# Algorithms for Placing and Connecting Facilities and their Comparative Analysis

Klara Kedem, Irina Rabaev and Neta Sokolovsky

Ben-Gurion University of the Negev, Beer-Sheva 84105, Israel

## 1 Introduction

Base stations are fixed stations used to send, receive and transmit signals. Each base station consists of a tower, communication equipment and antenna(s). An antenna transmits and receives radio waves. A base station serves users in a specific region defined by the area spanned by the antenna(s). Usually these regions are circular, but other considerations can be taken into account, such as topographic data, propagation model, number of covered customers, etc. If two base stations are not in the range of each other due to transmission constrains, we may use a relay stations to connect them. A relay station is a station with equipment to receive a signal and retransmit it. It is mounted on a tower or on an existing base station.

In this paper we discuss algorithms for placing base stations (also referred to as *facilities* or *servers*) and their interconnection: "Given a set of customers and a set of potential locations for base stations, pick the minimum number of base stations that serve all the customers and then connect the chosen base stations." This problem is referred to in the literature as the Connected Facility Location Problem [1, 2, 4]. We divide the problem into two sub problems and solve each one separately. The first sub problem is to choose a set of base stations that serves all the customers. We present the Integer Programming method that yields the optimal solution for this problem and then review three approximation solutions (Section 2). The second sub problem deals with adding connectivity between the chosen facilities. Here we present two heuristics: one is based on Dynamic Programming and the other is based on Integer Programming (Section 3). We have implemented all the algorithms discussed in the paper, the description of our implementation and the experimental results are given in Section 4.

## 2 Finding the set of facilities that serve all the customers

In this section we discuss the algorithms for finding the smallest set of base stations that serves **all** the customers. Given $n$ locations where base stations can be placed, and given $m$ customers, the goal is to build the minimal number of base stations, such that their union serves all the customers. ¿From the geometric point of view: there is a set of customers represented by points $P = \{p_1, p_2, \ldots, p_m\}$ and a set of regions $R = \{r_1, r_2, \ldots, r_n\}$. Pick the smallest number of regions such that their union covers all the customers. We discuss four algorithms that solve the stated problem.

### 2.1 The optimal solution

We use Integer Programming to find the optimal set of facilities that serves all the customers. The variables $x_i$, $i = 1, \ldots, n$, in our problem represent binary decisions whether to build a base station at a given location or not, where a positive decision is represented by 1, and a negative decision is represented by 0.

Let us formulate the constraints of the problem. Denote by $T_i$ the set of base stations that can serve customer $p_i$, $i = 1 \ldots m$, (the customer $p_i$ is in their range). At least one base station should serve customer $p_i$, therefore at least one variable from $T_i$ is assigned the value 1. Thus, we have the constrains $\sum_{x_j \in T_i} x_j \geq 1, \forall\, i \in \{1, \ldots, m\}$ . The cost of the solution is the total cost of the selected servers, so the goal is to minimize $\sum_i c_i x_i$, where $c_i$ is the cost of building base station $x_i$.

### 2.2 The greedy algorithm

Our first sub problem is actually the Set Cover problem. One method to approximate the Set Cover problem is the greedy algorithm. It is quite straightforward: at each step choose the region

with the maximum number of remaining customers in it. Erase these customers from $P$ and proceed iteratively. The algorithm continues until all customers have been covered. Performance ratio of the greedy algorithm is $O(\ln m)$ [3, 5].

We implemented this greedy algorithm, but improved its output in order to get rid of redundant base stations. In order to find the redundant servers, we divide the solution set into connected components (two servers $a$ and $b$ are connected if and only if there exist servers $a = x_1, x_2, \ldots, x_k = b$, $k \geq 2$, such that for $1 \leq i \leq k-1$ the regions spanned by $x_i$ and $x_{i+1}$ intersect). For each base station $x_i$ in the connected component $C$ we check if for each customer $p$ in its range, there exists another base station $x_j$ in $C$ that covers $p$. If it is so, the server $x_i$ is redundant and we discard it.

### 2.3 A randomized algorithm

In this section we discuss a randomized algorithm for finding the cover set. The algorithm uses an assumption that the number of intersections between regions spanned by the base stations is bounded by a constant $l$ (e.g. each region intersects at most $l$ other regions). The algorithm works as follows. At each step randomly pick a customer $p_i \in P$ and add **all** the servers that cover it to the set $S$ of picked base stations. Update $P$ by deleting from it all the customers currently covered by $S$ and repeat until all customers are covered. We discard redundant base stations as in section 2.2.

To estimate the performance of this algorithm let us define $OPT$ be the optimal set cover for the problem. At each step $i$ the algorithm adds $n_i$ base stations to the solution, where $n_i \leq l$ (the assumption). Since $p_i$ must be covered, one of this $n_i$ base stations must appear in $OPT$. The algorithm stops when all the base stations from $OPT$ have been added to $S$. So, for each base station from $OPT$ the algorithm might pick at most $l-1$ other base stations, hence $|S| \leq l|OPT|$ and the approximation ratio of the algorithm is $O(l)$.

### 2.4 Adding the greedy approach into Randomized algorithm

The algorithm we discuss in this section is a variation of the random algorithm in section 2.3. Instead of randomly picking the next customer from $P$ we pick a customer covered by the least number of servers. For each customer $p_i$ we assign a number $n_i$ - the number of facilities that can serve customer $p_i$. At each step the algorithm picks a customer $p_i$ with minimal $n_i$ and adds to the solution all the base stations that cover it. We discard redundant servers as in section 2.2.

## 3 Connecting the base stations

We now want to connect the picked base stations. In a connected system each base station can transmit to any other (either directly or through auxiliary base stations). If it is impossible to connect two servers $a$ and $b$ due to the transmission radius or propagation constrains, we have to add relay stations to connect $a$ and $b$. We build the connected system in three stages:
**(1)** We first find the base stations that can be connected by *only* adding relay stations to them.
**(2)** Next we divide the servers into clusters: $a$ and $b$ are in one cluster if there exist servers $a = x_1, x_2, ..., x_k = b$, $k \geq 2$, such that for $1 \leq i \leq k-1$ $x_i$ and $x_{i+1}$ are in the range of each other.
**(3)** In the final step we connect clusters by building new servers and adding relay stations.

Stages 1 and 2 can be done simultaneously by an MST algorithm. In stage 3 we use a greedy strategy: at each step connect the nearest pair of clusters by adding a small number of new servers. Next, replace these clusters by their union and the additional new base stations. Repeat until only one cluster is left.

To connect a pair of clusters we use two different approaches: Dynamic Programming and Integer Programming. We describe them below.

### 3.1 Dynamic Programming

The input to the Dynamic Programming algorithm consists of two clusters of base stations $A$ and $B$ and the set of the possible base station locations, $D$.

We build a full graph $G = (V, E)$, where $V = A \cup B \cup D$. We define a weight function $w : E \to R$ that maps edges to real-valued weight $w(u, v)$, where $w(u, v)$ = cost of connecting $u$ and $v$. If $u$ and $v$ belong to the same cluster, then $w(u, v) = 0$ , if it is impossible to connect $u$ and $v$, then $w(u, v) = \infty$, if one or both of the nodes are in $D$ then $w(u, v)$ = cost of building new server(s) + cost of connecting $u$ and $v$ by relay stations. The problem of connecting two clusters of base

stations is now posed as finding the least-weight path from $u$ to $v$ in $G$, where $u \in A$, $v \in B$ and the weight of a path is the sum of the weights of its edges. We use Dynamic Programming to find all the shortest paths between pairs of vertices in G and then find the shortest among the paths from a node in $A$ and a node in $B$.

### 3.2 Integer Programming

We define two types of binary variables $x_i$ and $y_{i,j}$: $x_i$ represents the decision whether to create a new base station $i$, and $y_{i,j}$ whether to add a relay station on a server $i$ that transmits towards server $j$. Each variable may get value 0 or 1, where 1 means that we choose to build the corresponding base station or to add the corresponding relay station, and 0 means we don't. In order to build a path between clusters $A$ and $B$ we keep the following properties:

**(1)** For cluster A (B) we choose exactly one relay station (which will be placed on a new base station) that has the ability to connect directly to the cluster A (B). We define $T_A$ (resp. $T_B$) the set of relay stations that can connect directly to one of the antennas in cluster $A$ ($B$). Then the constrains we have are: $\sum_{y_{ij} \in T_A} y_{ij} = 1, \sum_{y_{ij} \in T_B} y_{ij} = 1$.

**(2)** If we choose to add relay station $y_{i,j}$ on base station $i$, we have to add relay station $y_{j,i}$ on base station $j$, i.e. if $y_{ij} = 1$ then $y_{ji} = 1$. Hence, for each $i$ and $j$, $y_{ij} - y_{ji} = 0$.

**(3)** On each new base station should be exactly two relay stations in order to continue the connectivity. Let $BT_i$ be a set of relay stations that can be placed on base station $x_i$ (this set is either empty or consists of exactly two relay stations), then for each $x_i \notin A \cup B \quad 2x_i - \sum_{y_{ij} \in BT_i} y_{ij} = 0$.

It is clear that if these properties hold, then we get a valid connection between the two clusters. The cost of the solution is the total cost of new servers and relay stations. Our goal is to find the cheapest solution, so the objective function is to minimize $\sum_{x_i \ \ y_{ij} \in BT_i} (x_i c_i + y_{ij} c_{ij})$, where $c_i$ is the cost of building a base station $x_i$ and $c_{ij}$ is the cost of adding a relay station $y_{ij}$. If $x_i \in A \cup B$ then $c_i = 0$.

## 4 Implementation and the experimental results

We have implemented the two algorithms based on Integer programming (sections 2.1 and 3.2) in C++, and the approximation algorithms for finding cover set (sections $2.2 - 2.4$) and the Dynamic Programming algorithm (section 3.1) in Java . The algorithms were tested on Linux operating system on Pentium-III machine with 512 MB memory. While implementing the algorithms, we tried to use efficient data structures, but our code was not fully optimized.

The algorithms for the first sub problem were tested and compared on the following inputs:
(1) the number of customers varied between 100 to 5000,
(2) the customers were randomly picked in clusters (standing for villages) within a large square with side size 15000 units,
(3) the number of possible base station locations varied between 50 to 3500,
(4) the transmission radius of the base station was R = 250 units,
(5) potential locations for base stations were chosen in the following way:
- at grid points $G$ with distance 2R between two points
- at grid points of $G$ shifted by a vector ($\sqrt{2}$R, $\sqrt{2}$R)
- additional 70% of random points.

Table 1 shows that the outputs of the approximation algorithms were very close to optimal. We ran these algorithms on about 50 examples and got encouraging results: all of the approximation algorithms found near-optimal solution.

| | | | Cover size | | | |
|---|---|---|---|---|---|---|
| | # customers | # bs | Randomized | Greedy | Alg.3 | IP |
| 1 | 5000 | 3060 | 215 | 215 | 213 | 210 |
| 2 | 5000 | 3056 | 233 | 235 | 235 | 229 |
| 3 | 5000 | 3162 | 239 | 241 | 237 | 231 |
| 4 | 3000 | 3267 | 206 | 208 | 204 | 201 |
| 5 | 3000 | 3162 | 207 | 211 | 205 | 203 |

**Table 1. Finding the set of facilities that serve all the customers**

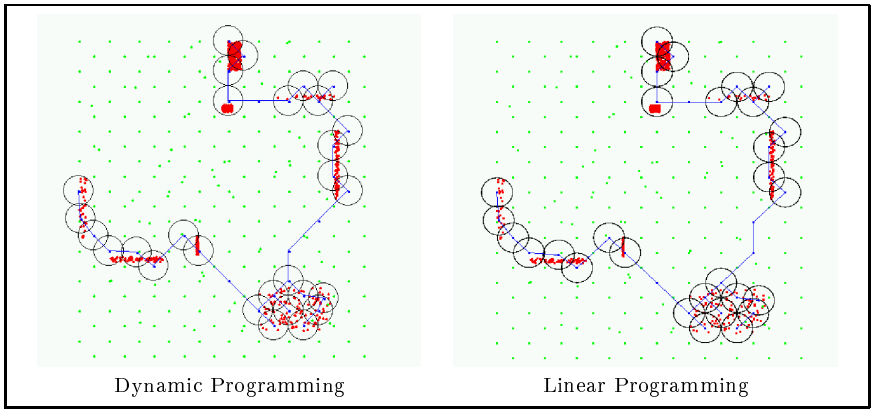


**Figure 1. Connecting the base stations**

The example illustrated in Fig.1 compares the performances of the Dynamic Programming and Integer Programming algorithms for the $2^{nd}$ sub problem. The input for the two programs consisted of 29 existing base stations and 235 possible locations for new base stations. The programs gave different connections with the same cost, but the run-time of the Dynamic Programming algorithm was 38308 msec, while the run-time of Integer Programming algorithm was 53581 msec. We used 5 kinds of relay stations with transmission radius 300, 350, 400, 450 and 500 units. In Fig.1 the red points represent the set of customers, green points represent the set of potential locations for base stations, black circles represent the regions spanned by the base stations that were picked in order to serve all the customers, blue points represent the base stations we use for connection and blue lines show the transmission. We run the programs on many example and in all of them, the costs of connecting the base stations produced by Integer Programming and Dynamic Programming algorithms were the same. Some of the results are summarized in Table 2.

| | # existing bs | # potential places for bs | cost of connecting | Time in msec DP | IP |
|---|---|---|---|---|---|
| 1 | 15 | 39 | 15084 | 2 | 15 |
| 2 | 9 | 74 | 30162 | 10 | 22 |
| 3 | 19 | 66 | 19110 | 9 | 38 |
| 4 | 17 | 51 | 17078 | 6 | 17 |
| 5 | 21 | 99 | 25730 | 3699 | 10287 |

**Table 2. Summary of experimental results**

In our experimentations we found that approximation algorithms are good in the sense they provide near-optimal solutions. For the problem of connecting the facilities into interconnected system we have described two heuristics based on Dynamic and Integer Programming accordingly. The solutions founded by these algorithms were different in connections, but had the same cost, which is not surprising since both optimize. In addition, the run-time of the Dynamic Programming algorithm was better in all our experiments.

# References

[1] S. Guha and S. Khuller. Approximation algorithms for connected dominating sets. In *European Symposium on Algorithms*, pages 179–193, 1996.

[2] S. Guha and S. Khuller. Connected facility location problems. In *IMACS Series in Discrete Mathematics and Theoretical Computer Science*, volume 40, pages 179–190, 1997.

[3] P. Slavík. A tight analysis of the greedy algorithm for set cover. In *ACM Symposium on Theory of Computing*, pages 435–441, 1996.

[4] C. Swamy and A. Kumar. Primal-dual algorithms for connected facility location problems. In *APPROX*, pages 256–270, 2002.

[5] U.Feige. A threshold of $\ln n$ for approximating set cover. In *The Twenty-Eighth Annual ACM Symposium on the Theory of Computing*, pages 314–318, 1996.