper. Res. 252(1)) pp. 204–212

[6] Levanova, T., Belan, S. 2017 *Levanova T. V., Belan S. E. Application of upper bounds for analysis of approximate algorithms for a competitive location problem with elastic demand* (Herald of Omsk University. 4 (86)) pp. 4–10

[7] Levanova, T., Gnsarev, A. 2016 *Variable neighborhood search approach for the location and design problem* (In: Kochetov, Y.e.a. (ed.) DOOR-2016. LNCS, vol. 9869) pp. 570–577

[8] Levanova T., Gnsarev A. 2018 *Development of threshold algorithms for a location problem with elastic demand* (LNCS, vol. 10665) pp. 382–389

[9] Levanova T.V., Gnsarev A.Y. 2018 *Ant colony optimization for competitive facility location problem with elastic demand* (OPTA-SCL CEUR Workshop Proc., vol. 2008) pp. 239–248

[10] Zabudsky G.G., Veremchuk N.S. 2018 *Branch and bound method for the Weber problem with rectangular facilities on lines in the presence of forbidden gaps* (Communications in Computer and Information Science 871) pp. 29–41

Appendix

Let us illustrate the solution process with an example using the notation from section 1. Let 6 points of possible location of facilities be given, i.e. \(|N| = 6\). In each of these points, a client is located. The competitor’s facilities are located in points of set \(C = \{1, 2, 3\}\). Set \(S = \{4, 5, 6\}\) contains the numbers of possible locations of the company’s facilities. The available budget is 3 units, \(B = 3\). In the test instances from work [2], the opening cost of facilities is equal to the number of facility type. Thus, in a numerical experiment, the formula \(x_{s,b} = 1\) means that facility of type \(b\) is opened in point \(s\) and \(b\) budget unit have been spent. The approximation algorithm found a solution to the problem, where \(x_{s,3} = 1\), the other variables are equal to 0, and the corresponding value of the objective function of the problem (1)-(4) is \(LB = 5.0021\). Consider the solution process using Fig. 1. In this figure, the vertices are shown as rectangles.
Each rectangle contains the number of the vertex in order they are formed, the amount of budget to distribute, and the value of the upper bound. The input data of the algorithm is the upper bound for the entire set of solutions ($UB(\text{Root}) = 21.9812$) and the lower bound ($LB = 5.0021$) obtained by some approximation algorithm.

On the first iteration, the algorithm selects the first point from the set $S$, $i = 4$. At this point, 0, 1, 2, 3 budget units can be distributed. Vertices 2, 3, 4, 5 in the branching tree correspond to these actions. In vertex 5, the entire budget is spent, and a new valid solution is found: $x_{4,3} = 1$, the other variables are equal to 0, the corresponding value of the objective function is $F(5) = 10,0041$. Since $F(5) > LB$, the value of the lower bound is updated, $LB = 10,0041$. Among vertices 2, 3, 4, node number 3 has the highest upper bound $UB(1) = 21.9812$, so it is selected for branching in the second iteration, 1 budget unit was spent.

On the second iteration, current available budget is equal to $\tilde{B} = 2$, the algorithm selects the second point from the set $S$, $i = 5$. In vertex 8, the entire budget is spent, and a new valid solution is found: $x_{4,1} = x_{5,2} = 1$, the other variables are equal to 0. The value of the objective function is $F(8) = 16,4487$, which is over than current $LB$. The new value of the lower bound becomes $LB=16,4487$. Vertex number 2 is excluded from further consideration because $UB(5) < LB$. Vertex number 7 has the highest upper bound of all non viewed vertices ($UB = 21.9812$), so it is a candidate for branching in the next iteration.

On the third iteration, the algorithm selects the next point from the set $S$, $i = 6$, available budget is equal to $\tilde{B} = 1$. Form vertices 9 and 10, they are the last ones on this branch of the tree, and new feasible solutions are built. Vertex number 10 has the highest upper bound ($UB(10) = 21.9812$), a new feasible solution is $x_{4,1} = x_{5,1} = x_{6,1} = 1$. The corresponding value of the objective function is equal to $F = 20,8904$, this is the new best found solution. Since the upper bounds on all unverified vertices are less than the best found solution $F$, the optimum $F_{\text{opt}} = F = 20,8904$ is obtained.