Greedy Facility Location Algorithms Analyzed Using Dual Fitting with Factor-Revealing LP

KAMAL JAIN

Microsoft Research, Redmond, Washington

MOHAMMAD MAHDIAN

Laboratory for Computer Science, MIT, Cambridge, Massachusetts

AND

EVANGELOS MARKAKIS, AMIN SABERI, AND VIJAY V. VAZIRANI

Georgia Institute of Technology, Atlanta, Georgia

Abstract. In this article, we will formalize the method of dual fitting and the idea of factor-revealing LP. This combination is used to design and analyze two greedy algorithms for the metric uncapacitated facility location problem. Their approximation factors are 1.861 and 1.61, with running times of $O(m \log m)$ and $O(n^3)$, respectively, where n is the total number of vertices and m is the number of edges in the underlying complete bipartite graph between cities and facilities. The algorithms are used to improve recent results for several variants of the problem.

Categories and Subject Descriptors: F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems—*compuations on discrete structures*; G.2.1 [**Discrete Mathematics**]: Combinatorics—*combinatorial algorithms*; G.2.2 [**Discrete Mathematics**]: Graph Theory—*network problems*

General Terms: Algorithms, Theory

Additional Key Words and Phrases: Approximation algorithms, dual-fitting method, facility location problem, primal-dual method

---

This article is based on these preliminary versions: MAHDIAN, M., MARKAKIS, E., SABERI, A., AND VAZIRANI, V. V. 2001. A greedy facility location algorithm analyzed using dual fitting. In *Proceedings of the 5th International Workshop on Randomization and Approximation Techniques in Computer Science*. Lecture Notes in Computer Science, vol. 2129. Springer-Verlag, New York, pp. 127–137; JAIN, K., MAHDIAN, M., AND SABERI, A. 2002. A new greedy approach for facility location problems. In *Proceedings of the ACM Symposium on Theory of Computing*. ACM, New York.

M. Mahdian was supported in part by National Science Foundation (NSF) grant CCR-9701304.

Authors' addresses: K. Jain, Microsoft Research, One Microsoft Way, Redmond, WA 98052, e-mail: kamalj@microsoft.com; M. Mahdian, Laboratory for Computer Science, MIT, Cambridge, MA 02139, e-mail: mahdian@theory.lcs.mit.edu; E. Markakis, A. Saberi, and V. V. Vazirani, College of Computing, Georgia Tech, Atlanta, GA 30332, e-mail: {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, which offers the two fundamental algorithm design techniques of rounding and the primal–dual schema (see Vazirani [2001]). Interestingly enough, the LP-duality based analysis [Lovasz 1975; Chvatal 1979] for perhaps the most central problem of this theory, the set cover problem, did not use either of these techniques. Moreover, the analysis used for set cover does not seem to have found use outside of this problem and its generalizations [Rajagopalan and Vazirani 1999], leading to a somewhat unsatisfactory state of affairs.

In this article, we formalize the technique used for analyzing set cover as the *method of dual fitting*, and we also introduce the idea of using a *factor-revealing LP*. Using this combination we analyze two greedy algorithms for the metric uncapacitated facility location problem. Their approximation factors are 1.861 and 1.61, with running times of $O(m \log m)$ and $O(n^3)$ respectively, where m and n denote the total number of edges and vertices in the underlying complete bipartite graph between cities and facilities. In other words, $m = n_c \times n_f$ and $n = n_c + n_f$, where n_c is the number of cities and n_f is the number of facilities.

1.1. Dual Fitting with Factor-Revealing LP.

The set cover problem offers a particularly simple setting for illustrating most of the dominant ideas in approximation algorithms (see Vazirani [2001]). Perhaps the reason that the method of dual fitting was not clear so far was that the set cover problem did not require its full power. However, in retrospect, its salient features are best illustrated again in the simple setting of the set cover problem—we do this in Section 9.

The method of dual fitting 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 first interprets the combinatorial algorithm as a primal-dual-type algorithm—an algorithm that is iteratively making primal and dual updates. Strictly speaking, this is not a primal-dual algorithm, since the dual solution computed is, in general, infeasible (see Section 9 for a discussion on this issue). However, one shows that the primal integral solution found by the algorithm is fully paid for by the dual computed. By *fully paid for* we mean that the objective function value of the primal solution is bounded by that of the dual. The main step in the analysis consists of dividing the dual by a suitable factor, say γ, and showing that the shrunk dual is feasible, that is, it *fits* into the given instance. The shrunk dual is then a lower bound on OPT, and γ is the approximation guarantee of the algorithm.

Clearly, we need to find the minimum γ that suffices. Equivalently, this amounts to finding the worst possible instance—one in which the dual solution needs to be shrunk the most in order to be rendered feasible. For each value of n_c, the number of cities, we define a factor-revealing LP that encodes the problem of finding the worst possible instance with n_c cities as a linear program. This gives a family of LP's, one for each value of n_c. The supremum of the optimal solutions to these LP's is then the best value for γ. In our case, we do not know how to compute this supremum directly. Instead, we obtain a feasible solution to the dual of each of these LP's. An upper bound on the objective function values of these duals can be

computed, and is an upper bound on the optimal γ. In our case, this upper bound is 1.861 for the first algorithm and 1.61 for the second one. In order to get a closely matching tight example, we numerically solve the factor-revealing LP for a large value of n_c.

The technique of factor-revealing LPs is similar to the idea of LP bounds in coding theory. LP bounds give the best known bounds on the minimum distance of a code with a given rate by bounding the solution of a linear program (see Mahdian et al. [1977]). In the context of approximation algorithms, [Goemans and Kleinberg 1998] use a similar method in the analysis of their algorithm for the minimum latency problem.

1.2. THE FACILITY LOCATION PROBLEM. In the (uncapacitated) facility location problem, we have a set $\mathcal{F}$ of n_f *facilities* and a set $\mathcal{C}$ of n_c *cities*. For every facility $i \in \mathcal{F}$, a nonnegative number f_i is given as the *opening cost* of facility i. Furthermore, for every facility $i \in \mathcal{F}$ and city $j \in \mathcal{C}$, we have a *connection cost* (a.k.a. service cost) c_{ij} between facility i and city j. The objective is to open a subset of the facilities in $\mathcal{F}$, and connect each city to an open facility so that the total cost is minimized. We will consider the *metric* version of this problem, that is, the connection costs satisfy the triangle inequality.

This problem has occupied a central place in operations research since the early 60's [Balinski 1996; Kaufman et al. 1977; Kuehn and Hamburger 1963; Stollsteimer 1961, 1963], and has been studied from the perspectives of worst case analysis, probabilistic analysis, polyhedral combinatorics and empirical heuristics (see Cornuejols et al. [1990] and Nemhauser and Wolsey [1990]). Although the first approximation algorithm for this problem, a greedy algorithm achieving a guarantee of $O(\log n)$ in the general (nonmetric) case due to [Hochbaum 1982], dates back to almost 20 years ago, renewed interest in recent years has resulted in much progress. Recently, the problem has found several new applications in network design problems such as placement of routers and caches [Guha et al. 2000a; Li et al. 1999], agglomeration of traffic or data [Andrews and Zhang 1998; Guha et al. 2000b], and web server replications in a content distribution network (CDN) [Jamin et al. 2000; Qiu et al. 2001].

The first constant factor approximation algorithm for this problem was given by Shmoys et al. [1997]. Later, the factor was improved by Chudak [1998] and Chudak and Shmoys [1998] to $1 + 2/e$. Both these algorithms were based on LP-rounding, and therefore had high running times.

Jain and Vazirani [1999] 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 [Jain and Vazirani 1999, 2000; Charikar et al. 2001]. Mettu and Plaxton [2000] simplified the JV algorithm and improved its running time by a logarithmic factor.

Strategies based on local search and greedy improvement for facility location problem have also been studied. The work of Korupolu et al. [1998] shows that a simple local search heuristic proposed by Kuehn and Hamburger [1963] yields a $(5 + \epsilon)$-approximation algorithm with a running time of $O(n^6 \log n/\epsilon)$, for any $\epsilon > 0$. Charikar and Guha [1999] improved the factor slightly to 1.728 by combining the JV algorithm, greedy augmentation, and the LP-based algorithm [Chudak and Shmoys 1998]. They also combined greedy improvement and cost scaling to improve the

factor of the JV algorithm to 1.853. For a metric defined by a sparse graph, Thorup [2001] has obtained a $(3 + o(1))$-approximation algorithm with running time $\tilde{O}(|E|)$. Regarding hardness results, Guha and Khuller [1999] showed that the best approximation factor possible for this problem is 1.463, assuming $NP \not\subseteq DTIME[n^{O(\log \log n)}]$.

Since the publication of the first draft of this article, two new algorithms have been proposed for the facility location problem. The first algorithm, due to Sviridenko [2002], uses the LP-rounding method to achieve an approximation factor of 1.58. The second algorithm, due to Mahdian et al. [2002], combines our second algorithm with the idea of cost scaling to achieve an approximation factor of 1.52, which is currently the best known factor for this problem.

1.3. OUR RESULTS. Our first 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 to 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—when a city gets connected to an open facility, it withdraws whatever it has contributed towards the opening cost of other facilities. This step of withdrawing contribution is important, since it ensures that the primal solution is fully paid for by the dual.

The second algorithm has a minor difference with the first one: A city might change the facility to which it is connected and connect to a closer facility. If so, it offers this difference toward opening the latter facility.

The approximation factor of the algorithms are 1.861 and 1.61, with running times of $O(m \log m)$ and $O(n^3)$ respectively where n is the total number of vertices and m is the number of edges in the underlying complete bipartite graph between cities and facilities.

We have experimented our algorithms on randomly generated instances as well as instances obtained from the Operations Research library [Beasley 2002] and GT-ITM Internet topology generator [Zegura et al. 1996]. The cost of the integral solution found is compared against the solution of the LP-relaxation of the problem, rather than OPT (computing which would be prohibitively time consuming). The results are encouraging: The average error of our algorithms is about 3% and 1% respectively, and is a significant improvement over the JV algorithm which has an error of even 100% in some cases.

The primal-dual algorithm of Jain and Vazirani [1999] is versatile in that it can be used to obtain algorithms for many variants of the facility location problem, such as k-median, a common generalization of k-median and facility location, capacitated facility location with soft capacities [Jain and Vazirani 1999], and prize collecting facility location and facility location with outliers [Charikar et al. 2001]. In Section 8, we apply our algorithms to several variants of the problem. First, we consider a common generalization of the facility location and k-median problems. In this problem, which we refer to as the *k-facility location problem*, an instance of the facility location problem and an integer k are given and the objective is to find the cheapest solution that opens at most k facilities. The k-median problem is a special case of this problem in which all opening costs are 0. The k-median problem is studied extensively [Arya et al. 2001; Charikar and Guha 1999; Charikar et al. 1999; Jain and Vazirani 1999] and the best known approximation algorithm

for this problem, due to Arya et al. [2001], achieves a factor of $3+\epsilon$. The k-facility location problem has also been studied in operations research [Cornuejols et al. 1990], and the best previously known approximation factor for this problem was 6 [Jain and Vazirani 1999].

Next, we show an application of our algorithm to the facility location game. We also use our algorithm to improve recent results for some other variants of the problem. In the facility location problem with outliers, we are not required to connect all cities to open facilities. We consider two versions of this variant: In the robust version, we are allowed to leave l cities unconnected. 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. [2001]. In this article, we will modify our algorithm to obtain a factor 2 approximation algorithm for these versions, improving the best known result of factor 3 [Charikar et al. 2001].

In the fault-tolerant variant, each city has a specified number of facilities it should be connected to. This problem was proposed in [Jain and Vazirani 2000] and the best factor known is 2.47 [Guha et al. 2001]. We can achieve a factor of 1.61 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. *Algorithm* 1

In the following algorithm, we use a notion of cost effectiveness. Let us say that a *star* consists of one facility and several cities. The cost of a star is the sum of the opening cost of the facility and the connection costs between the facility and all the cities in the star. More formally, the cost of the star (i, C'), where i is a facility and $C' \subseteq C$ is a subset of cities, is $f_i + \sum_{j \in C'} c_{ij}$. The cost effectiveness of the star (i, C') is the ratio of the cost of the star to the size of C', that is, $(f_i + \sum_{j \in C'} c_{ij})/|C'|$.

Algorithm 1

(1) Let U be the set of unconnected cities. In the beginning, all cities are unconnected i.e. $U := C$ and all facilities are unopened.

(2) While $U \neq \emptyset$:

—Among all stars, 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 $f_i := 0$, $U := U \setminus C'$.

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. Also, notice that although the number of stars 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 in increasing order of their connection cost to i. It can be easily seen that the most cost-effective star will consist of a facility and a set, containing the first k cities in this order, 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.

The facility location problem can be captured by an integer program due to Balinski [1996]. For the sake of convenience, we give another equivalent formulation for the problem. Let $\mathcal{S}$ be the set of all stars. The facility location problem can be thought of as picking a minimum cost set of stars such that each city is in at least one star. This problem can be captured by the following integer program. In this program, x_S is an indicator variable denoting whether star S is picked and c_S denotes the cost of star S.

$$\begin{aligned} \text{minimize} \quad & \sum_{S \in \mathcal{S}} c_S x_S \\ \text{subject to} \quad & \forall j \in \mathcal{C} : \sum_{S : j \in S} x_S \geq 1 \\ & \forall S \in \mathcal{S} : \; x_S \in \{0, 1\} \end{aligned} \tag{1}$$

The LP-relaxation of this program is:

$$\begin{aligned} \text{minimize} \quad & \sum_{S \in \mathcal{S}} c_S x_S \\ \text{subject to} \quad & \forall j \in \mathcal{C} : \sum_{S : j \in S} x_S \geq 1 \\ & \forall S \in \mathcal{S} : \; x_S \geq 0 \end{aligned} \tag{2}$$

The dual program is:

$$\begin{aligned} \text{maximize} \quad & \sum_{j \in \mathcal{C}} \alpha_j \\ \text{subject to} \quad & \forall S \in \mathcal{S} : \sum_{j \in S \cap \mathcal{C}} \alpha_j \leq c_S \\ & \forall j \in \mathcal{C} : \; \alpha_j \geq 0 \end{aligned} \tag{3}$$

There is an intuitive way of interpreting the dual variables. We can think of α_j as the contribution of city j, or its share toward the total expenses. Note that the first inequality of the dual can also be written as $\sum_{j \in C} \max(0, \alpha_j - c_{ij}) \leq f_i$ for every facility i. We can now see how the dual variables can help us find the most cost-effective star in each iteration of the greedy algorithm: if we start raising the dual variables of all unconnected cities simultaneously, the most cost-effective

star will be the first star (i, C') 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.

Restatement of Algorithm 1

(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 ($U := C$), all facilities are unopened, and α_j is set to 0 for every j.
(2) While $U \neq \emptyset$, increase the time, and simultaneously, for every city $j \in U$, increase the parameter α_j at the same rate, until one of the following events occurs (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, connect city j to facility i and remove j from U.
   (b) For some unopened facility i, we have $\sum_{j \in U} \max(0, \alpha_j - c_{ij}) = f_i$. This means that the total contribution of the cities is sufficient to open facility i. In this case, open this facility, and for every unconnected city j with $\alpha_j \geq c_{ij}$, connect j to i, and remove it from U.

In each iteration of algorithm 1 the process of opening a facility and/or connecting some cities will be defined as an *event*. It is easy to prove the following lemma by induction.

LEMMA 2.1. *The sequence of events executed by Algorithm* 1 *and its restatement are identical.*

PROOF. By induction. □

This restatement can also be seen as a modification of JV algorithm [Jain and Vazirani 1999]. The only difference is that in JV algorithm cities, when connected to an open facility, are not excluded from U; 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.

Also, it is worth noting that despite the similarity between Algorithm 1 and Hochbaum's greedy algorithm for facility location (which is equivalent to the set cover algorithm applied on the set of stars), they are not equivalent. This is because we set f_i to zero after picking a set containing f_i. As the following example shows, the approximation factor of Hochbaum's algorithm is $\Omega(\frac{\log n}{\log \log n})$ on instances with metric inequality: Consider k facilities with opening cost p^k located in the same place. Also $k-1$ groups of cities $S_1, S_2, \ldots, S_{k-1}$. The group S_i consists of p^{k-i+1} cities with distance $\sum_{j=1\cdots i} p^{j-1}$ from the facilities. Other distances are obtained from the triangle inequality. Hochbaum's algorithm opens all facilities and therefore its solution costs more than kp^k. The optimum solution is $p^k + \sum_{i=1\cdots k-1} \sum_{j=1\cdots i} p^{j-1}$. It is easy to show that with a careful choice of k, the ratio of these two expressions is $\Omega(\frac{\log n}{\log \log n})$. We do not know whether the approximation factor of Hochbaum's algorithm on metric instances is strictly less than $\log n$.

3. *Analysis of Algorithm* 1

In this section, we will give an LP-based analysis of the 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 $\sum_j \alpha_j$ of the contributions. However, $\boldsymbol{\alpha}$ is not a feasible dual solution as it was in JV algorithm. The reason is that in every iteration of the restatement of Algorithm 1, we exclude a subset of cities and withdraw their contribution from all facilities. So at the end, for some facility i, $\sum_j \max(\alpha_j - c_{ij}, 0)$ can be greater than f_i and hence the corresponding constraint of the dual program is violated.

However, if we find a γ for which $\boldsymbol{\alpha}/\gamma$ is feasible, $\sum_j \alpha_j/\gamma$ would be a lower bound to the optimum and therefore the approximation factor of the algorithm would be at most γ. This observation motivates the following definition.

Definition 3.1. Given α_j $(j = 1, \ldots, n_c)$, a facility i is called at most γ-overtight if and only if

$$\sum_j \max\left(\frac{\alpha_j}{\gamma} - c_{ij}, 0\right) \leq f_i.$$

Using the above definition, it is trivial that $\boldsymbol{\alpha}/\gamma$ is a feasible dual if and only if each facility is at most γ-overtight. Now, we want to find such a γ. Note that in the above sum we only need to consider the cities j for which $\alpha_j \geq \gamma c_{ij}$. Let us assume without loss of generality that this is the case only for the first k cities. Moreover, assume without loss of generality that $\alpha_1 \leq \alpha_2 \leq \cdots \leq \alpha_k$. The next two lemmas express the constraints on $\boldsymbol{\alpha}$ imposed by the problem or our algorithm. The first lemma mainly captures the metric property and the second one expresses the fact that the total contribution offered to a facility at any time during the algorithm is no more than its cost.

LEMMA 3.2. *For every two cities j, j' and facility i, $\alpha_j \leq \alpha_{j'} + c_{ij'} + c_{ij}$.*

PROOF. If $\alpha_{j'} \geq \alpha_j$, the inequality obviously holds. Assume $\alpha_j > \alpha_{j'}$. 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 could be connected to facility i' at some time $t < \alpha_j$. Hence, $\alpha_j \leq c_{i'j}$. Furthermore, by the triangle inequality, $c_{i'j} \leq c_{i'j'} + c_{ij'} + c_{ij} \leq \alpha_{j'} + c_{ij'} + c_{ij}$. □

LEMMA 3.3. *For every city j and facility i, $\sum_{l=j}^{k} \max(\alpha_j - c_{il}, 0) \leq f_i$.*

PROOF. Assume, for the sake of contradiction, that for some j and some i the inequality does not hold, that is, $\sum_{l=j}^{k} \max(\alpha_j - c_{il}, 0) > f_i$. By the ordering on cities, for $l \geq j$, $\alpha_l \geq \alpha_j$. Let time $t = \alpha_j$. By the assumption, facility i is fully paid for before time t. For any city l, $j \leq l \leq k$ for which $\alpha_j - c_{il} > 0$ the edge (i, l) is tight before time t. Moreover, there must be at least one such city. For this city, $\alpha_l < \alpha_j$, since the algorithm will stop growing α_l as soon as l has a tight edge to a fully paid for facility. The contradiction establishes the lemma. □

Subject to the constraints introduced by Lemmas 3.2 and 3.3, we want to find the minimum γ for which $\sum_{j=1}^{k}(\alpha_j/\gamma - c_{ij}) \leq f_i$. In other words, we want to find the maximum of the ratio $\sum_{j=1}^{k} \alpha_j/(f + \sum_{j=1}^{k} d_j)$. We can define variables f, d_j, and α_j, corresponding to facility cost, distances, and contributions,

respectively, and write the following maximization program:

$$
\begin{array}{lll}
z_k = \text{maximize} & \dfrac{\sum_{j=1}^{k} \alpha_j}{f + \sum_{j=1}^{k} d_j} & \\
\text{subject to} & \alpha_j \le \alpha_{j+1} & \forall j \in \{1, \ldots, k-1\} \\
& \alpha_j \le \alpha_l + d_j + d_l & \forall j, l \in \{1, \ldots, k\} \\
& \sum_{l=j}^{k} \max(\alpha_j - d_l, 0) \le f & \forall j \in \{1, \ldots, k\} \\
& \alpha_j, d_j, f \ge 0 & \forall j \in \{1, \ldots, k\}
\end{array}
\tag{4}
$$

It is not difficult to prove that z_k (the maximum value of the objective function of program 4) is equal to the optimal solution of the following linear program, which we call the *factor-revealing LP*.

$$
\begin{array}{lll}
z_k = \text{maximize} & \displaystyle\sum_{j=1}^{k} \alpha_j & \\
\text{subject to} & f + \sum_{j=1}^{k} d_j \le 1 & \\
& \alpha_j \le \alpha_{j+1} & \forall j \in \{1, \ldots, k-1\} \\
& \alpha_j \le \alpha_l + d_j + d_l & \forall j, l \in \{1, \ldots, k\} \\
& x_{jl} \ge \alpha_j - d_l & \forall j, l \in \{1, \ldots, k\} \\
& \sum_{l=j}^{k} x_{jl} \le f & \forall j \in \{1, \ldots, k\} \\
& \alpha_j, d_j, f \ge 0 & \forall j \in \{1, \ldots, k\}
\end{array}
\tag{5}
$$

LEMMA 3.4. *Let* $\gamma = \sup_{k \ge 1}\{z_k\}$. *Every facility is at most* γ*-overtight*

PROOF. Consider facility i. We want to show that $\sum_j max(\alpha_j/\gamma - c_{ij}, 0) \le f_i$. Suppose without loss of generality that the subset of cities j such that $\alpha_j \ge \gamma c_{ij}$ is $\{j = 1, 2, \ldots, k\}$ for some k. Moreover $\alpha_1 \le \alpha_2 \le \cdots \le \alpha_k$. Let $d_j = c_{ij}$, $j = 1, \ldots, k$, and $f = f_i$. By Lemmas 3.2 and 3.3, it follows immediately that the constraints of program 4 are satisfied. Therefore, α_i, d_i, f constitute a feasible solution of program 4. Consequently,

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

By what we said so far, we know that the approximation factor of our algorithm is at most $\sup_{k \ge 1}\{z_k\}$. In the following theorem, we prove, by demonstrating an infinite family of instances, that the approximation ratio of Algorithm 1 is not better than $\sup_{k \ge 1}\{z_k\}$.

THEOREM 3.5. *The approximation factor of our algorithm is precisely* $\sup_{k \ge 1}\{z_k\}$.

PROOF. Consider an optimum feasible solution of program 4. 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

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

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

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

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 $\sum_{l=j}^{k} \max(\alpha_j - d_l, 0) \le f$ guarantees that city i can get connected to facility i before facility $k+1$). Therefore, the cost of the restatement of Algorithm 1 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 $\sum_{j=1}^{k} c_{k+1,j} + f_{k+1} = f + \sum_{j=1}^{k} d_j \le 1$.

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

The only thing that remains is to find an upper bound on $sup_{k \ge 1}\{z_k\}$. By solving the factor-revealing LP for any particular value of k, we get a lower bound on the value of γ. In order to prove an upper bound on γ, we need to present a general solution to the dual of the factor-revealing LP. Unfortunately, this is not an easy task in general. (For example, performing a tight asymptotic analysis of the LP bound is still an open question in coding theory). However, here empirical results can help us: we can solve the dual of the factor-revealing LP for small values of k to get an idea of how the general optimal solution looks. Using this, it is usually possible (although sometimes tedious) to prove a close-to-optimal upper bound on the value of z_k. We have used this technique to prove an upper bound of 1.861 on γ.

LEMMA 3.6. *For every $k \ge 1$, $z_k \le 1.861$.*

PROOF. By doubling a feasible solution of (4) it is easy to show that $z_k \le z_{2k}$. So we can assume, without loss of generality, that k is sufficiently large. Our objective is to multiply the inequalities of the program (4) by appropriate coefficients and add them up, to get an inequality of the form

$$\sum_{j=1}^{k} \alpha_j - 1.861 \sum_{j=1}^{k} d_j \le 1.861\, f. \tag{6}$$

Notice that finding the right multipliers for inequalities is equivalent to finding a feasible solution for the dual of the linear program.

We start by looking at the third inequality of program (4). This inequality implies that for every j and every $l_j \ge j$, we have

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

In fact, numerical computations for small values of k suggest that if for each j, we pick the right l_j, then replacing the third inequality of (4) by the above inequality does not change the solution of the linear program considerably. Furthermore, such

computations suggest that a step function of the following form is a reasonably good choice for l_j:

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

Here p_1 and p_2 are constants that we will fix later in the proof, to get the best possible bound. We now multiply Inequality 7 by a multiplier θ_i (that will be fixed later), and add up these inequalities:

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

We will pick θ_j's in such a way that

$$\sum_{j=1}^{k} \theta_j \leq 1.861. \tag{9}$$

This guarantees that the right-hand side of Inequality (8) is what we want in Inequality (6). Let us denote the lefthand side of Inequality (8) by A. The coefficients of α_j and $-d_j$ in A are equal to

$$\text{coeff}_A[\alpha_j] = \begin{cases} (p_2 k - j + 1)\theta_j & j \leq p_1 k \\ (k - j + 1)\theta_j & j > p_1 k \end{cases} \tag{10}$$

$$\text{coeff}_A[-d_j] = \begin{cases} \sum_{i=1}^{j} \theta_i & j \leq p_2 k \\ \sum_{i=p_1 k+1}^{j} \theta_i & j > p_2 k \end{cases} \tag{11}$$

We will pick θ_j's in such a way that the sum of coefficients of α_j's in A is at least k. This means that we need θ_j's to satisfy the following inequality.

$$\sum_{j=1}^{p_1 k} (p_2 k - j + 1)\theta_j + \sum_{j=p_1 k+1}^{k} (k - j + 1)\theta_j \geq k. \tag{12}$$

The above inequality guarantees that the average coefficient of α_j's on the left-hand-side of Inequality (8) is at least one, which is what we want to get in Inequality (6). However, some of these coefficients are less than one and others are greater than one, while at the end, we want all of them to be greater than one. In order to increase the coefficients that are less than one at the expense of decreasing the ones that are greater than one, we use the inequality $\alpha_i \geq \alpha_j - d_j - d_i$. Since the sum of the coefficients of α_j's in A is greater than k, using the above inequality, we can obtain an expression B that is less than or equal to A, such that all α_j's have coefficient at least one in B. However, every time we use the inequality $\alpha_i \geq \alpha_j - d_j - d_i$ to decrease the coefficient of α_i and increase the coefficient of α_j, the coefficients of $-d_i$ and $-d_j$ will also be increased by the same amount. Therefore, at the end, the coefficient of $-d_j$ in B will be at most $\text{coeff}_A[-d_j] + |\text{coeff}_A[\alpha_j] - 1|$. Thus, using

Eqs. (10) and (11), and making the assumption $p_1 < p_2$ (which will be satisfied by our choice of p_1 and p_2), we can write the coefficient of $-d_j$ in B as follows:

$$\text{coeff}_B[-d_j] = \begin{cases} \sum_{i=1}^{j} \theta_i + |(p_2k - j + 1)\theta_j - 1 \mid j \le p_1k \\ \sum_{i=1}^{j} \theta_i + |(k - j + 1)\theta_j - 1 \mid p_1k < j \le p_2k. \\ \sum_{i=p_1k+1}^{j} \theta_i + |(k - j + 1)\theta_j - 1 \mid j > p_2k \end{cases} \tag{13}$$

Therefore, if we can find $p_1 < p_2$ and θ_j's such that they satisfy conditions (9) and (12), and

$$\text{coeff}_B[-d_j] \le 1.861 \qquad \text{for all } j, \tag{14}$$

then we can get Inequality 6 by combining the inequalities in the linear program (4), and hence we obtain $z_k \le 1.861$.

In order to compute θ_j's for $j \le p_2k$, we set the coefficients given in Eq. (13) to 1.8609, and solve it as a recurrence (here we have 1.8609 instead of 1.861 to allow room for numerical errors in the computation of p_1 and p_2). This suggests the following values for θ_j's ($j \le p_2k$).

$$\theta_j = \begin{cases} \frac{2.8609}{p_2k} & \text{if } j \le p_1k \\ \frac{2.8609(p_2-p_1)}{p_2(1-p_1)k} & \text{if } p_1k < j \le p_2k \end{cases} \tag{15}$$

For $j > p_2k$, numerical computations suggest that we can take $\theta_j = 0$. Now that we have fixed the values of θ_j, we only need to verify the conditions (9), (12), and (14).

If p_1 and p_2 are such that $(p_2k-j+1)\theta_j \ge 1$ for $j \le p_1k$ and $(k-j+1)\theta_j \ge 1$ for $p_1k < j \le p_2k$, then it is easy to see that $\text{coeff}_B[-d_j] = 1.8609 + O(\frac{1}{k}) < 1.861$ for $j \le p_2k$. Therefore, in order to satisfy condition (14) for $j \le p_2k$, it is enough to pick p_1 and p_2 such that

$$(p_2k - p_1k + 1)\frac{2.8609}{p_2k} \approx \frac{2.8609(p_2 - p_1)}{p_2} \ge 1, \tag{16}$$

and

$$(k - p_2k + 1)\frac{2.8609(p_2 - p_1)}{p_2(1 - p_1)k} \approx \frac{2.8609(p_2 - p_1)(1 - p_2)}{p_2(1 - p_1)} \ge 1. \tag{17}$$

For $j > p_2k$, by Eq. (13) and our choice of θ_j's, we have $\text{coeff}_B[-d_j] = \frac{2.8609(p_2-p_1)}{p_2(1-p_1)k}(p_2k - p_1k) + 1$. Thus, condition (14) for $j > p_2k$ is equivalent to the following.

$$\frac{2.8609(p_2 - p_1)^2}{p_2(1 - p_1)} + 1 \le 1.861. \tag{18}$$

Also,

$$\sum_{j=1}^{k} \theta_j = \frac{2.8609}{p_2k} p_1k + \frac{2.8609(p_2 - p_1)}{p_2(1 - p_1)k}(p_2k - p_1k),$$

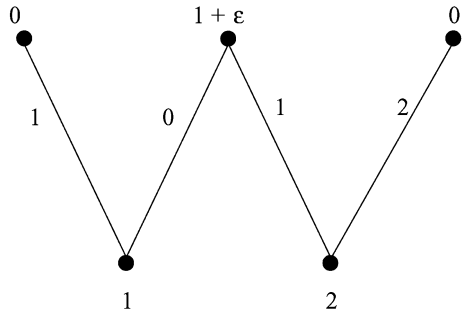

FIG. 1. The approximation ratio of Algorithm 1 is at least 1.5.

and therefore condition (9) is equivalent to the following:

$$\frac{2.8609p_1}{p_2} + \frac{2.8609(p_2 - p_1)^2}{p_2(1 - p_1)} \leq 1.861. \tag{19}$$

Similarly, straightforward calculations show that condition (12) is equivalent to the following.

$$2.8609\left(p_1 - \frac{p_1^2}{2p_2} + \frac{(p_2 - p_1)^2}{p_2(1 - p_1)} - \frac{(p_2 - p_1)^2(p_1 + p_2)}{2p_2(1 - p_1)}\right) \geq 1. \tag{20}$$

The only thing that remains is to find numeric values for p_1 and p_2 subject to the constraints (16)–(20). It is easy to verify that $(p_1, p_2) = (0.1991, 0.5696)$ satisfies all these constraints. This completes the proof of the lemma. □

Figure 1 shows a tight example for $k = 2$, for which the approximation factor of the algorithm is 1.5. The cost of the missing edges is given by triangle inequality. Numerical computations using the software CPLEX show that $z_{300} \approx 1.81$. Thus, the approximation factor of our algorithm is between 1.81 and 1.861. We do not know the exact approximation ratio.

4. *Algorithm* 2

Algorithm 2 is similar to the restatement of Algorithm 1. The only difference is that in Algorithm 1 cities stop offering money to facilities as soon as they get connected to a facility, but here they still offer some money to other facilities. The amount that an already-connected city offers to a facility j is equal to the amount that it would save in connection cost by switching its facility to j. As we will see in the next section, this change reduces the approximation factor of the algorithm from 1.861 to 1.61.

Algorithm 2

(1) We introduce a notion of time. The algorithm starts at time 0. At this time, each city is defined to be unconnected ($U := C$), all facilities are unopened, and α_j is set to 0 for every j.

At every moment, each city j offers some money from its contribution to each *unopened* facility i. The amount of this offer is computed as follows: If j is unconnected, the offer is equal to $\max(\alpha_j - c_{ij}, 0)$ (i.e., if the contribution of j is more than the cost that it has to pay to get connected to i, it offers to pay this extra amount to i); If j is already connected to some other

facility i', then its offer to facility i is equal to $\max(c_{i'j} - c_{ij}, 0)$ (i.e., the amount that j offers to pay to i is equal to the amount j would save by switching its facility from i' to i).

(2) While $U \neq \emptyset$, increase the time, and simultaneously, for every city $j \in U$, increase the parameter α_j at the same rate, until one of the following events occurs (if two events occur at the same time, we process them in an arbitrary order).

   (a) For some unopened facility i, the total offer that it receives from cities is equal to the cost of opening i. In this case, we open facility i, and for every city j (connected or unconnected) which has a nonzero offer to i, we connect j to i. The amount that j had offered to i is now called the *contribution* of j toward i, and j is no longer allowed to decrease this contribution.

   (b) For some unconnected city j, and some open facility i, $\alpha_j = c_{ij}$. In this case, connect city j to facility i and remove j from U.

Clearly, the main issue in the facility location problem is to decide which facilities to open. Once this is done, each city should be connected to the closest open facility. Observe that Algorithm 2 makes greedy choices in deciding which facilities to open and once it opens a facility, it does not alter this decision. In this sense, it is also a greedy algorithm.

5. *Analysis of Algorithm* 2

The following fact should be obvious from the description of Algorithm 2.

LEMMA 5.1. *The total cost of the solution found by Algorithm* 2 *is equal to the sum of the* α_j*'s.*

Now, as in the analysis of Algorithm 1, we need to find a number γ, such that for every star S, $\sum_{j \in S \cap \mathcal{C}} \alpha_j \leq \gamma c_S$. Such a γ will be an upper bound on the approximation ratio of the algorithm, since if for every facility i that is opened in the optimal solution and the collection A of cities that are connected to it, we write the inequality $\sum_{j \in A} \alpha_j \leq \gamma(f_i + \sum_{j \in A} c_{ij})$ and add up these inequalities, we find that the cost of our solution is at most γ times the cost of the optimal solution.

5.1. DERIVING THE FACTOR-REVEALING LP. Our proof follows the methodology of Section 3: express various constraints that are imposed by the problem or by the structure of the algorithm as inequalities and get a bound on the value of γ defined above by solving a series of linear programs.

Consider a star S consisting of a facility having opening cost f (with a slight misuse of the notation, we call this facility f), and k cities numbered 1 through k. Let d_j denote the connection cost between facility f and city j, and α_j denote the contribution of the city j at the end of Algorithm 2. We may assume without loss of generality that

$$\alpha_1 \leq \alpha_2 \leq \cdots \leq \alpha_k. \tag{21}$$

We need more variables to capture the execution of Algorithm 2. For every i ($1 \leq i \leq k$), consider the situation of the algorithm at time $t = \alpha_i - \epsilon$, where ϵ is very small, that is, just a moment before city i gets connected for the first time. At this time, each of the cities $1, 2, \ldots, i-1$ might be connected to a facility. For every $j < i$, if city j is connected to some facility at time t, let $r_{j,i}$ denote the connection cost between this facility and city j; otherwise, let $r_{j,i} := \alpha_j$. The latter case occurs if and only if $\alpha_i = \alpha_j$. It turns out that these variables (f, d_j's, α_j's,

and $r_{j,i}$'s) are enough to write down some inequalities to bound the ratio of the sum of α_j's to the cost of S (i.e., $f + \sum_{j=1}^{k} d_j$).

First, notice that once a city gets connected to a facility, its contribution remains constant and it cannot revoke its contribution to a facility, so it can never get connected to another facility with a higher connection cost. This implies that for every j,

$$r_{j,j+1} \geq r_{j,j+2} \geq \cdots \geq r_{j,k}. \tag{22}$$

Now, consider time $t = \alpha_i - \epsilon$. At this time, the amount city j offers to facility f is equal to

$$\begin{array}{ll} \max(r_{j,i} - d_j, 0) & \text{if } j < i, \text{ and} \\ \max(t - d_j, 0) & \text{if } j \geq i. \end{array}$$

Notice that by the definition of $r_{j,i}$ this holds even if $j < i$ and $\alpha_i = \alpha_j$. It is clear from Algorithm 2 that the total offer of cities to a facility can never become larger than the opening cost of the facility. Therefore, for all i,

$$\sum_{j=1}^{i-1} \max(r_{j,i} - d_j, 0) + \sum_{j=i}^{k} \max(\alpha_i - d_j, 0) \leq f. \tag{23}$$

We will also use the triangle inequality to derive the following constraint: for every $1 \leq j < i \leq k$,

$$\alpha_i \leq r_{j,i} + d_i + d_j. \tag{24}$$

In order to prove this, note that $\alpha_i = \alpha_j$ implies $r_{j,i} = \alpha_i$. That makes the inequality trivial. For the case that $\alpha_j < \alpha_i$, consider cities i and j at time $t = \alpha_i - \epsilon$. Let f' be the facility that j is connected to at time t. By the triangle inequality and the definition of $r_{j,i}$, the connection cost $c_{f'i}$ is at most $r_{j,i} + d_i + d_j$. Furthermore, $c_{f'i}$ cannot be less than t, since if it were, our algorithm would have connected city i to the facility f' at a time earlier than t, which is a contradiction.

The above inequalities form the following factor-revealing LP.

$$\begin{array}{ll} \text{maximize} & \dfrac{\sum_{i=1}^{k} \alpha_i}{f + \sum_{i=1}^{k} d_i} \\ \text{subject to} & \forall\, 1 \leq i < k : \ \alpha_i \leq \alpha_{i+1} \\ & \forall\, 1 \leq j < i < k : \ r_{j,i} \geq r_{j,i+1} \\ & \forall\, 1 \leq j < i \leq k : \ \alpha_i \leq r_{j,i} + d_i + d_j \\ & \forall\, 1 \leq i \leq k : \ \displaystyle\sum_{j=1}^{i-1} \max(r_{j,i} - d_j, 0) + \sum_{j=i}^{k} \max(\alpha_i - d_j, 0) \leq f \\ & \forall\, 1 \leq j \leq i \leq k : \ \alpha_j, d_j, f, r_{j,i} \geq 0 \end{array} \tag{25}$$

Notice that although the above optimization program is not written in the form of a linear program, it is easy to change it to a linear program by introducing new variables and inequalities.

TABLE I. SOLUTION OF THE FACTOR-REVEALING LP

k	$\max_{i \le k} z_i$
10	1.54147
20	1.57084
50	1.58839
100	1.59425
200	1.59721
300	1.59819
400	1.59868
500	1.59898

LEMMA 5.2. *If z_k denotes the solution of the factor-revealing LP, then for every star S consisting of a facility and k cities, the sum of α_j's of the cities in S in Algorithm* 2 *is at most $z_k c_S$.*

PROOF. Inequalities (21), (22), (23), and (24) derived above imply that the values $\alpha_j, d_j, f, r_{j,i}$ that we get by running Algorithm 2 constitute a feasible solution of the factor-revealing LP. Thus, the value of the objective function for this solution is at most z_k. □

Lemmas 5.1 and 5.2 imply the following:

LEMMA 5.3. *Let z_k be the solution of the factor-revealing LP, and $\gamma := \sup_k\{z_k\}$. Then Algorithm* 2 *solves the metric facility location problem with an approximation factor of γ.*

5.2. SOLVING THE FACTOR-REVEALING LP. As mentioned earlier, the optimization program (25) can be written as a linear program. This enables us to use an LP-solver to solve the factor-revealing LP for small values of k, in order to compute the numerical value of γ. Table I shows a summary of results that are obtained by solving the factor-revealing LP using CPLEX. It seems from the experimental results that z_k is an increasing sequence that converges to some number close to 1.6 and hence $\gamma \approx 1.6$.

We are using the same idea as Lemma 3.6 in Section 3 to prove the upper bound of 1.61 on z_k.

LEMMA 5.4. *Let z_k be the solution to the factor-revealing LP. Then, for every k, $z_k \le 1.61$.*

PROOF. Using the same argument as in Lemma 3.6, we can assume, without loss of generality, that k is sufficiently large. As in the proof of Lemma 3.6, our objective is to multiply the inequalities of the factor-revealing LP (25) by appropriate coefficients and add them up, to get an inequality of the form

$$\sum_{i=1}^{k} \alpha_i \le 1.61 \sum_{j=1}^{k} d_j + 1.61\, f. \tag{26}$$

Again, we notice that the fourth inequality of the program (25) implies that for every i and every $l_i \geq i$, we have

$$\sum_{j=i}^{l_i}(\alpha_i - d_j) \leq f - \sum_{j=1}^{i-1}\max(r_{j,i} - d_j, 0). \tag{27}$$

Let $x_{j,i} := \max(r_{j,i} - d_j, 0)$. The above inequality can be written as follows.

$$\alpha_i \leq \frac{1}{l_i - i + 1}\left(\sum_{j=i}^{l_i} d_j + f - \sum_{j=1}^{i-1} x_{j,i}\right). \tag{28}$$

As in the proof of Lemma 3.6, numerical computations suggest that a step function of the following form is a reasonably good choice for l_i. Here p_1 and p_2 are constants that will be fixed later in the proof, to get the best possible bound

$$l_i = \begin{cases} p_2 k & \text{if } i \leq p_1 k \\ k & \text{if } i > p_1 k. \end{cases}$$

Inequality (28) and the third inequality in the program (25) are two inequalities that bound α_i from above. In order to obtain the left hand side of Inequality (26), we add Inequality (28) for some values of i, and the third inequality of the factor-revealing LP for other values of i. Numerical computations suggest that one good choice is to add Inequality (28) for $i \leq p_2 k$ and the third inequality of the program (25) for $i > p_2 k$.

By the third inequality of the program (25) and the definition of $x_{j,i}$, we have

$$\alpha_i \leq x_{j,i} + 2d_j + d_i \tag{29}$$

for every $j \leq p_2 k$ and $i > p_2 k$. Notice that the variables $x_{j,i}$ appear with a positive coefficient in Inequality (29) and with a negative coefficient in the Inequality (28). Therefore, we can hope that after adding Inequality (28) for $i \leq p_2 k$ to Inequality (29) for $i > p_2 k$, $x_{j,i}$'s with positive coefficient get canceled out by the ones with negative coefficient. By the second inequality of the program (25), for every j, $x_{j,i}$ is a decreasing function of i. In particular, if we define $y_j := x_{j,p_2 k}$ for every $j \leq p_2 k$, then we have $x_{j,i} \geq y_j$ for every $j < i \leq p_2 k$ and $x_{j,i} \leq y_j$ for every $i > p_2 k$. Therefore, in both inequalities (28) and (29), we can estimate the variable $x_{j,i}$ by y_j. Thus,

$$\alpha_i \leq \sum_{j=i}^{l_i} \frac{d_j}{l_i - i + 1} + \frac{1}{l_i - i + 1} f - \sum_{j=1}^{i-1} \frac{y_j}{l_i - i + 1} \tag{30}$$

for every $i \leq p_2 k$, and

$$\alpha_i \leq d_i + 2d_j + y_j \tag{31}$$

for every $i > p_2 k$ and $j \leq p_2 k$.

We let $\ell \leq p_2 k$ be the index for which $2d_\ell + y_\ell$ is at its minimum (i.e., for every $j \leq p_2 k$, $2d_\ell + y_\ell \leq 2d_j + y_j$), and use Inequality (31) for $j = \ell$ in order to

get the most out of this inequality. By adding up Inequality (30) for $i \le p_2 k$ and Inequality (31) for $i > p_2 k$ and denoting $\zeta := \sum_{i=1}^{p_2 k} \frac{1}{l_i - i + 1}$, we obtain

$$\begin{aligned}
\sum_{i=1}^{k} \alpha_i &\le \sum_{i=1}^{p_2 k} \sum_{j=i}^{l_i} \frac{d_j}{l_i - i + 1} + \zeta f - \sum_{i=1}^{p_2 k} \sum_{j=1}^{i-1} \frac{y_j}{l_i - i + 1} \\
&\quad + \sum_{i=p_2 k+1}^{k} d_i + (2d_\ell + y_\ell)(1 - p_2)\, k = \sum_{j=1}^{p_2 k} \zeta d_j - \sum_{j=1}^{p_2 k} \sum_{i=j+1}^{p_2 k} \frac{d_j + y_j}{l_i - i + 1} \\
&\quad + \sum_{j=p_2 k+1}^{k} \left(1 + \sum_{i=p_1 k+1}^{p_2 k} \frac{1}{k - i + 1}\right) d_j + (2d_\ell + y_\ell)(1 - p_2)\, k + \zeta f \\
&\le \sum_{j=1}^{p_2 k} \zeta d_j + \sum_{j=p_2 k+1}^{k} \left(1 + \sum_{i=p_1 k+1}^{p_2 k} \frac{1}{k - i + 1}\right) d_j + \zeta f \\
&\quad + (2d_\ell + y_\ell) \left((1 - p_2)k - \frac{1}{2} \sum_{j=1}^{p_2 k} \sum_{i=j+1}^{p_2 k} \frac{1}{l_i - i + 1}\right),
\end{aligned}$$

where the last inequality is a consequence of the inequality $2d_\ell + y_\ell \le 2d_j + y_j \le 2d_j + 2y_j$ for $j \le p_2 k$. Now, let $\zeta' := 1 + \sum_{i=p_1 k+1}^{p_2 k} \frac{1}{k-i+1}$ and $\delta := (1 - p_2) - \frac{1}{2k} \sum_{j=1}^{p_2 k} \sum_{i=j+1}^{p_2 k} \frac{1}{l_i - i + 1}$. Therefore, the above inequality can be written as

$$\sum_{i=1}^{k} \alpha_i \le \sum_{j=1}^{p_2 k} \zeta d_j + \sum_{j=p_2 k+1}^{k} \zeta' d_j + \zeta f + \delta(2d_\ell + y_\ell)\, k, \tag{32}$$

where

$$\zeta = \sum_{i=1}^{p_2 k} \frac{1}{l_i - i + 1} = \ln \frac{p_2(1 - p_1)}{(p_2 - p_1)(1 - p_2)} + o(1), \tag{33}$$

$$\zeta' = 1 + \sum_{i=p_1 k+1}^{p_2 k} \frac{1}{k - i + 1} = 1 + \ln \frac{1 - p_1}{1 - p_2} + o(1), \text{ and} \tag{34}$$

$$\begin{aligned}
\delta &= 1 - p_2 - \frac{1}{2k} \sum_{j=1}^{p_2 k} \sum_{i=j+1}^{p_2 k} \frac{1}{l_i - i + 1} \\
&= \frac{1}{2} \left(2 - p_2 - p_2 \ln \frac{p_2}{p_2 - p_1} - \ln \frac{1 - p_1}{1 - p_2}\right) + o(1).
\end{aligned} \tag{35}$$

Now if we choose p_1 and p_2 such that $\delta < 0$, and let $\gamma := \max(\zeta, \zeta')$, then inequality (32) implies that

$$\sum_{i=1}^{k} \alpha_i \le (\gamma + o(1)) \left(f + \sum_{i=1}^{k} d_j\right).$$

Using Eqs. (33), (34), and (35), it is easy to see that subject to the condition $\delta < 0$, the value of γ is minimized when $p_1 \approx 0.439$ and $p_2 \approx 0.695$, which gives us $\gamma < 1.61$. □

Also, as in the proof of Theorem 3.5, we can use the optimal solution of the factor-revealing LP that is computed numerically (see Table I) to construct an example on which our algorithm performs at least z_k times worse than the optimum. These results imply the following.

THEOREM 5.5. *Algorithm* 2 *solves the facility location problem in time* $O(n^3)$, *where* $n = \max(n_f, n_c)$, *with an approximation ratio between* 1.598 *and* 1.61.

6. *The Tradeoff between Facility and Connection Costs*

We defined the cost of a solution in the facility location problem as the sum of the facility cost (i.e., total cost of opening facilities) and the connection cost. We proved in the previous section that Algorithm 2 achieves an overall performance guarantee of 1.61. However, sometimes it is useful to get different approximation guarantees for facility and connection costs. The following theorem gives such a guarantee. The proof is similar to the proof of Lemma 5.3.

THEOREM 6.1. *Let* $\gamma_f \geq 1$ *and* $\gamma_c := \sup_k\{z_k\}$, *where* z_k *is the solution of the following optimization program.*

$$
\begin{aligned}
\text{maximize} \quad & \frac{\sum_{i=1}^{k} \alpha_i - \gamma_f f}{\sum_{i=1}^{k} d_i} \\
\text{subject to} \quad & \forall\, 1 \leq i < k:\ \alpha_i \leq \alpha_{i+1} \\
& \forall\, 1 \leq j < i < k:\ r_{j,i} \geq r_{j,i+1} \\
& \forall\, 1 \leq j < i \leq k:\ \alpha_i \leq r_{j,i} + d_i + d_j \\
& \forall\, 1 \leq i \leq k:\ \sum_{j=1}^{i-1} \max(r_{j,i} - d_j, 0) + \sum_{j=i}^{k} \max(\alpha_i - d_j, 0) \leq f \\
& \forall\, 1 \leq j \leq i \leq k:\ \alpha_j, d_j,\ f, r_{j,i} \geq 0
\end{aligned}
\tag{36}
$$

Then for every instance $\mathcal{I}$ *of the facility location problem, and for every solution SOL for* $\mathcal{I}$ *with facility cost* F_{SOL} *and connection cost* C_{SOL}, *the cost of the solution found by Algorithm* 2 *is at most* $\gamma_f F_{SOL} + \gamma_c C_{SOL}$.

We have computed the solution of the optimization program (36) for $k = 100$, and several values of γ_f between 1 and 3, to get an estimate of the corresponding γ_c's. The result is shown in the diagram in Figure 2. Every point (γ_f, γ_c') on the thick line in this diagram represents a value of γ_f, and the corresponding estimate for the value of γ_c. The dashed line shows the following lower bound, which can be proved easily by adapting the proof of Guha and Khuller [1999] for hardness of the facility location problem.

THEOREM 6.2. *Let* γ_f *and* γ_c *be constants with* $\gamma_c < 1 + 2\exp(-\gamma_f)$. *Assume there is an algorithm* $\mathcal{A}$ *such that for every instance* $\mathcal{I}$ *of the metric facility location problem,* $\mathcal{A}$ *finds a solution whose cost is not more than* $\gamma_f F_{SOL} + \gamma_c C_{SOL}$ *for*

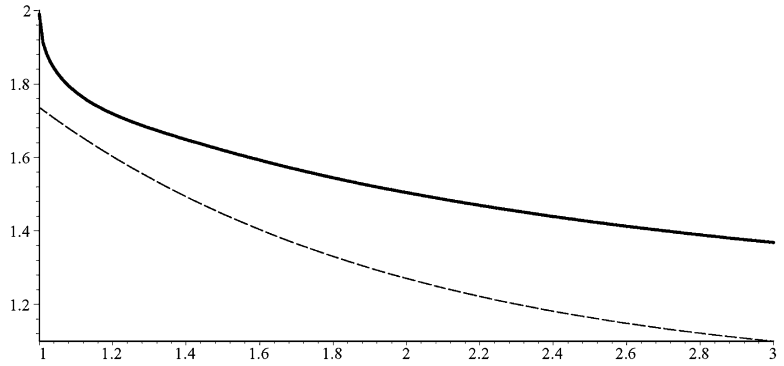

FIG. 2. The tradeoff between γ_f and γ_c.

every solution SOL for $\mathcal{I}$ with facility and connection costs F_{SOL} and C_{SOL}. Then **NP** $\subseteq$ DTIME$[n^{O(\log\log n)}]$.

Similar tradeoff problems are considered by Charikar and Guha [1999]. However, an important advantage that we get here is that all the inequalities $ALG \leq \gamma_f F_{SOL} + \gamma_c C_{SOL}$ are satisfied by a *single* algorithm. In Section 8, we will use the point $\gamma_f = 1$ of this tradeoff to design algorithms for other variants of the facility location problem. Other points of this tradeoff can also be useful in designing other algorithms based on our algorithm. For example, Mahdian et al. [2002] use the point $\gamma_f = 1.1$ of this tradeoff to obtain a 1.52-approximation algorithm for the metric facility location problem.

7. *Experimental Results*

We have implemented our algorithms, as well as the JV algorithm, using the programming language C. We have made four kinds of experiments. In all cases, the solution of the algorithms is compared to the optimal solution of the LP-relaxation, computed using the package CPLEX to obtain an upper bound on the approximation factor of the algorithms.

The test bed of our first set of experiments consists of randomly generated instances on a 10,000 $\times$ 10,000 grid: In each instance, cities and facilities are points, drawn randomly from the grid. The connection cost between a city and a facility is set to be equal to the Euclidean distance of the corresponding points. Furthermore, the opening cost of each facility is drawn uniformly at random from the integers between 0 and 9999.

For the second set of experiments, we have generated random graphs (according to the distribution $G(n, p)$) and assigned uniform random weights on the edges. Cities and facilities correspond to the nodes of this graph, and the connection cost between a city and a facility is defined to be the shortest path between the corresponding nodes. The opening costs of facilities are generated at random.

The instance sizes in both of the above types vary from 50 cities and 20 facilities to 400 cities and 150 facilities. For each size, 15 instances are generated and the average error of the algorithm (compared to the LP lower bound) is computed. The results of these experiments are shown in Table II.

TABLE II. RANDOM GRAPHS AND RANDOM POINTS ON A GRID

		Random Points on a Grid			Random Graphs		
n_c	n_f	JV	ALG 1	ALG 2	JV	ALG 1	ALG 2
50	20	1.0927	1.0083	1.0004	1.0021	1.0007	1.0001
100	20	1.0769	1.0082	1.0004	1.0014	1.0022	1.0
100	50	1.2112	1.0105	1.0013	1.0225	1.0056	1.0005
200	50	1.159	1.0095	1.001	1.0106	1.0094	1.0002
200	100	1.301	1.0105	1.0016	1.0753	1.0178	1.0018
300	50	1.1151	1.0091	1.0011	1.0068	1.0102	1.0002
300	80	1.1787	1.0116	1.001	1.0259	1.0171	1.0004
300	100	1.2387	1.0118	1.0014	1.0455	1.0185	1.0009
300	150	1.327	1.0143	1.0015	1.1365	1.0249	1.0018
400	50	1.0905	1.0092	1.0005	1.0044	1.012	1.0
400	100	1.8513	1.0301	1.0026	1.0313	1.0203	1.0003
400	150	1.8112	1.0299	1.0023	1.1008	1.0234	1.0009

TABLE III. GT-ITM MODEL

n_c	n_f	JV	ALG 1	ALG 2
100	20	1.004	1.0047	1.0001
160	20	1.5116	1.0612	1.0009
160	40	1.065	1.0063	1.0
208	52	2.2537	1.074	1.019
240	60	1.0083	1.0045	1.0001
300	75	1.8088	1.0478	1.0006
312	52	1.7593	1.0475	1.0008
320	32	1.0972	1.0015	1.0
400	100	1.0058	1.0048	1.0
416	52	1.0031	1.0048	1.0

An Internet topology generator software, namely GT-ITM, is used to generate the third set of instances. GT-ITM is a software package for generating graphs that have a structure modeling the topology of the Internet [Zegura et al. 1996]. This model is used because of the applications of facility location problems in network applications such as placing web server replicas [Qiu et al. 2001]. In this model we consider transit nodes as potential facilities and stub nodes as cities. The connection cost is the distance produced by the generator. The opening costs are again random numbers. We have generated 10 instances for each of the 10 different instance sizes. The results are shown in Table III.

We also tested all algorithms on 15 instances from Beasley [2002], which is a library of test data sets for several operations research problems. Our results are shown in Table IV.

As we can see from the tables, Algorithm 2 behaves extremely well, giving almost no error in many cases. Algorithm 1 has an error of 7% on the worst instance and an average error of 2–3%. On the other hand, the JV algorithm has much larger error, sometimes as high as 100%. We should also note that the running times of the three algorithms did not vary significantly. In the biggest instances of 1000 cities and 100 facilities all the algorithms ran in approximately 1–2 seconds. The implementation

TABLE IV. INSTANCES FROM OPERATIONS RESEARCH LIBRARY

n_c	n_f	JV	ALG 1	ALG 2
50	16	1.0642	1.0156	1.0
50	16	1.127	1.0363	1.0
50	16	1.1968	1.0258	1.0
50	16	1.2649	1.0258	1.0022
50	25	1.1167	1.006	1.0028
50	25	1.2206	1.0393	1.0
50	25	1.3246	1.0277	1.0
50	25	1.4535	1.0318	1.0049
50	50	1.3566	1.0101	1.0017
50	50	1.5762	1.0348	1.0061
50	50	1.7648	1.0378	1.0022
50	50	2.0543	1.0494	1.0075
1000	100	1.0453	1.0542	1.0023
1000	100	1.0155	1.0226	1.0
1000	100	1.0055	1.0101	1.0

of the algorithms as well as all the data sets are available upon request. For other experimental results, see Cameron et al. [2002].

8. *Variants of the Problem*

In this section, we show that our algorithms can also be applied to several variants of the metric facility location problem.

8.1. THE k-MEDIAN PROBLEM.

The *k-median problem* differs from the facility location problem in two respects: there is no cost for opening facilities, and there is an upper bound k, that is supplied as part of the input, on the number of facilities that can be opened. The *k-facility location problem* is a common generalization of k-median and the facility location problem. In this problem, we have an upper bound k on the number of facilities that can be opened, as well as costs for opening facilities. The k-median problem is studied extensively [Arya et al. 2001; Charikar and Guha 1999; Charikar et al. 1999; Jain and Vazirani 1999] and the best known approximation algorithm for this problem, due to Arya et al. [2001], achieves a factor of $3+\epsilon$. It is also straightforward to adapt the proof of hardness of the facility location problem [Guha and Khuller 1999] to show that there is no $(1+\frac{2}{e}-\epsilon)$-approximation algorithm for k-median, unless $\mathbf{NP} \subseteq \text{DTIME}[n^{O(\log\log n)}]$. Notice that this proves that k-median is a strictly harder problem to approximate than the facility location problem because the latter can be approximated within a factor of 1.61.

Jain and Vazirani [1999] reduced the k-median problem to the facility location problem in the following sense: Suppose $\mathcal{A}$ is an approximation algorithm for the facility location problem. Consider an instance $\mathcal{I}$ of the problem with optimum cost OPT, and let F and C be the facility and connection costs of the solution found by $\mathcal{A}$. We call algorithm $\mathcal{A}$ a Lagrangian Multiplier Preserving α-approximation (or LMP α-approximation for short) if for every instance $\mathcal{I}$, $C \leq \alpha(OPT - F)$. Jain and Vazirani [1999] show that an LMP α-approximation algorithm for the metric facility location problem gives rise to a 2α-approximation algorithm for the metric

k-median problem. They have noted that this result also holds for the k-facility location problem.

LEMMA 8.1 [JAIN AND VAZIRANI 1999]. *An LMP α-approximation algorithm for the facility location problem gives a 2α-approximation algorithm for the k-facility location problem.*

Here we use Theorem 6.1 together with the scaling technique of Charikar and Guha [1999] to give an LMP 2-approximation algorithm for the metric facility location problem based on Algorithm 2. This will result in a 4-approximation algorithm for the metric k-facility location problem, whereas the best previously known was 6 Jain and Vazirani [1999].

LEMMA 8.2. *Assume there is an algorithm $\mathcal{A}$ for the metric facility location problem such that for every instance $\mathcal{I}$ and every solution SOL for $\mathcal{I}$, $\mathcal{A}$ finds a solution of cost at most $F_{SOL} + \alpha C_{SOL}$, where F_{SOL} and C_{SOL} are facility and connection costs of SOL, and α is a fixed number. Then there is an LMP α-approximation algorithm for the metric facility location problem.*

PROOF. Consider the following algorithm: The algorithm constructs another instance $\mathcal{I}'$ of the problem by multiplying the facility opening costs by α, runs $\mathcal{A}$ on this modified instance $\mathcal{I}'$, and outputs its answer. It is easy to see that this algorithm is an LMP α-approximation. □

Now we only need to prove the following: The proof of this theorem follows the general scheme that is explained in Section 9.

THEOREM 8.3. *For every instance $\mathcal{I}$ and every solution SOL for $\mathcal{I}$, Algorithm 2 finds a solution of cost at most $F_{SOL} + 2C_{SOL}$, where F_{SOL} and C_{SOL} are facility and connection costs of SOL.*

PROOF. By Theorem 6.1 we only need to prove that the solution of the factor-revealing LP 36 with $\gamma_f = 1$ is at most 2. We first write the maximization program 36 as the following equivalent linear program.

$$
\begin{aligned}
\text{maximize} \quad & \sum_{i=1}^{k} \alpha_i - f \\
\text{subject to} \quad & \sum_{i=1}^{k} d_i = 1 \\
& \forall\, 1 \le i < k : \ \alpha_i - \alpha_{i+1} \le 0 \\
& \forall\, 1 \le j < i < k : \ r_{j,i+1} - r_{j,i} \le 0 \\
& \forall\, 1 \le j < i \le k : \ \alpha_i - r_{j,i} - d_i - d_j \le 0 \qquad (37) \\
& \forall\, 1 \le j < i \le k : \ r_{j,i} - d_i - g_{i,j} \le 0 \\
& \forall\, 1 \le i \le j \le k : \ \alpha_i - d_j - h_{i,j} \le 0 \\
& \forall\, 1 \le i \le k : \ \sum_{j=1}^{i-1} g_{i,j} + \sum_{j=i}^{k} h_{i,j} - f \le 0 \\
& \forall\, i, j : \ \alpha_j, d_j, f, r_{j,i}, g_{i,j}, h_{i,j} \ge 0
\end{aligned}
$$

We need to prove an upper bound of 2 on the solution of the above LP. Since this program is a maximization program, it is enough to prove the upper bound for any relaxation of the above program. Numerical results (for a fixed value of k, say $k = 100$) suggest that removing the second, third, and seventh inequalities of the above program does not change its solution. Therefore, we can relax the above program by removing these inequalities. Now, it is a simple exercise to write down the dual of the relaxed linear program and compute its optimal solution. This solution corresponds to multiplying the third, fourth, fifth, and sixth inequalities of the linear program (37) by $1/k$, and the first one by $2 - 1/k$, and adding up these inequalities. This gives an upper bound of $2 - 1/k$ on the value of the objective function. Thus, for $\gamma_f = 1$, we have $\gamma_c \leq 2$. In fact, γ_c is precisely equal to 2, as shown by the following solution for the program (36):

$$\alpha_i = \begin{cases} 2 - \frac{1}{k} & i = 1 \\ 2 & 2 \leq i \leq k \end{cases}$$

$$d_i = \begin{cases} 1 & i = 1 \\ 0 & 2 \leq i \leq k \end{cases}$$

$$r_{j,i} = \begin{cases} 1 & j = 1 \\ 2 & 2 \leq j \leq k \end{cases}$$

$$f = 2(k - 1).$$

This example shows that the above analysis of the factor-revealing LP is tight. □

Lemma 8.2 and Theorem 8.3 provide an LMP 2-approximation algorithm for the metric facility location problem. This result improves all the results in Jain and Vazirani [1999], and gives straightforward algorithms for some other problems considered by Charikar et al. [2001].

Notice that Theorem 6.2 shows that finding an LMP $(1 + \frac{2}{e} - \epsilon)$-approximation for the metric facility location problem is hard. Also, the integrality gap examples found by Guha [2000] show that Lemma 8.1 is tight. This shows that one cannot use Lemma 8.1 as a black box to obtain a smaller factor than $2 + \frac{4}{e}$ for the k-median problem. Note that a $(3 + \epsilon)$-approximation is already known [Arya et al. 2001] for the problem. Hence, if one wants to beat this factor using the Lagrangian relaxation technique, then it will be necessary to look into the underlying LMP algorithm as has already been done by Charikar and Guha [1999].

8.2. FACILITY LOCATION GAME. An important consideration, in cooperative game theory, while distributing the cost of a shared utility, is that the cost shares should satisfy the *coalition participation constraint*, that is, the total cost share of any subset of the users shall not be larger than their stand-alone cost of receiving the service, so as to prevent this subset from seceding. In general, this turns out to be a stringent condition to satisfy. For the facility location problem, Goemans and Skutella [2000] showed that such a cost allocation is only possible for a very special case. Furthermore, intractability sets in as well, for instance, in the case of the facility location problem, computing the optimal cost of serving a set of users is **NP**-hard.

Jain and Vazirani [2001] relax this notion: for a constant k, ensure that the cost share of any subset is no more than k times its stand-alone cost. They also observe that LP-based approximation algorithms directly yield a cost sharing method

compatible with this relaxed notion. However, this involves solving an LP, as in the case of LP-rounding. We observe that our facility location algorithms automatically yield such a cost sharing method, with $k = 1.861$ and $k = 1.61$, respectively, by defining the cost share of city j to be α_j.

8.3. ARBITRARY DEMANDS. In this version, for each city j, a nonnegative integer demand d_j, is specified. An open facility i can serve this demand at the cost of $c_{ij}d_j$. One way to look at this modification is to reduce it to the unit demand case by making d_j copies of city j. This reduction suggests that we need to change our algorithms, so that each city j raises its contribution α_j at rate d_j. It is not difficult to see that the modified algorithms still have the same running time even in more general cases, where d_j is fractional or exponentially large, and achieve the same approximation ratio.

8.4. FAULT-TOLERANT FACILITY LOCATION WITH UNIFORM CONNECTIVITY REQUIREMENTS. We are given a connectivity requirement r_j for each city j, which specifies the number of open facilities to which city j should be connected. We can see that this problem is closely related to the set multicover problem with the additional restriction that every set can be picked at most once [Rajagopalan and Vazirani 1999]. The greedy algorithm for set-cover can be adapted for this variant of the multi-cover problem achieving the same approximation factor. We can use the same approach to deal with fault tolerant facility location: The mechanism of raising dual variables and opening facilities is the same as in our initial algorithms. 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. We can show that when all r_j's are equal, our algorithms still achieve the approximation factors of 1.861 and 1.61.

8.5. FACILITY LOCATION WITH PENALTIES. In this version we are not required to connect every city 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 our algorithms for this problem as follows: If α_j reaches p_j before j is connected to any open facility, the city j 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 variables. At this point, the algorithm terminates and unconnected cities remain unconnected. Using the linear programming formulation introduced in Charikar et al. [2001, inequalities (4.6)–(4.10)], we can show that the approximation ratio and running time of our modified algorithms have not changed.

8.6. ROBUST FACILITY LOCATION. In this variant, we are given a number l and we are only required to connect $n_c - l$ cities to open facilities. This problem can be reduced to the previous one via Lagrangian relaxation. Very recently, Charikar et al. [2001] proposed a primal-dual algorithm, based on the 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 the 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, using Algorithm 1 rather than the JV algorithm. Using a proof similar to the proof of the

Theorem 3.2 in Charikar et al. [2001], we can prove that this algorithm solves the robust facility location problem with an approximation factor of 2.

8.7. DEALING WITH CAPACITIES. In real applications, it is 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*, that is, the cost of serving each city decreases when the number of cities increases (since publication of the first draft of this article, this problem has also been studied in Hajiaghayi et al. [2003]). In order to capture this 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 the metric uncapacitated facility location problem: 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 of this problem can be interpreted as a solution to the original problem with the same cost.

We can reduce the variant of the capacitated facility location problem in which each facility can be opened many times [Jain and Vazirani 1999] to this problem by defining $s_i = f_i/u_i$. If in the solution to this problem k cities are connected to facility i, we open this facility $\lceil k/u_i \rceil$ times. The cost of the solution will be at most two times the original cost so any α-approximation for the uncapacitated facility location problem can be turned into a 2α-approximation for this variant of the capacitated version. We can also use the same technique as in Jain and Vazirani [1999] to give a factor 3-approximation algorithm for this problem based on the LMP 2-approximation algorithm for uncapacitated facility location problem.

9. *Discussion*

The method of dual fitting can be seen as an implementation of the primal-dual schema in which, instead of relaxing complementary slackness conditions (which is the most common way of implementing the schema), we relax feasibility of the dual. However, we prefer to reserve the term, *primal-dual*, for algorithms that produce feasible primal and dual solutions.

Let us show how the combination of dual fitting with factor-revealing LP applies to the set cover problem. The duality-based restatement of the greedy algorithm (see Vazirani [2001]) is: All elements in the universal set U increase their dual variables uniformly. Each element contributes its dual towards paying for the cost of each of the sets it is contained in. When the total contribution offered to a set equals its cost, the set is picked. At this point, the newly covered elements freeze their dual variables and withdraw their contributions from all other sets. As stated in the introduction, the latter (important) step ensures that the primal is fully paid for by the dual. However, we might not get a feasible dual solution. To make the dual solution feasible we look for the smallest positive number Z, so that when the dual solution is shrunk by a factor of Z, it becomes feasible. An upper bound on the approximation factor of the algorithm is obtained by maximizing Z over all possible instances.

Clearly Z is also the maximum factor by which any set is over-tight. Consider any set S. We want to see what is the worst factor, over all sets and over all possible instances of the problem, by which a set S is over-tight. Let the elements in S be $1, 2, \ldots, k$. Let x_i be the dual variable corresponding to the element i at the end of

the algorithm. Without loss of generality we may assume that $x_1 \leq x_2 \leq \cdots \leq x_k$. It is easy to see that at time $t = x_i^-$, total duals offered to S is at least $(k - i + 1)\, x_i$. Therefore, this value cannot be greater than the cost of the set S (denoted by c_S). So, the optimum solution of the following mathematical program gives an upper bound on the value of Z. (Note that c_S is a variable not a constant).

$$\begin{aligned} \text{maximize} \quad & \frac{\sum_{i=1}^{k} x_i}{c_S} \\ \text{subject to} \quad & \forall 1 \leq i < k : \; x_i \leq x_{i+1} \\ & \forall 1 \leq i \leq k : \; (k - i + 1)\, x_i \leq c_S \\ & \forall 1 \leq i \leq k : \; x_i \geq 0 \\ & c_S \geq 1 \end{aligned} \tag{38}$$

The above optimization program can be turned into a linear program by adding the constraint $c_S = 1$ and changing the objective function to $\sum_{i=1}^{k} x_i$. We call this linear program the *factor-revealing LP*. Notice that the factor-revealing LP has nothing to do with the LP formulation of the set cover problem; it is only used in order to analyze this particular algorithm. This is the important distinction between the factor-revealing LP technique, and other LP-based techniques in approximation algorithms.

Formulating the analysis of the algorithm as a factor-revealing LP can be very useful because it can help us to empirically compute the upper bound given by the factor-revealing LP on the approximation ratio of the algorithm. We can also use the empirical results to decide if we need to add more constraints or if we can relax some of them in such a way that the factor does not change drastically. We can also solve its dual for a couple hundred values to observe its trend which will help us to introduce a general dual solution in order to find an upper bound on the solution of the factor-revealing LP and therefore the approximation factor of the algorithm.

One may even get a tight example from a feasible solution of the factor-revealing LP. For example, from any feasible solution x of the factor-revealing LP 38, one can construct the following instance: There are k elements $1, \ldots, k$, a set $S = \{1, \ldots, k\}$ of cost $1 + \epsilon$ which is the optimal solution, and sets $S_i = \{i\}$ of cost x_i for $i = 1, \ldots, k$. It is easy to verify that our algorithm works $\sum x_i$ times worst than the optimal on this instance. This means that the approximation ratio of the set cover algorithm is precisely equal to the solution of the factor-revealing LP, which is H_n.

This seems to be a useful tool for analyzing approximation algorithms. For many algorithms, the proof of the approximation ratio is mainly based on combining several inequalities (usually linear inequalities) to derive a bound on the approximation ratio. It might be possible to "automatize" such proofs using a method similar to the one used in this article. Proving the results in this article would have been very difficult without using these techniques. It would be interesting to find other examples that apply this method.

Algorithms obtained using the method of dual fitting for facility location and set cover problems have additional interesting properties which have been exploited in Devanur et al. [2003] to yield approximate budget balanced strategy-proof mechanisms for the corresponding games.

Finally, in terms of practical impact, what is the significance of improving the approximation guarantee for facility location from 3 to 1.81 or 1.61 when practitioners

are seeking algorithms that come within 2% to 5% of the optimal? The superior experimental results of our algorithms, as compared with the JV algorithm, seem to provide the answer and to support the argument made in Vazirani [2001, Preface, page IX] that the approximation factor should be viewed as a "measure that forces us to explore deeper into the combinatorial structure of the problem and discover more powerful tools for exploiting this structure" and the observation that "sophisticated algorithms do have the error bounds of the desired magnitude, 2% to 5%, on typical instances, even though their worst-case error bounds are much higher."

ACKNOWLEDGMENTS. We would like to thank Michel Goemans, Mohammad Ghodsi, Nicole Immorlica, Nisheeth K. Vishnoi, Milena Mihail, and Christos Gkantsidis for their helpful comments and discussions. We would also like to thank Sudipto Guha for informing us of the OR library. Finally, we thank the anonymous referees for their comments.

REFERENCES

ANDREWS, M., AND ZHANG, L. 1998. The access network design problem. In *Proceedings of the 39th Annual IEEE Symposium on Foundations of Computer Science*. IEEE Computer Society Press, Los Alamitos, Calif.

ARYA, V., GARG, N., KHANDEKAR, R., MEYERSON, A., MUNAGALA, K., AND PANDIT, V. 2001. Local search heuristics for *k*-median and facility location problems. In *Proceedings of 33rd ACM Symposium on Theory of Computing*. ACM, New York.

BALINSKI, M. L. 1966. On finding integer solutions to linear programs. In *Proceedings of the IBM Scientific Computing Symposium on Combinatorial Problems*. 225–248.

BEASLEY, J. E. 2002. Operations research library. http://mscmga.ms.ic.ac.uk/info.html.

CAMERON, C. W., LOW, S. H., AND WEI, D. X. 2002. High-density model for server allocation and placement. In *Proceedings of the 2000 ACM SIG Metrics*. ACM, New York, pp. 152–159.

CHARIKAR, M., AND GUHA, S. 1999. Improved combinatorial algorithms for facility location and *k*-median problems. In *Proceedings of the 40th Annual IEEE Symposium on Foundations of Computer Science*. IEEE Computer Society Press, Los Alamitos, Calif., 378–388

CHARIKAR, M., GUHA, S., TARDOS, E., AND SHMOYS, D. B. 1999. A constant-factor approximation algorithm for the *k*-median problem. In *Proceedings of the 31st Annual ACM Symposium on Theory of Computing*. ACM, New York, 1–10.

CHARIKAR, M., KHULLER, S., MOUNT, D., AND NARASIMHAN, G. 2001. Facility location with outliers. In *12th Annual ACM-SIAM Symposium on Discrete Algorithms* (Washington DC). ACM, New York.

CHUDAK, F. A. 1998. Improved approximation algorithms for uncapacitated facility location. In *Integer Programming and Combinatorial Optimization*, R. E. Bixby, E. A. Boyd, and R. Z. Ríos-Mercado, Eds. Lecture Notes in Computer Science, vol. 1412. Springer-Verlag, Berlin, Germany, Berlin, Germany, 180–194.

CHUDAK, F. A., AND SHMOYS, D. 1998. Improved approximation algorithms for the uncapacitated facility location problem. Unpublished manuscript.

CHVATAL, V. 1979. A greedy heuristic for the set covering problem. *Math. Oper. Res. 4*, 233–235.

CORNUEJOLS, G., NEMHAUSER, G. L., AND WOLSEY, L. A. 1990. The uncapacitated facility location problem. In *Discrete Location Theory*, P. Mirchandani and R. Francis, Eds. Wiley, New York, 119–171.

DEVANUR, N., MIHAIL, M., AND VAZIRANI, V. V. 2003. Strategyproof mechanisms for facility location and set cover games via the method of dual fitting. In *Proceeding of the ACM Conference on Electronic Commerce*. ACM, New York.

GOEMANS, M., AND KLEINBERG, J. 1998. An improved approximation ratio for the minimum latency problem. *Math. Prog. 82*, 111–124.

GOEMANS, M. X., AND SKUTELLA, M. 2000. Cooperative facility location games. In *Proceedings of the Symposium on Discrete Algorithms*. 76–85.

GUHA, S. 2000. Approximation algorithms for facility location problems. PhD dissertation, Stanford Univ., Stanford, Calif.

GUHA, S., AND KHULLER, S. 1999. Greedy strikes back: Improved facility location algorithms. *J. Algorithms 31*, 228–248.

GUHA, S., MEYERSON, A., AND MUNAGALA, K. 2000a. Hierarchical placement and network design problems. In *Proceedings of the 41th Annual IEEE Symposium on Foundations of Computer Science*. IEEE Computer Society Press, Los Alamitos, Calif.

GUHA, S., MEYERSON, A., AND MUNAGALA, K. 2000b. Improved combinatorial algorithms for single sink edge installation problems. Tech. Rep. STAN-CS-TN00 -96, Stanford Univ. Stanford, Calif.

GUHA, S., MEYERSON, A., AND MUNAGALA, K. 2001. Improved algorithms for fault tolerant facility location. In *Proceedings of the Symposium on Discrete Algorithms*. 636–641.

HAJIAGHAYI, M., MAHDIAN, M., AND MIRROKNI, V. 2003. The facility location problem with general cost functions. *Networks 42*, 1 (Aug.), 42–47.

HOCHBAUM, D. S. 1982. Heuristics for the fixed cost median problem. *Math. Prog. 22*, 2, 148–162.

JAIN, K., AND VAZIRANI, V. V. 1999. 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*. IEEE Computer Society Press, Los Alamitos, Calif. 2–13.

JAIN, K., AND VAZIRANI, V. V. 2000. An approximation algorithm for the fault tolerant metric facility location problem. In *Approximation Algorithms for Combinatorial Optimization, Proceedings of APPROX 2000*, K. Jansen and S. Khuller, Eds. Lecture Notes in Computer Science, vol. 1913. Springer-Verlag, New York, 177–183.

JAIN, K., AND VAZIRANI, V. V. 2001. Applications of approximation algorithms to cooperative games. In *Proceedings of the ACM Symposium on Theory of Computing*. ACM, New York, 364–372.

JAMIN, S., JIN, C., JIN, Y., RAZ, D., SHAVITT, Y., AND ZHANG, L. 2000. On the placement of internet instrumentations. In *Proceedings of IEEE INFOCOM'00*. IEEE Computer Society Press, Los Alamitos, Calif. 26–30.

KAUFMAN, L., VAN EEDE, M., AND HANSEN, P. 1977. A plant and warehouse location problem. *Oper. Res. Quart. 28*, 547–557.

KORUPOLU, M. R., PLAXTON, C. G., AND RAJARAMAN, R. 1998. Analysis of a local search heuristic for facility location problems. In *Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms*. ACM, New York, 1–10.

KUEHN, A. A., AND HAMBURGER, M. J. 1963. A heuristic program for locating warehouses. *Manag. Sci. 9*, 643–666.

LI, B., GOLIN, M., ITALIANO, G., DENG, X., AND SOHRABY, K. 1999. On the optimal placement of web proxies in the internet. In *Proceedings of IEEE INFOCOM'99*. IEEE Computer Society Press, Los Alamitos, Calif., 1282–1290.

LOVASZ, L. 1975. On the ratio of optimal integral and fractional covers. *Disc. Math. 13*, 383–390.

MAHDIAN, M., YE, Y., AND ZHANG, J. 2002. Improved approximation algorithms for metric facility location problems. In *Proceedings of 5th International Workshop on Approximation Algorithms for Combinatorial Optimization (APPROX 2002)*.

MCELIECE, R. J., RODEMICH, E. R., RUMSEY JR., H., AND WELCH, L. R. 1977. New upper bounds on the rate of a code via the delsarte-macwilliams inequalities. *IEEE Trans. Inf. Theory 23*, 157–166.

METTU, R. R., AND PLAXTON, G. 2000. The online median problem. In *Proceedings of the IEEE Symposium on Foundations of Computer Science*. IEEE Computer Society Press, Los Alamitos, Calif., 339–348.

NEMHAUSER, G. L., AND WOLSEY, L. A. 1990. *Integer and Combinatorial Optimization*. Wiley, New York.

QIU, L., PADMANABHAN, V. N., AND VOELKER, G. M. 2001. On the placement of web server replicas. In *Proceedings of IEEE INFOCOM* (Anchorage, Ak.). IEEE Computer Society Press, Los Alamitos, Calif., 1587–1596.

RAJAGOPALAN, S., AND VAZIRANI V. V. 1999. Primal-dual RNC approximation algorithms for set cover and covering integer programs. *SIAM J. Comput. 28*, 2, 525–540.

SHMOYS, D. B., TARDOS, E., AND AARDAL, K. I. 1997. Approximation algorithms for facility location problems. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*. ACM, New York, 265–274.

STOLLSTEIMER, J. F. 1961. The effect of technical change and output expansion on the optimum number, size and location of pear marketing facilities in a California pear producing region. PhD dissertation. University of California at Berkeley.

STOLLSTEIMER, J. F. 1963. A working model for plant numbers and locations. *J. Farm Econom. 45*, 631–645.

SVIRIDENKO, M. 2002. An improved approximation algorithm for the metric uncapacitated facility location problem. In *Proceedings of the Ninth Conference on Integer Programming and Combinatorial Optimization*. 240–257.

THORUP, M. 2001. Quick *k*-median, *k*-center, and facility location for sparse graphs. In *Automata, Languages and Programming, 28th International Colloquium* (Crete, Greece). Lecture Notes in Computer Science, vol. 2076. Springer-Verlag, New York, 249–260.

VAZIRANI, V. V. 2001. *Approximation Algorithms*. Springer-Verlag, Berlin, Germany.

ZEGURA, E. W., CALVERT, K. L., AND BHATTACHARJEE, S. 1996. How to model an internetwork. In *Proceedings of IEEE INFOCOM*. Vol. 2 (San Francisco, Calif.) IEEE Computer Society Press, Los Alamitos, Calif. 594–602.

RECEIVED JULY 2002; REVISED JULY 2003; ACCEPTED JULY 2003