A Greedy Facility Location Algorithm Analyzed using Dual Fitting

Mohammad Mahdian* Evangelos Markakis Amin Saberi
Vijay Vazirani†

Abstract

We present a natural greedy algorithm for the metric uncapacitated facility location problem and use the method of dual fitting to analyze its approximation ratio, which turns out to be 1.861. The running time of our algorithm is $O(m \log m)$, where m is the total number of edges in the underlying complete bipartite graph between cities and facilities. We use our algorithm to improve recent results for some variants of the problem, such as the fault tolerant and outlier versions. In addition, we introduce a new variant which can be seen as a special case of the concave cost version of this problem.

Keywords: facility location, approximation algorithm, greedy, LP-Duality theory, linear programming.

*77 Massachusetts Avenue, Department of Mathematics, MIT, Cambridge, MA, 02139-4307, USA. mahdian@math.mit.edu

†801 Atlantic Drive, College of Computing, Georgia Tech, Atlanta, GA, 30332, USA. (vangelis, saberi, vazirani)@cc.gatech.edu

1 Introduction

A large fraction of the theory of approximation algorithms, as we know it today, is built around the theory of linear programming. Two fundamental algorithm design techniques, based on linear programming, yield approximation algorithms for a large number of important problems: LP-rounding and the primal–dual schema. One unsatisfying aspect of the current picture is that perhaps the most central problem of this theory, the set cover problem, is not solved using either of these techniques. Instead, it is solved using another method [18] *dual fitting*. The greedy algorithm for set cover can also be analyzed without using LP-duality [15]; however, the analysis using dual fitting is more powerful. In particular, it extends in a seamless manner to generalizations and variants, e.g., see [20].

The method can be described as follows, assuming a minimization problem: The basic algorithm is combinatorial – in the case of set cover it is in fact a simple greedy algorithm. Using the linear programming relaxation of the problem and its dual, one shows that the primal integral solution found by the algorithm is fully paid for by the dual computed; however, the dual is infeasible. The main step in the analysis consists of dividing the dual by a suitable factor and showing that the shrunk dual is feasible, i.e., it fits into the given instance. The shrunk dual is then a lower bound on OPT, and the factor is the approximation guarantee of the algorithm.

This method seems quite basic. However, to our knowledge, it does not seem to have found use outside of the set cover problem and its generalizations and variants. Perhaps the most important contribution of this paper is to apply this method to the fundamental metric uncapacitated facility location problem.

Our combinatorial algorithm for the metric uncapacitated facility location problem is a simple greedy algorithm. It is a small modification of Hochbaum's greedy algorithm for this problem. The latter was in fact the first approximation for this problem, with an approximation guarantee of $O(\log n)$. In contrast, our greedy algorithm achieves an approximation ratio of 1.861 and has a running time of $O(m \log m)$, where m is the number of edges of the underlying complete bipartite graph between cities and facilities, i.e. $m = n_c \times n_f$, where n_c is the number of cities and n_f is the number of facilities. Although this approximation factor is not the best known for this problem, our algorithm is natural and simple, and achieves the best approximation ratio within the same running time. For a metric defined by a sparse graph, Thorup [22] has obtained an algorithm, achieving a better running time, namely $\tilde{O}(|E|)$, where $|E|$ is the number of edges in the sparse graph. The approximation factor of his algorithm is $3 + o(1)$.

The first constant factor approximation algorithm for this problem was given by Shmoys, Tardos and Aardal [21]. Later, the factor was improved by Chudak and Shmoys [6] to $1+2/e$. This was the best known algorithm until the recent work of Charikar and Guha [2], who slightly improved the factor to 1.728. The above mentioned algorithms are based on LP-rounding, and therefore have high running times. Jain and Vazirani [13] gave a primal–dual algorithm, achieving a factor of 3, and having the same running time as ours (we will refer to this as the JV algorithm). Their algorithm was adapted for solving several related problems such as the fault-tolerant and outlier versions, and the k-median problem [13, 14, 4]. Mettu and Plaxton [19] used a restatement of their algorithm for the on-line median problem.

Strategies based on local search and greedy improvement for facility location problem

have also been studied. The work of Korupolu et. al. [16] shows that a simple local search heuristic proposed by Kuehn and Hamburger [17] yields a constant factor approximation for the facility location problem. Guha and Khuller [9] showed that greedy improvement can be used as a post-processing step to improve the approximation guarantee of certain facility location algorithms. The best approximation ratio for facility location [2] was obtained by combining a local search heuristic with the best LP-based algorithm known. They also combined greedy improvement and cost scaling to improve the factor of the JV algorithm. They proposed two algorithms with approximation factors of $2.41 + \epsilon$ and 1.853 and running times of $\tilde{O}(n^2/\epsilon)$ and $\tilde{O}(n^3)$ respectively, where n is the total number of vertices of the underlying graph. Regarding hardness results, Guha and Khuller [9] showed that the best approximation factor that we can get for this problem is 1.463, assuming $NP \not\subseteq DTIME[n^{O(\log \log n)}]$.

Our greedy algorithm is quite similar to the greedy set cover algorithm: iteratively pick the most cost-effective choice at each step, where cost-effectiveness is measured as the ratio of the cost incurred and the number of new cities served. In order to use LP-duality to analyze this algorithm, we give an alternative description which can be seen as a modification of the JV algorithm. This algorithm constructs a primal and dual solution of equal cost. However, the dual is infeasible. We show that if each of the dual variables is divided by 1.861, we obtain a feasible dual. As a consequence, the approximation guarantee of our algorithm is 1.861.

We have run our algorithm on randomly generated instances to obtain experimental results. The cost of the integral solution found is compared against the cost of an optimal solution to the LP-relaxation, and not an optimal integral solution (finding which will of course be unpractical). The results are good: the error varies in a small range of 0.8% to 7.1%.

We also use our algorithm to improve some recent results for some variants of the problem. In the facility location problem with outliers we are not required to connect all cities to some open facilities. In the robust version of this variant we are asked to choose l cities and connect the rest of them to some open facilities. In facility location with penalties we can either connect a city to a facility, or pay a specified penalty. Both versions were motivated by commercial applications, and were proposed by Charikar et al. [4]. In this paper we will modify our algorithm to obtain a factor 2 approximation algorithm for these versions, improving the best known result of factor 3.

In the fault tolerant variant, each city has a specified number of facilities it should be connected to. This problem was proposed in [14] and the best factor known is 2.47 [10]. We can show that we can achieve a factor 1.861 algorithm, when all cities have the same connectivity requirement. In addition, we introduce a new variant which can be seen as a special case of the concave cost version of this problem: the cost of opening a facility at a location is specified and it can serve exactly one city. In addition, a *setup cost* is charged the very first time a facility is opened at a given location.

2 The algorithm

Before stating the algorithm, we give a formal definition of the problem.
Metric uncapacitated facility location : Let G be a bipartite graph with bipartition

(F, C), where F is the set of facilities and C is the set of cities. Suppose also that $\mid C \mid = n_c$ and $\mid F \mid = n_f$. Thus, the total number of vertices in the graph $n = n_c + n_f$ and the total number of edges $m = n_c \times n_f$. Let f_i be the cost of opening facility i, and c_{ij} be the cost of connecting city j to facility i. The connection costs satisfy the triangle inequality. We want to find a subset $I \subseteq F$ of facilities that should be opened and a function $\phi : C \to I$ assigning cities to open facilities, such that the total cost of opening facilities and connecting cities to them is minimized.

In our algorithm we use a notion of cost-effectiveness. For each pair of a facility i with opening cost f_i, and a set of cities $C' \subseteq C$, we define its cost-effectiveness to be :

$$\frac{f_i + \sum_{j \in C'} c_{ij}}{\mid C' \mid}$$

In each iteration the algorithm picks the most cost-effective pair (i, C') greedily.

The algorithm is as follows:

Algorithm 1

1. In the beginning all cities are unconnected and all facilities are closed.
2. While $C \neq \emptyset$:
   - Among all pairs of facilities and subsets of C, find the most cost effective one, (i, C'), open facility i, if it is not already open, and connect all cities in C' to i.
   - Set $f_i := 0$, $C := C \backslash C'$.

Note that a facility can be chosen again after being opened, but its opening cost is counted only once since we set f_i to zero after the first time the facility is picked by the algorithm. As far as cities are concerned, every city j is removed from C, when connected to an open facility, and is not taken into consideration again.

Although the number of pairs of facilities and subsets of cities is exponentially large, in each iteration the most cost-effective pair can be found in polynomial time. For each facility i, we can sort the cities according to their connection cost to i. It can be easily seen that the most cost-effective pair will consist of a facility and a set, containing the first k cities in the increasing order of their connection cost to that facility, for some k.

The idea of cost-effectiveness essentially stems from a similar notion in the greedy algorithm for the set cover problem.

In that algorithm, the cost effectiveness of a set S is defined to be the cost of S over the number of uncovered elements in S. In each iteration, the algorithm picks the most cost-effective set until all elements are covered. The most cost-effective set can be found either by using direct computation, or by using the dual program of the linear programming formulation for the problem. The dual program can also be used to prove the approximation factor of the algorithm.

Similarly, we will use the LP-formulation of facility location to analyze our algorithm. As we will see, the dual formulation of the problem helps us to understand the nature of the problem and the greedy algorithm.

Consider the following integer program for this problem. In this program y_i is an indicator variable denoting whether facility i is open, and x_{ij} is an indicator variable denoting whether city j is connected to facility i. The first constraint ensures that each city is connected to at least one facility and the second that this facility should be open.

$$\begin{aligned}
\text{minimize} \quad & \sum_{i \in F, j \in C} c_{ij} x_{ij} + \sum_{i \in F} f_i y_i \\
\text{subject to} \quad & \sum_{i \in F} x_{ij} \geq 1 \quad \forall j \in C \\
& y_i - x_{ij} \geq 0 \quad \forall i \in F, j \in C \\
& x_{ij} \in \{0, 1\} \quad \forall i \in F, j \in C \\
& y_i \in \{0, 1\} \quad \forall i \in F
\end{aligned} \tag{1}$$

The LP-relaxation of this program can be obtained if we allow x_{ij} and y_i to be non-negative real numbers. The dual program of the LP-relaxation will then be:

$$\begin{aligned}
\text{maximize} \quad & \sum_{j \in C} \alpha_j \\
\text{subject to} \quad & \alpha_j - \beta_{ij} \leq c_{ij} \quad \forall i \in F, j \in C \\
& \sum_{j \in C} \beta_{ij} \leq f_i \quad \forall i \in F \\
& \alpha_j \geq 0 \quad \forall j \in C \\
& \beta_{ij} \geq 0 \quad \forall i \in F, j \in C
\end{aligned} \tag{2}$$

There is an intuitive way of interpreting the dual variables. We can think of α_j as the contribution of city j. This contribution goes towards connecting the city to some facility i and towards opening i. Using the inequalities of the dual program, we will have:

$$\sum_{j \in C} \max(0, \alpha_j - c_{ij}) \leq f_i \tag{3}$$

When inequality 3 holds with equality, it means that the total contribution of the cities towards opening facility i is enough to cover its cost, f_i, and thus we can open it.

We can now see how the dual variables can help us find the most cost-effective pair in each iteration of the greedy algorithm: if we start raising the dual variables of all unconnected cities simultaneously, the most cost-effective pair (i, C') will be the first pair for which $\sum_{j \in C'} \max(0, \alpha_j - c_{ij}) = f_i$.

Hence we can restate Algorithm 1 based on the above observation. This is in complete analogy to the greedy algorithm and its restatement using LP-formulation for set-cover.

Algorithm 2

1. We introduce a notion of time, so that each event can be associated with the time at which it happened. The algorithm starts at time 0. Initially, each city is defined to be unconnected, all facilities are closed, and α_j is set to 0 for every j.

2. While $C \neq \emptyset$:

   - For every city $j \in C$, increase the parameter α_j simultaneously, until one of the following events occur (if two events occur at the same time, we process them in arbitrary order).
     (a) For some unconnected city j, and some open facility i, $\alpha_j = c_{ij}$. In this case, we connect city j to facility i and we remove j from C.
     (b) For some closed facility i, we have :

$$\sum_{j \in C} \max(0, \alpha_j - c_{ij}) = f_i.$$

     This means that the total contribution of the cities is sufficient to open facility i. In this case, we open this facility, and for every unconnected city j with $\alpha_j \geq c_{ij}$, we connect j to i, and we remove it from C.

In each iteration of algorithm 1 the process of opening a facility and connecting some cities to it can be thought of as an event.

Theorem 1 *The events executed by algorithms 1 and 2 are identical.*

Proof. By induction. ■

Algorithm 2 can also be seen as a modification of JV algorithm [13]. The only difference is that in JV algorithm cities, when connected to an open facility, are not excluded from C, hence they might contribute towards opening several facilities. Due to this fact they have a second cleanup phase, in which some of the already open facilities will be closed down.

3 Analysis of the Algorithm

In this section we will give an LP-based analysis of this algorithm. As stated before, the contribution of each city goes towards opening at most one facility and connecting the city to an open facility. Therefore the total cost of the solution produced by our algorithm will be equal to the sum of the contributions ($\sum_j \alpha_j$). But ($\boldsymbol{\alpha}$, $\boldsymbol{\beta}$), where $\beta_{ij} = max(\alpha_j - c_{ij}, 0)$, is no longer a dual feasible solution as it was in the JV algorithm. The reason is that $\sum_j max(\alpha_j - c_{ij}, 0)$ can be greater than f_i and hence one of the constraints of the dual program, ($\sum_j \beta_{ij} \leq f_i$) is violated. However, if we show that for some number $R > 1$, we can define $\boldsymbol{\beta}$, in such a way that $(\boldsymbol{\alpha}/R, \boldsymbol{\beta}/R)$ is a feasible dual solution, then by the Weak Duality theorem, $(\sum_j \alpha_j)/R$ is a lower bound for OPT (where OPT is the optimum solution to the problem). Hence we have proved that the approximation ratio of the algorithm is R.

Theorem 2 *Let $\alpha_j, j = 1, \ldots, n_c$ denote the contribution of city j when algorithm 2 terminates. If for every facility i, and every set of k cities we have:*

$$\sum_{j=1}^{k} \alpha_j \leq R \left(f_i + \sum_{j=1}^{k} c_{ij} \right)$$

for some $R > 1$, then the approximation ratio of the algorithm is at most R.

Proof. Let $\beta_{ij} = max(\alpha_j - Rc_{ij}, 0)$. We will show that $(\boldsymbol{\alpha}/R, \boldsymbol{\beta}/R)$ is a feasible dual solution. To see that the first condition of the dual program is satisfied, we need to show that $\alpha_j - max(\alpha_j - Rc_{ij}, 0) \leq Rc_{ij}$. We can verify that this holds by considering the two possible cases $(\alpha_j > Rc_{ij})$ and $(\alpha_j \leq Rc_{ij})$. As far as the second constraint of the dual program is concerned, we need to show that $\sum_{j=1}^{n_c} max(\alpha_j - Rc_{ij}, 0) \leq Rf_i$. Let S be the set of cities for which $\alpha_j - Rc_{ij} > 0$. Then $\sum_{j=1}^{n_c} max(\alpha_j - Rc_{ij}, 0) = \sum_{j \in S}(\alpha_j - Rc_{ij})$. Thus the constraint becomes equivalent to the condition $\sum_{j \in S} \alpha_j \leq R\left(f_i + \sum_{j \in S} c_{ij}\right)$, which is true due to the assumptions of the theorem. Hence by the Weak Duality theorem it holds that $(\sum_j \alpha_j)/R \leq OPT$. We also know that the cost of the solution produced by our algorithm, SOL, is:

$$SOL = \sum_j \alpha_j \leq R \cdot OPT$$

This completes the proof. ■

From now on, we will assume without loss of generality that $\alpha_1 \leq \alpha_2 \leq \ldots \leq \alpha_{n_c}$. For the rest of the analysis, we will also need the following lemmata:

Lemma 3 *For every 2 cities j, j' and every facility i:*

$$\alpha_j \leq \alpha_{j'} + c_{ij'} + c_{ij} \tag{4}$$

Lemma 4 *For every city j and facility i, $\sum_{k=j}^{n_c} \max(\alpha_j - c_{ik}, 0) \leq f_i$*

The proofs of both lemmata are in the Appendix. Subject to the constraints introduced by these lemmata, we want to find a factor R such that for every facility i and every set of k cities:

$$\frac{\sum_{j=1}^{k} \alpha_j}{f_i + \sum_{j=1}^{k} c_{ij}} \leq R$$

This suggests considering the following program:

$$\begin{aligned}
z_k = \quad & \text{maximize} && \frac{\sum_{j=1}^{k} \alpha_j}{f + \sum_{j=1}^{k} d_j} && \\
& \text{subject to} && \alpha_j \leq \alpha_{j+1} && \forall j \in \{1, \ldots, k-1\} \\
& && \alpha_j \leq \alpha_l + d_j + d_l && \forall j, l \in \{1, \ldots, k\} \\
& && \sum_{l=j}^{k} \max(\alpha_j - d_l, 0) \leq f && \forall j \in \{1, \ldots, k\} \\
& && \alpha_j, d_j, f \geq 0 && \forall j \in \{1, \ldots, k\}
\end{aligned} \tag{5}$$

For a facility i and a set of k cities, S, the variables f and d_j's of this maximization program will correspond to the opening cost of i, f_i, and the costs of connecting each city $j \in S$ to i, c_{ij}.

It is easy to show that for every $k \geq 1$, $z_k > 1$ and $z_k \leq z_{k+1}$. We can also prove, by demonstrating an infinite family of instances, that the approximation ratio of Algorithm 2 is not better than $\sup_{k\geq 1}\{z_k\}$.

Theorem 5 *For every k, there is an instance of the facility location problem for which Algorithm 2 outputs a solution of cost at least z_k times the optimum solution.*

The following theorem combined with theorems 2 and 5 shows that the factor of our algorithm is exactly equal to $\sup_{k\geq 1}\{z_k\}$.

Theorem 6 *For every facility i and every set of k cities, $1 \leq k \leq n_c$, we have:*

$$\sum_{j=1}^{k} \alpha_j \leq z_k(f_i + \sum_{j=1}^{k} c_{ij})$$

The proofs of theorems 5 and 6 are in the Appendix. Hence, in order to prove that the approximation ratio of our algorithm is 1.861 it is enough to show that $sup_{k\geq 1}\{z_k\} \leq 1.861$.

It's not difficult to prove that z_k (the maximum value of the objective function of program 5) is equal to the optimal solution of the following linear program:

$$\begin{array}{lll}
z_k = \text{ maximize} & \sum_{j=1}^{k} \alpha_j & \\
\text{subject to} & f + \sum_{j=1}^{k} d_j \leq 1 & \\
& \alpha_j \leq \alpha_{j+1} & \forall j \in \{1, \ldots, k-1\} \\
& \alpha_j \leq \alpha_l + d_j + d_l & \forall j, l \in \{1, \ldots, k\} \\
& x_{jl} \geq \alpha_j - d_l & \forall j, l \in \{1, \ldots, k\} \\
& \sum_{l=j}^{k} x_{jl} \leq f & \forall j \in \{1, \ldots, k\} \\
& \alpha_j, d_j, f, x_{jl} \geq 0 & \forall j, l \in \{1, \ldots, k\}
\end{array}$$

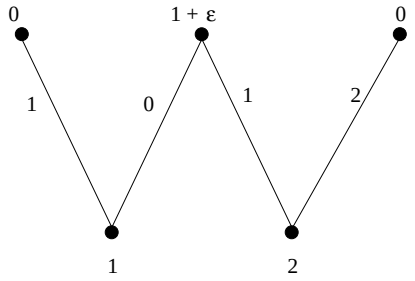
By finding an appropriate feasible solution to the dual of the above linear program, we managed to obtain the desirable upper bound for $sup_{k\geq 1}\{z_k\}$.

Theorem 7 *For every $k \geq 1$, $z_k \leq 1.861$.*

The proof involves many technical arguments and can be found in the Appendix.

We solved the above linear program for $k = 500$ using the software package AMPL, and we found that $z_{500} = 1.81$. This implies that the approximation factor of our algorithm is between 1.81 and 1.861. We still do not know if our algorithm achieves a better approximation guarantee than 1.861.

The following simple example shows that the approximation factor of the algorithm is at least 1.5.



The cost of the missing edges is equal to the cost of the shortest path in the above graph. Obviously the optimal solution is to open the facility with cost $1 + \epsilon$ and connect both cities to it. Therefore, $OPT = 2 + \epsilon$. But, if we run algorithm 2, at time 1 city 1 will be connected to the first facility from the left and will withdraw its contribution from the rest of them. Hence, city 2 will be connected at time 2 to the facility on the left and the total cost of the solution produced by the algorithm will be 3.

3.1 Running time analysis

In order to implement algorithm 2, for each facility, we keep track of the unpaid cost and the number of cities that contribute towards its cost. We also maintain a heap of events. The events are extracted and processed from the heap in increasing order of the time that they occur. There are three types of events:

(a) City j starts contributing towards opening facility i ($\alpha_j = c_{ij}$; i is not open). In this case we update the unpaid cost, the number of cities that contribute to i, and the expected opening time of facility i.

(b) Facility i is being paid for and hence opened. All cities contributing towards opening i will be connected to it, and the steps of the next event will be executed for them.

(c) City j connects to an open facility i ($\alpha_j = c_{ij}$; i is open). For all other facilities, which city j was contributing to, other than i, the number of contributors is decreased by one, and we recompute their expected opening time.

The total number of events is $O(m)$. Therefore, the cost of extracting an event from the heap is $O(\log m)$. We have at most m events of type (a), n_f events of type (b), and n_c events of type (c). The cost of processing the events are $O(\log m)$, $O(n_c)$ and $O(n_f \log m)$ respectively. Hence, the running time of the algorithm is $O(m \log m)$.

4 Experimental results

We implemented algorithm 2 in C to see how it behaves in practice. The test bed of our experiments consisted of randomly generated instances: In each instance, cities and facilities were points, drawn uniformly from a 10000×10000 grid, using a random number generator. We set the connection cost between a city and a facility to be equal to the euclidean distance of the corresponding points. Furthermore the opening cost of each facility was drawn uniformly between 0 and 9999 . As a lower bound for the optimal solution of each instance,OPT, we used the optimal solution of the LP-relaxation, which we computed using the package AMPL.

We varied the instance sizes from 50 cities and 20 facilities to 400 cities and 150 facilities. For each size we generated 20 instances and computed the ratio of the solution produced by our algorithm over the optimal solution of the LP-relaxation. Our results showed that the average ratio for each instance size varied between 1.025 and 1.034 whereas the worst ratio was 1.071. Hence the average error of our algorithm on these randomly generated instances varied between 2.5% and 3.4 %.

Some of the test data that we generated along with the corresponding results are shown in the Appendix.

5 Variants

5.1 Arbitrary demands

In this version, for each city j, a non-negative integer demand d_j, is specified. An open facility i can serve this demand at the cost of $c_{ij}d_j$. The best way to look at this modification is to reduce it to unit demand case by making d_j copies of city j. This reduction suggests that we need to change algorithm 2, so that each city j raises its contribution α_j at rate d_j. Note that the modified algorithm still works in more general cases, where d_j is fractional or exponentially large.

Theorem 8 *Our algorithm achieves an approximation ratio of 1.861 for the arbitrary demands facility location problem, and has a running time of* $O(m \log m)$.

5.2 Fault tolerant version

We are given a connectivity requirement r_j for each city j, which specifies the number of open facilities that city j should be connected to. We can see that this problem is closely related to the set multi-cover problem, in the case at which every set can be picked at most once [20]. In that problem we need to choose a collection of sets, so that every element e is covered a specified number of times, r_e. The greedy algorithm for set cover can be adapted for this variant achieving the same approximation factor. We can use the same approach to deal with the fault tolerant facility location:

The mechanism of raising dual variables and opening facilities is the same as in our initial algorithm. The only difference is that city j stops raising its dual variable and withdraws

its contribution from other facilities, when it is connected to r_j open facilities. Clearly, this algorithm is identical to algorithm 2 when $r_j = 1$ for every city j.

Theorem 9 *The algorithm described above has an approximation ratio of 1.861 for the fault tolerant facility location problem, when all r_j's are equal.*

5.3 Facility location with penalties

In this version we are not required to connect all cities to an open facility; however, for each city j, there is a specified penalty, p_j, which we have to pay, if it is not connected to any open facility. We can modify algorithm 2 for this problem, so that city j does not increase its dual variable more than p_j. If α_j reaches p_j before j is connected to any open facility, the city stops raising its dual variable and keeps its contribution equal to its penalty until it is either connected to an open facility or all remaining cities stop raising their dual variable. At this point, the algorithm terminates and unconnected cities remain unconnected. Using the same proof as the one we used for algorithm 2, we can show that the approximation ratio of this algorithm is 2.

Theorem 10 *Our algorithm achieves an approximation factor of 2 and has a running time of $O(m \log m)$.*

5.4 Robust facility location

We are given a number l and we are required to connect only $n_c - l$ cities to open facilities. This problem can be reduced to the previous one via Lagrangian relaxation.

Very recently, Charikar et al. [4] proposed a primal dual algorithm, based on JV algorithm, which achieves an approximation ratio of 3. As they showed, the linear programming formulation of this variant has an unbounded integrality gap. In order to fix this problem, they use the technique of parametric pruning, in which they guess the most expensive facility in the optimal solution. After that, they run JV algorithm on the pruned instance, where the only allowable facilities are those that are not more expensive than the guessed facility. Here we can use the same idea. Suppose that the most expensive facility is i with cost f_i. We modify the instance by setting its cost to zero. For all facilities whose cost is greater than f_i, we set the facility cost to ∞. Now we run algorithm 2, the only difference being that the algorithm terminates, when the number of unconnected cities is at most l. If the number of unconnected cities at the termination time is less than l, we can choose some cities among those with maximum α and disconnect them.

To analyze the algorithm, let us focus on the case, in which the algorithm guesses i correctly. The instance is modified by setting the cost of all facilities which are more expensive than f_i to ∞, and the cost of i to zero. Hence, the cost of the optimal solution to the modified instance, OPT', is equal to $OPT - f_i$, where OPT is the cost of the optimal solution to the original instance. With an argument similar to the proof presented for the original problem we can show that, when the algorithm terminates, the total contribution of the connected cities is at most $2OPT'$. In addition, the solution to the original instance might include the cost of opening the most expensive facility i plus the last facility that

we opened, which is also at most f_i. Hence the cost of our solution, SOL, is:

$$SOL \leq 2OPT^{'} + 2f_i \leq 2OPT$$

Theorem 11 *The approximation factor of the algorithm stated above for the robust facility location problem is 2.*

5.5 Dealing with capacities

In real applications, it's not usually the case that the cost of opening a facility is independent of the number of cities it will serve. But we can assume that we have *economy of scales* i.e. the cost of serving each city is decreasing when the number of cities is increasing. In order to capture that property, we define the following variant of the capacitated metric facility location problem. For each facility i, there is an initial opening cost f_i. After facility i is opened, it will cost s_i to serve each city. This variant can be solved using metric uncapacitated facility location problem. We'll prove that these two problems are reducible to each other. One direction is trivial (set all s_i's to be zero) for the other direction we just have to change the metric such that for each city j and facility i, $c^{'}_{ij} = c_{ij} + s_i$. Clearly, $c^{'}$ is also a metric and the solution of the metric uncapacitated version to this problem can be interpreted as a solution to the original problem with the same cost.

There is also another variant of the capacitated problem first appeared in [13]. Suppose that each open facility can serve at most u_i cities. We can open a facility an unlimited number of times and if we open it y_i times it can serve $y_i u_i$ cities. We reduce the first problem to this one by defining $s_i = f_i/u_i$. If in the solution to this problem k cities are connected to facility i, we open facility i $\lceil k/u_i \rceil$ times. The cost of the solution will be at most two times the original cost so we can give an algorithm with approximation factor equal to two times the best approximation factor known for uncapacitated version, which improves the best factor known to this problem which was four [13].

Note added April 2, 2001: A small modification of our greedy algorithm, analyzed using dual fitting, has been shown to achieve an approximation ratio of 1.61. This becomes the current best factor for the metric uncapacitated facility location problem [12]. The running time of their algorithm is the same as ours.

Acknowledgments

The first and third authors would like to thank Dr. Mohammad Ghodsi, Computer Engineering Department, Sharif University of Technology, for introducing them to the facility location problem. We would also like to thank Nisheet K. Vishnoi for valuable discussions.

References

[1] M. L.Balinski. On finding integer solutions to linear programs. *Proc. IBM Scientific Computing Symposium on Combinatorial Problems*, pp. 225-248, 1966.

[2] M. Charikar, S. Guha. Improved combinatorial algorithms for facility location and k-median problems. In *Proceedings of the 40th Annual IEEE Symposium on Foundations of Computer Science*, pp. 378-388, October 1999.

[3] M. Charikar, S. Guha, E. Tardos, and D. B. Shmoys. A constant-factor approximation algorithm for the k-median problem. In *Proceedings of the 31st Annual ACM Symposium on Theory of Computing*, pp. 1-10, May 1999.

[4] M. Charikar, S. Khuller, D. Mount, and G. Narasimhan. Algorithms for Facility Location Problems with Outliers. In *Proceedings of the 12th ACM-SIAM Symposium on Discrete Algorithms*, 2001.

[5] F. A. Chudak. Improved approximation algorithms for uncapacitated facility location. In R. E. Bixby, E. A. Boyd, and R. Z. Ros-Mercado, editors, *Integer Programming and Combinatorial Optimization*, volume 1412 of Lecture Notes in Computer Science, pp. 180-194, Springer, Berlin, 1998.

[6] F. Chudak and D. Shmoys. Improved approximation algorithms for the capacitated facility location problem. *Proc. 10th Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 875-876, 1999.

[7] V. Chvatal. A greedy heuristic for the set covering problem. *Math. Oper. Res.* 4 pp. 233-235, 1979.

[8] G. Cornuejols, G.L. Nemhauser, and L.A. Wosley. The uncapacitated facility location problem. In P. Mirchandani and R. Francis, editors, *Discrete Location Theory*, pp. 119-171, John Wiley and Sons, Inc., New York, 1990.

[9] S. Guha and S. Khuller. Greedy strikes back: Improved facility location algorithms. *Journal of Algorithms*, 31 pp. 228-248, 1999.

[10] S. Guha, A. Meyerson, and K. Munagala Improved Approximation Algorithms for Fault-tolerant Facility Location. In *Proceedings of the 12th ACM-SIAM Symposium on Discrete Algorithms*, 2001.

[11] D. S. Hochbaum. Heuristics for the fixed cost median problem.*Math. Programming*, 22:148-162, 1982.

[12] K. Jain and M. Mahdian and A. Saberi. Private communication, 2001.

[13] K. Jain and V. V. Vazirani. Primal-dual approximation algorithms for metric facility location and k-median problems. In *Proceedings of the*

40th Annual IEEE Symposium on Foundations of Computer Science, pp. 2-13, October 1999.

[14] K. Jain and V. Vazirani. An approximation algorithm for the fault tolerant metric facility location problem.*APPROX*, 2000.

[15] D. S. Johnson. Approximation algorithms for combinatorial problems.*J. Comput. System Sci.*, 9:256-278, 1974.

[16] M. R. Korupolu, C. G. Plaxton, and R. Rajaraman. Analysis of a local search heuristic for facility location problems. In *Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 1-10, January 1998.

[17] A. A. Kuehn and M. J. Hamburger. A heuristic program for locating warehouses. *Management Science*, 9 pp. 643-666, 1963.

[18] L. Lovasz. On the ratio of Optimal Integral and Fractional Covers. *Discrete Math.* 13 pp. 383-390, 1975.

[19] R. Mettu and G. Plaxton. The online median problem. *Proceedings of 41st IEEE FOCS*, 2000.

[20] S. Rajagopalan and V. V. Vazirani. Primal-dual RNC approximation of covering integer programs. *SIAM J. Comput.*, 28 pp. 526-541, 1999.

[21] D. B. Shmoys, E. Tardos, and K. Aardal. Approximation algorithms for facility location problems. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, pp. 265-274, May 1997.

[22] M. Thorup. Quick k-median, k-center, and facility location for sparse graphs. To appear in ICALP 2001.

Appendix

Proofs of theorems and lemmata

Proof of lemma 3. If $\alpha_{j'} \geq \alpha_j$, inequality 4 obviously holds. Assume that $\alpha_j > \alpha_{j'}$. Let $i^{'}$ be the facility that city $j^{'}$ is connected to by our algorithm. Thus, facility $i^{'}$ is open at time $\alpha_{j'}$. The contribution α_j cannot be greater than $c_{i'j}$ because in that case city j would be connected to facility $i^{'}$ at some time $t < \alpha_j$. Hence $\alpha_j \leq c_{i'j}$. Furthermore by triangle inequality we have:

$$c_{i'j} \leq c_{i'j'} + c_{ij'} + c_{ij} \leq \alpha_{j'} + c_{ij'} + c_{ij}$$

This completes the proof. ■

Proof of lemma 4. Assume, for the sake of contradiction, that for some j and some i the inequality does not hold. Consider the time $t = \alpha_j$. At this time, all cities $j, j+1, \ldots, n_c$ are unconnected. Therefore, the total contribution for facility i is at least $\sum_{k=j}^{n_c} \max(\alpha_j - c_{ik}, 0) > f_i$. This means that facility i is open before time t. But the total contribution of unconnected cities towards the opening cost of a facility is zero after the facility is opened. Hence, we have:

$$\sum_{k=j}^{n_c} \max(\alpha_j - c_{ik}, 0) \leq f_i \qquad \forall i \in F, j \in C \tag{6}$$

■

Proof of theorem 5. Consider an optimum feasible solution of program 5. We construct an instance of the facility location problem with k cities and $k+1$ facilities as follows: The cost of opening facility i is

$$f_i = \begin{cases} 0 & \text{if } 1 \leq i \leq k \\ f & \text{if } i = k+1 \end{cases}$$

The connection cost between a city j and a facility i is:

$$c_{ij} = \begin{cases} \alpha_j & \text{if } 1 \leq i = j \leq k \\ d_j & \text{if } 1 \leq j \leq k, i = k+1 \\ d_i + d_j + \alpha_i & \text{otherwise} \end{cases}$$

It is easy to see that the connection costs satisfy the triangle inequality. On this instance, our algorithm connects city 1 to facility 1, then it connects city 2 to facility 2, and finally connects city k to facility k. (The inequality $\sum_{l=j}^{k} \max(\alpha_j - d_l, 0) \leq f$ guarantees that city i can get connected to facility i before facility $k+1$). Therefore, the cost of Algorithm 2 is equal to $\sum_{j=1}^{k} c_{jj} + \sum_{i=1}^{k} f_i = \sum_{j=1}^{k} \alpha_j = z_k$.

On the other hand, the optimal solution for this instance is to connect all the cities to facility $k+1$. The cost of this solution is equal to $\sum_{j=1}^{k} c_{k+1,j} + f_{k+1} = f + \sum_{j=1}^{k} d_j \leq 1$.

Thus, our algorithm outputs a solution whose cost is at least z_k times the cost of the optimal solution. ■

Proof of theorem 6.

Let $d_j = c_{ij}$, $j = 1, \ldots, k$, and $f = f_i$. We prove that α_j, d_j, f form a feasible solution of program 5.

We have already assumed that $\alpha_1 \leq \alpha_2 \leq \ldots \alpha_{n_c}$. Furthermore by lemmata 3 and 4 it follows immediately that the rest of the constraints of program 5 are satisfied. Therefore, α_i, d_i, f constitute a feasible solution of program 5. Consequently

$$\frac{\sum_{j=1}^{k} \alpha_j}{f_i + \sum_{j=1}^{k} c_{ij}} \leq z_k.$$

This completes the proof. ■

Proof of theorem 7. Let $r = 1.8609$. Assume, without loss of generality that k is sufficiently large. Consider a feasible solution of the program 5. It is clear from the third inequality that for every j, j' we have

$$\sum_{i=j}^{j'} (\alpha_j - d_i) \leq f. \tag{7}$$

Now, we define l_j and θ_j as follows:

$$l_j = \begin{cases} p_2 k & \text{if } j \leq p_1 k \\ k & j > p_1 k \end{cases}$$

$$\theta_j = \begin{cases} \frac{r+1}{p_2 k} & \text{if } j \leq p_1 k \\ \frac{(r+1)(p_2 - p_1)}{p_2(1-p_1)k} & p_1 k < j \leq p_2 k \\ 0 & j > p_2 k \end{cases}$$

where $p_1 = 0.1991$ and $p_2 = 0.5696$. We consider Inequality 7 for every $j \leq p_2 k$ and $j' = l_j$, and multiply both sides of this inequality by θ_j. By adding up all these inequalities, we obtain

$$\sum_{j=1}^{p_1 k} \sum_{i=j}^{p_2 k} \theta_j (\alpha_j - d_i) + \sum_{j=p_1 k+1}^{p_2 k} \sum_{i=j}^{k} \theta_j (\alpha_j - d_i) \leq (\sum_{j=1}^{p_2 k} \theta_j) f. \tag{8}$$

The coefficient of f in the right-hand side of the above inequality is equal to $\sum_{j=1}^{p_2 k} \theta_j = \frac{r+1}{p_2 k} p_1 k + \frac{(r+1)(p_2 - p_1)}{p_2(1-p_1)k}(p_2 k - p_1 k) = (r+1)\left(\frac{p_1}{p_2} + \frac{(p_2-p_1)^2}{p_2(1-p_1)}\right) \approx 1.8609 < 1.861$. Also, the coefficients of α_j and d_j in the left-hand side of Inequality 8 are equal to

$$\text{coeff}[\alpha_j] = \begin{cases} (p_2 k - j + 1)\theta_j & j \leq p_1 k \\ (k - j + 1)\theta_j & j > p_1 k \end{cases} \tag{9}$$

$$\text{coeff}[d_j] = \begin{cases} \sum_{i=1}^{j} \theta_i & j \leq p_2 k \\ \sum_{i=p_1 k+1}^{j} \theta_i & j > p_2 k \end{cases} \tag{10}$$

Notice that the sum of coefficients of α_j's is equal to

$$\begin{aligned}
\sum_{j=1}^{k} \text{coeff}[\alpha_j] &= \sum_{j=1}^{p_1 k} \frac{r+1}{p_2 k}(p_2 k - j + 1) + \sum_{j=p_1 k+1}^{p_2 k} \frac{(r+1)(p_2-p_1)}{p_2(1-p_1)k}(k-j+1) \\
&= \frac{r+1}{p_2 k}(p_1 p_2 k^2 - \frac{p_1 k(p_1 k-1)}{2}) \\
&\quad + \frac{(r+1)(p_2-p_1)}{p_2(1-p_1)k}((p_2-p_1)k^2 - \frac{(p_1 k + p_2 k - 1)(p_2 k - p_1 k)}{2}) \\
&> (r+1)\left(p_1 - \frac{p_1^2}{2p_2} + \frac{(p_2-p_1)^2}{p_2(1-p_1)} - \frac{(p_2-p_1)^2(p_1+p_2)}{2p_2(1-p_1)}\right)k \\
&\approx 1.00004k \\
&> k
\end{aligned}$$

Now, we use the inequality $\alpha_i \geq \alpha_j - d_j - d_i$ on the expression on the left-hand side of Inequality 8 to reduce the coefficients of α_j's that are greater than 1, and increase the coefficient of α_j's that are less than 1. Since the sum of these coefficients is greater than k, using this inequality and the inequality $\alpha_j \geq 0$ we can obtain an expression E that is less than or equal to the left-hand side of Inequality 8, and in which all α_j's have coefficient 1. The coefficient of d_j in this expression will be equal to its coefficient in the left-hand side of Inequality 8, plus the absolute value of the change in the coefficient of the corresponding α_j. Therefore, by Equations 9 and 10 this coefficient is equal to:

$$\text{coeff}_E[d_j] = \begin{cases} \sum_{i=1}^{j} \theta_i + |(p_2 k - j + 1)\theta_j - 1| & j \leq p_1 k \\ \sum_{i=1}^{j} \theta_i + |(k - j + 1)\theta_j - 1| & p_1 k < j \leq p_2 k \\ \sum_{i=p_1 k+1}^{j} \theta_i + |(k - j + 1)\theta_j - 1| & j > p_2 k \end{cases}$$

If $j \leq p_1 k$, we have $(p_2 k - j + 1)\theta_j > (p_2 k - p_1 k)\frac{r+1}{p_2 k} = (r+1)(p_2 - p_1)/p_2 \approx 1.8609 > 1$ Therefore,

$$\begin{aligned}
\text{coeff}_E[d_j] &= \sum_{i=1}^{j} \theta_i + (p_2 k - j + 1)\theta_j - 1 \\
&= \frac{r+1}{p_2 k} j + \frac{r+1}{p_2 k}(p_2 k - j + 1) - 1 \\
&= \frac{r+1}{p_2 k}(p_2 k + 1) - 1 \\
&= r + O(\frac{1}{k}) \\
&< 1.861
\end{aligned}$$

Similarly, if $p_1 k < j \leq p_2 k$, we have $(k - j + 1)\theta_j > (k - p_2 k)\frac{(r+1)(p_2-p_1)}{p_2(1-p_1)k} = \frac{(r+1)(p_2-p_1)(1-p_2)}{p_2(1-p_1)} \approx 1.00003 > 1$. Therefore,

$$\text{coeff}_E[d_j] = \sum_{i=1}^{j} \theta_i + (k - j + 1)\theta_j - 1$$

$$
\begin{aligned}
&= \frac{r+1}{p_2 k} p_1 k + \frac{(r+1)(p_2-p_1)}{p_2(1-p_1)k}(j - p_1 k) + \frac{(r+1)(p_2-p_1)}{p_2(1-p_1)k}(k - j + 1) - 1 \\
&= \frac{r+1}{p_2} p_1 + \frac{(r+1)(p_2-p_1)}{p_2(1-p_1)k}(k - p_1 k + 1) - 1 \\
&= (r+1)\left(\frac{p_1}{p_2} + \frac{(p_2-p_1)}{p_2}\right) - 1 + O(\frac{1}{k}) \\
&= r + O(\frac{1}{k}) \\
&< 1.861
\end{aligned}
$$

Finally, if $j > p_2 k$, the coefficient of d_j is equal to

$$
\begin{aligned}
\text{coeff}_E[d_j] &= \sum_{i=p_1 k}^{j} \theta_i + |0 - 1| \\
&= \frac{(r+1)(p_2-p_1)}{p_2(1-p_1)k}(p_2 k - p_1 k) + 1 \\
&= \frac{(r+1)(p_2-p_1)^2}{p_2(1-p_1)} + 1 \\
&\approx 1.8609 \\
&< 1.861
\end{aligned}
$$

Therefore, in each case, the coefficient of d_j is less than or equal to 1.861. Thus, we have proved that

$$\sum_{j=1}^{k} \alpha_j - \sum_{j=1}^{k} 1.861 d_j < 1.861 f.$$

This clearly implies that $z_k < 1.861$. ■

Experiments

As stated in the main paper, we ran our algorithm on randomly generated instances of the problem. The number of cities, n_c, varied between 50 and 400 whereas the number of facilities, n_f, ranged between 20 and 150. For each instance size (n_c, n_f) we generated 20 instances and computed the average and the worst ratio of the solution of our algorithm over the optimal solution of the LP-relaxation. The results of our experiments are shown in the following table.

n_c	n_f	instances	average ratio	worst ratio
50	20	20	1.033	1.070
100	20	20	1.025	1.071
100	50	20	1.026	1.059
200	50	20	1.032	1.059
200	100	20	1.027	1.064
300	50	20	1.034	1.070
300	80	20	1.030	1.057
300	100	20	1.033	1.053
300	150	20	1.029	1.048
400	100	20	1.030	1.060
400	150	20	1.030	1.050

Table 1: experimental results