A constant-factor approximation algorithm for the k-median problem

Moses Charikar* Sudipto Guha† Éva Tardos‡ David B. Shmoys§

July 23, 2002

Abstract

We present the first constant-factor approximation algorithm for the metric k-median problem. The k-median problem is one of the most well-studied clustering problems, i.e., those problems in which the aim is to partition a given set of points into clusters so that the points within a cluster are relatively close with respect to some measure. For the metric k-median problem, we are given n points in a metric space. We select k of these to be cluster centers, and then assign each point to its closest selected center. If point j is assigned to a center i, the cost incurred is proportional to the distance between i and j. The goal is to select the k centers that minimize the sum of the assignment costs. We give a $6\frac{2}{3}$-approximation algorithm for this problem. This improves upon the best previously known result of $O(\log k \log \log k)$, which was obtained by refining and derandomizing a randomized $O(\log n \log \log n)$-approximation algorithm of Bartal.

1 Introduction

For the *metric k-median problem*, we are given n points in a metric space. We select at most k of these to be cluster centers, and then assign each input point j to the selected center that is closest to it. If location j is assigned to a center i, we incur a cost proportional to the distance between i and j. The goal is to select the k centers so as to minimize the sum of the assignment costs. We give a $6\frac{2}{3}$-approximation algorithm for this problem, that is, a polynomial-time algorithm that finds a feasible solution of objective function value within a factor of $6\frac{2}{3}$ of the optimum.

Lin & Vitter [25] considered the k-median problem with arbitrary assignment costs, and gave a polynomial-time algorithm that finds, for any $\epsilon > 0$, a solution for which the objective function value is within a factor of $1+\epsilon$ of the optimum, but is infeasible: it opens as many as $(1 + 1/\epsilon)(\ln n + 1)k$ cluster centers. Lin & Vitter also provided evidence that this result is best possible via a reduction from the set cover problem.

*moses@cs.stanford.edu. Stanford University, Stanford, CA 94305. Research supported by the Pierre and Christine Lamond Fellowship, NSF Grant IIS-9811904 and NSF Award CCR-9357849, with matching funds from IBM, Mitsubishi, Schlumberger Foundation, Shell Foundation, and Xerox Corporation.

†sudipto@cs.stanford.edu. Stanford University, Stanford, CA 94305. Research Supported by IBM Cooperative Fellowship, NSF Grant IIS-9811904 and NSF Award CCR-9357849, with matching funds from IBM, Mitsubishi, Schlumberger Foundation, Shell Foundation, and Xerox Corporation.

‡eva@cs.cornell.edu. Cornell University, Ithaca, NY 14853. Research partially supported by NSF grants CCR-9700163 ONR grants N00014-98-1-0589 and N00014-96-1-0050.

§shmoys@cs.cornell.edu. Cornell University, Ithaca, NY 14853. Research partially supported by NSF grants CCR-9912422 and DMS-9505155 and ONR grant N00014-96-1-00500.

Consequently, it is quite natural to consider special cases. The problem is solvable in polynomial time on trees [21, 30]. However, for general metric spaces, the problem is NP-hard to solve exactly. Arora, Raghavan & Rao [1] give a polynomial-time approximation scheme for the k-median problem with 2-dimensional Euclidean inputs.

We study the metric k-median problem, that is, we assume that the input points are located in a metric space, or in other words, assume that the assignment costs are symmetric and satisfy the triangle inequality. Lin & Vitter [24] also gave a polynomial-time algorithm for the metric k-median problem that, for any $\epsilon > 0$, finds a solution of cost no more than $2(1+\epsilon)$ times the optimum, while using at most $(1+1/\epsilon)k$ cluster centers. The first non-trivial approximation algorithm that produces a feasible solution (i.e., one that uses at most k centers) is due to Bartal [4, 5]. By combining his result on the approximation of any metric by tree metrics with the fact that the k-median problem can be solved optimally in a tree metric, Bartal gave a randomized $O(\log n \log\log n)$-approximation algorithm for the k-median problem. This algorithm was subsequently derandomized and refined to yield an $O(\log k \log\log k)$-approximation algorithm by Charikar, Chekuri, Goel, & Guha [7].

Approximation algorithms have been studied for a variety of clustering problems. The *k-center problem* is the min-max analogue of the k-median problem: one opens centers at k locations out of n, so as to minimize the maximum distance that an unselected location is from its nearest center. Hochbaum & Shmoys [18] and subsequently Dyer & Frieze [13] gave 2-approximation algorithms for the metric case problem (which is best possible unless $\mathcal{P} = \mathcal{NP}$), and also gave extensions for weighted variants. Gonzalez [15] considered the variant in which the objective is to minimize maximum distance between a pair of points in the same cluster, and independently gave a 2-approximation algorithm (which is also best possible).

The k-median problem is closely related to the *uncapacitated facility location problem*, which is a central problem in the Operations Research literature (see, e.g., the survey of Cornuéjols, Nemhauser, & Wolsey [12]). In this problem, each location has a cost f_i, the cost of opening a center (or facility) at location i. There is no restriction on the number of facilities that can be opened, but instead the goal is to minimize the total cost: the cost of the selected facilities plus the sum of the assignment cost of each location to its closest open facility, where we assume that the cost of assigning location j to facility i is proportional to the distance between i and j. Shmoys, Tardos, & Aardal [28] use the techniques of Lin & Vitter [24] to give the first constant-factor approximation algorithm for the metric uncapacitated facility location problem. The quality of approximation has been improved in a sequence of papers [17, 9, 10]. The best result previously known is a $(1 + 2/e)$-approximation algorithm due to Chudak & Shmoys [9, 10] (though, as we shall briefly discuss below, subsequent to the proceedings version of this paper, some improvements have been obtained). Guha & Khuller [17] have shown two hardness of approximation results as well: the problem is MAX SNP-hard (which was independently observed by Sviridenko [29]), and even more surprisingly, for any $\rho < 1.463$, no ρ-approximation algorithm exists unless unless $\mathcal{NP} \subseteq DTIME(n^{O(\log\log n)})$. Subsequently, Sviridenko [29] and Feige [14] indicated that the hardness result may be strengthened to depend only on the assumption that $\mathcal{P} \neq \mathcal{NP}$. All of these algorithms rely on solving a natural linear programming relaxation of the problem and rounding the optimal fractional solution. Korupolu, Plaxton, & Rajaraman [23] analyze variants of simple local search algorithms for several clustering problems and show, for example, that for any $\epsilon > 0$, this leads to a $(5 + \epsilon)$-approximation algorithm for the uncapacitated facility location problem.

Our algorithms are based on the approach of solving a natural linear programming relaxation of the problem, and rounding the optimal fractional solution. The linear programming relaxation is analogous to the relaxation used by the approximation algorithms for the facility location problem, and has also been studied for the k-median problem on trees [32, 31].

We obtain our result by combining the filtering technique of Lin & Vitter [25] with a more sophisticated method for selecting which centers to open. The filtering technique of Lin & Vitter [25] guarantees that the cost of the solution does not exceed the LP optimum by too much by making sure that in the integer solution, each location pays an assignment cost not too much more than the corresponding part of the cost of the optimal fractional solution. This kind of location-by-location guarantee is not possible for the k-median problem: some locations will necessarily pay a significantly greater assignment cost in the integer solution than in the fractional solution. However, we show how to select the centers so that there is only a small constant increase in the average assignment cost.

Our results can be extended in a number of ways. For example, if one relaxes the condition that the costs satisfy the triangle inequality to require only that $c_{ik} \leq \delta(c_{ij} + c_{jk})$ for each i, j, k, then an analogously weaker, but still constant, performance guarantee holds instead. One direct application of this observation is to obtain a constant performance guarantee for the variant of the k-median problem where the objective is to minimize the sum of the squares of the distances from the vertices to their nearest centers. More substantially, Charikar & Guha [6, 16] consider a capacitated version of the k-median problem, in which each center can be assigned at most U locations. Charikar & Guha give a polynomial-time algorithm that finds a solution of cost at most 16 times the true optimal cost, but allows centers to be assigned at most $4U$ locations.

There has also been a great deal of subsequent work to improve on our results. Jain & Vazirani [20] give an extremely elegant primal-dual 3-approximation algorithm for the uncapacitated facility location problem, and show how to use that procedure to obtain a 6-approximation algorithm for the k-median problem. Charikar & Guha [8] refine this result to obtain a 4-approximation algorithm for the k-median problem, in addition to providing a number of new techniques that, combined with the previously known algorithms, yield a 1.728-approximation algorithm for the uncapacitated facility location problem. For the median problem, Mettu & Plaxton [26] give the quite surprising result that one can efficiently compute a *single* permutation of the locations such that, for *every* $k > 0$, the first k locations in this order serve as a k-median solution of cost that is within a constant factor of optimal. Most recently, Arya, Garg, Khandekar, Meyerson, Munagala, and Pandit [2] showed that the natural local search heuristic that interchanges p locations as centers/non-centers is guaranteed to find a solution that is within a factor of $3 + (2/p)$, for any positive integer p. On the other hand, Jain, Mahdian, and Saberi [19] have shown that no performance guarantee better than $1 + (2/e)$ can be obtained in polynomial time, unless $NP \subseteq DTIME[n^{O(\log\log n)}]$.

2 The k-median problem

It is more natural to state our algorithm in terms of a slight generalization of the usual k-median problem, which we described in the introduction. We shall let the input consist of a set of locations N and a bound k, where there is a specified assignment cost c_{ij} between each pair of points $i, j \in N$. In addition, we shall also be given a demand d_j for each location $j \in N$; this demand can either be viewed as a weight that indicates the importance of the location, or as specifying the number of clients present at that location. The usual statement of the k-median problem corresponds to the special case in which $d_j = 1$, for each $j \in N$. Note that this generalization also allows the possibility that there are locations with no demand (i.e., where $d_j = 0$), but can still be selected as centers.

We shall assume that the assignment costs are non-negative, symmetric, and satisfy the triangle inequality: that is, $c_{ij} = c_{ji}$ for each $i, j \in N$, and $c_{ij} + c_{jk} \geq c_{ik}$ for each $i, j, k \in N$. The problem is to select k of the locations as centers, and assign each location in N to one of the k selected centers so as to minimize the total weighted assignment cost incurred.

The k-median problem can be stated as the following integer program, where the 0-1 variable y_i, $i \in N$, indicates if the location i is selected as a center, and the 0-1 variable x_{ij}, $i,j \in N$, indicates if location j is assigned to the center at i:

$$\text{minimize} \sum_{i,j \in N} d_j c_{ij} x_{ij} \tag{1}$$

subject to

$$\begin{aligned}
\sum_{i \in N} x_{ij} &= 1, && \text{for each } j \in N, && (2)\\
x_{ij} &\leq y_i, && \text{for each } i,j \in N, && (3)\\
\sum_{i \in N} y_i &\leq k, && && (4)\\
x_{ij} &\in \{0,1\}, && \text{for each } i,j \in N, && (5)\\
y_i &\in \{0,1\}, && \text{for each } i \in N. && (6)
\end{aligned}$$

The constraints (2) ensure that each location $j \in N$ is assigned to some center $i \in N$, the constraints (3) ensure that whenever a location j is assigned to a center i, then a center must have been opened at i, and (4) ensures that at most k centers are open.

2.1 Outline of the method

In this section we will give an outline of the various steps that we will use to prove the performance guarantee. We will indicate how the different steps interact, relegating the details of the individual steps to the subsequent sections.

Consider the linear programming relaxation to the integer program (1)–(6), where the 0-1 constraints (5) and (6) are replaced, respectively, with

$$\begin{aligned}
x_{ij} &\geq 0, && \text{for each } i,j \in N, && (7)\\
y_i &\geq 0, && \text{for each } i \in N. && (8)
\end{aligned}$$

Let $(\bar{x}, \bar{y})$ denote a feasible solution to the LP relaxation and let $\bar{C}_{LP}$ denote its objective function value. For each location $j \in N$, let $\bar{C}_j$ denote the cost incurred by this fractional solution for assigning (one client at) location j, i.e.,

$$\bar{C}_j = \sum_{i \in N} c_{ij} \bar{x}_{ij} \qquad \text{for each } j \in N. \tag{9}$$

Note that $\bar{C}_{LP} = \sum_{j \in N} d_j \bar{C}_j$. We will round the fractional solution $(\bar{x}, \bar{y})$ to a feasible solution to the integer program of cost at most $6\frac{2}{3}\bar{C}_{LP}$. The outline of our approximation algorithm is as follows.

Step 1: First we simplify the problem instance by consolidating nearby locations. We will not change the linear programming solution $(\bar{x}, \bar{y})$ but modify only the demands. Let d' denote the new set of demands. This modification will have the following properties:

- The modification will not increase the cost of the fractional solution $(\bar{x}, \bar{y})$:

$$\bar{C}'_{LP} = \sum_{i,j \in N} d'_j c_{ij} \bar{x}_{ij} \leq \sum_{i,j \in N} d_j c_{ij} \bar{x}_{ij} = \bar{C}_{LP}.$$

- In the resulting instance, all locations with positive demand will be far from each other:

$$c_{ij} > 4\max(\bar{C}_i, \bar{C}_j) \text{ for each } i, j \in N \text{ such that } d'_i > 0 \text{ and } d'_j > 0. \tag{10}$$

- Each feasible integer solution for the modified instance with demands d'_j, $j \in N$, can be converted to a feasible integer solution for the original instance, at an added cost of at most $4\bar{C}_{LP}$.

Let $N' = \{j : d'_j > 0\}$ denote the set of locations with positive demand. We will show that this modified instance is simpler, e.g., $|N'| \leq 2k$.

Step 2: Next, we simplify the structure of the solution by consolidating nearby fractional centers. First, we modify the solution $(\bar{x}, \bar{y})$ to obtain a new solution (x', y') such that

$$y'_j = 0, \quad \text{for each } j \in N \text{ such that } d'_j = 0, \tag{11}$$

$$y'_i \geq \frac{1}{2}, \quad \text{for each } i \in N \text{ such that } d'_i > 0. \tag{12}$$

We will refer to such a solution as a $\frac{1}{2}$-*restricted solution.* Further, the cost of the $\frac{1}{2}$-restricted solution produced is at most $2\bar{C}_{LP}$.

We further modify the $\frac{1}{2}$-restricted solution, without increasing its cost, to obtain an $\{\frac{1}{2}, 1\}$-*integral solution* $(\hat{x}, \hat{y})$: that is, $\hat{y}_i = 0$ for each $i \notin N'$ and $\hat{y}_i$ is either $\frac{1}{2}$ or 1 for each $i \in N'$.

Step 3: Finally, we show how to convert a feasible $\{\frac{1}{2}, 1\}$-integral solution to the linear programming relaxation to a feasible integral solution of cost at most $\frac{4}{3}$ times the cost of the $\{\frac{1}{2}, 1\}$-integral solution.

Using these steps, we obtain the following theorem.

Theorem 1 *For the metric k-median problem, the outlined method yields a $6\frac{2}{3}$-approximation algorithm.*

Proof: Suppose that the feasible solution $(\bar{x}, \bar{y})$ to be rounded is an optimal solution to the linear program (1)–(4) and (7)–(8). Clearly, $\bar{C}_{LP}$ is a lower bound on the optimal value of the k-median problem. Steps 1–3 above create a feasible integer solution to the modified input of cost at most $2\frac{2}{3}\bar{C}_{LP}$. This can then be converted to an integer solution to the original instance while adding at most $4\bar{C}_{LP}$ to the cost. This results in a feasible integer solution of cost at most $6\frac{2}{3}\bar{C}_{LP}$, i.e., at most $6\frac{2}{3}$ times the minimum possible. □

3 Step 1: Consolidating locations

Recall that $\bar{C}_j = \sum_{i \in N} c_{ij}\bar{x}_{ij}$ is the cost that the linear program pays for assigning one unit of demand at location j. The goal of the first step of the method is to consolidate demands at nearby locations. More formally, we want to guarantee that for all pairs of locations i, j both with positive demand, $c_{ij} > 4\max(\bar{C}_i, \bar{C}_j)$. The modified instance of the location problem is obtained without changing the LP solution $(\bar{x}, \bar{y})$. For some locations j, we will simply "move" the demand d_j to a nearby location $j' \in N$, which is no more than $4\bar{C}_j$ away from location j.

Assume for notational simplicity that $N = \{1, \ldots, n\}$, and the locations are indexed in increasing order of $\bar{C}_j$, i.e., $\bar{C}_1 \leq \ldots \leq \bar{C}_n$. We create the modified instance as follows. Start with $d'_j \leftarrow d_j$, for each location $j \in N$.

- Consider the locations $j = 1, \ldots, n$ in this order. When considering location j, check if there is another location $i < j$ such that $d'_i > 0$ and $c_{ij} \le 4\bar{C}_j$. If there is such a location, then add the demand of location j to one such location i: select some i for which $c_{ij} \le 4\bar{C}_j$, and set

$$d'_i \leftarrow d'_i + d'_j; \tag{13}$$
$$d'_j \leftarrow 0. \tag{14}$$

Let d' denote the resulting demands, whereas d denotes the original demands. Let N' denote the set of locations with $d'_j > 0$, i.e., $N' = \{j \in N : d'_j > 0\}$. Note that $(\bar{x}, \bar{y})$, the feasible solution to the LP relaxation with the original demands, is also a feasible solution for the modified input. The next lemma follows directly from the algorithm.

Lemma 2 *Locations $i, j \in N'$ satisfy (10).*

Lemma 3 *The cost of the fractional solution $(\bar{x}, \bar{y})$ for the input with modified demands is at most its cost for the original input, that is,*

$$\sum_{i,j \in N} d'_j c_{ij} \bar{x}_{ij} \le \sum_{i,j \in N} d_j c_{ij} \bar{x}_{ij}. \tag{15}$$

Proof: The LP objective function value (1) of the solution $(\bar{x}, \bar{y})$, for any demands $\bar{d}$, can be written as $\sum_{j \in N} \bar{d}_j \bar{C}_j$. The lemma now follows directly, since d' is obtained by moving demand from a location j to a location i with $\bar{C}_i \le \bar{C}_j$. □

Lemma 4 *For any feasible integer solution (x', y') for the modified input with demands d', there is a feasible integer solution for the original input of cost at most $4\bar{C}_{LP}$ more than the cost of (x', y') with demands d'.*

Proof: Consider a location $j \in N$ that has its demand moved to j' in the modified input. Each location j was moved by at most $4\bar{C}_j$, and so we have that $c_{j'j} \le 4\bar{C}_j$. If j is assigned to the center to which j' is assigned by x', then, by the triangle inequality, restoring j to its original position increases its unit assignment cost by at most $4\bar{C}_j$. Summing over all j, we obtain the desired bound. □

4 Step 2: Consolidating centers

In this section, we consider the problem with demands d'_j for $j \in N'$ as defined in the previous section, i.e., we assume that the modified demands satisfy (10). Recall, from the outline, that we have a solution $(\bar{x}, \bar{y})$ at this point; these variables define the quantities used in (10).

We simplify the structure of the solution by consolidating nearby fractional centers. First, we modify the solution $(\bar{x}, \bar{y})$ to obtain a new solution (x', y') of cost at most $2\bar{C}_{LP}$ such that $y'_i = 0$ for each $i \notin N'$ and $y'_i \ge \frac{1}{2}$ for each $i \in N'$.

We further modify the $\frac{1}{2}$-restricted solution to obtain, without increasing its cost, a $\{\frac{1}{2}, 1\}$-integral solution $(\hat{x}, \hat{y})$: that is, $\hat{y}_i = 0$ for each $i \notin N'$ and $\hat{y}_i$ is either $\frac{1}{2}$ or 1 for each $i \in N'$.

The first step will be accomplished using the filtering technique of Lin & Vitter [25]. The main observation behind this technique is the following lemma, which shows that each location j has at least half of its demand assigned to relatively "nearby" partially open centers: for each $j \in N$, there is a total of at least 1/2 of an open center within a radius of $2\bar{C}_j$.

Lemma 5 *For any feasible fractional solution* $(\bar{x}, \bar{y})$, $\sum_{i:c_{ij} \le 2\bar{C}_j} \bar{y}_i \ge \frac{1}{2}$ *for each* $j \in N$.

Proof: Recall that $\bar{C}_j = \sum_{i \in N} c_{ij}\bar{x}_{ij}$. In any weighted average, less than half of the total weight can be given to values more than twice the average; that is,

$$\sum_{i:c_{ij} > 2\bar{C}_j} \bar{x}_{ij} < \frac{1}{2}. \tag{16}$$

Since $\sum_{i \in N} \bar{x}_{ij} = 1$ and $\bar{x}_{ij} \le \bar{y}_i$ for each $i \in N$,

$$\sum_{i:c_{ij} \le 2\bar{C}_j} \bar{y}_i \ge \sum_{i:c_{ij} \le 2\bar{C}_j} \bar{x}_{ij} \ge \frac{1}{2}. \tag{17}$$

This proves the lemma. □

Now we are ready to prove the following result.

Theorem 6 *There is a* $\frac{1}{2}$*-restricted solution* (x', y') *of cost at most* $2\bar{C}_{LP}$.

Proof: We will modify the solution $(\bar{x}, \bar{y})$ by moving each fractional center to the location in N' closest it. We will prove that this produces a $\frac{1}{2}$-restricted solution of cost at most twice the cost of the solution $(\bar{x}, \bar{y})$.

We start by setting $x' \leftarrow \bar{x}$ and $y' \leftarrow \bar{y}$. First consider only the constraints (11). For each location $i \in N$ at which there is a partially open center (i.e., $y'_i > 0$) and yet $d'_i = 0$, we completely close the fractional center at location i, and move it to the closest location $j \in N'$. The fractional center at i is moved to j by setting

$$\begin{aligned} y'_j &\leftarrow \min(1, y'_i + y'_j); & (18)\\ y'_i &\leftarrow 0. & (19) \end{aligned}$$

When we move the fractional center at i to location j, then we also change the assignments accordingly; that is, for each location $j' \in N$, we set

$$\begin{aligned} x'_{jj'} &\leftarrow x'_{jj'} + x'_{ij'}; & (20)\\ x'_{ij'} &\leftarrow 0. & (21) \end{aligned}$$

First observe that these changes result in a feasible linear programming solution that also satisfies the constraints (11). We will worry about the constraints (12) later.

Next we show that these changes at most double the cost of the solution. Consider some location j' with positive demand $d'_{j'}$. For any fractional center i that is moved, say, to location j, consider the corresponding change in the assignment cost of j'. The fraction $x_{ij'}$ previously assigned to i is now assigned to j instead. By the triangle inequality, we get that $c_{jj'} \le c_{ij} + c_{ij'}$. By the fact that the fractional center at i is moved to j, and not to j', we know that $c_{ij} \le c_{ij'}$. Hence, $c_{jj'} \le 2c_{ij'}$. By considering all demand points j' and all fractional centers i that are moved, we see that this claim implies that the cost of (x', y') is at most twice the cost of $(\bar{x}, \bar{y})$.

Finally, we must show that the new solution satisfies the constraints (12). Note that by Step 1, any two locations with positive demand are far apart, i.e., the demands d' satisfy (10). This implies that for each location j with positive demand, all partially opened centers within a $2\bar{C}_j$ radius of j must be closer to j than to any other location in N'; hence all of these will be moved to j. Lemma 5 implies that, the sum of the fractional center values on the locations within this radius of j is at least $\frac{1}{2}$. All of these fractional centers are moved to j, and hence we see that $y'_j \geq 1/2$, as required. $\square$

Note that the existence of a $\frac{1}{2}$-restricted solution for the modified instance implies that there are at most $2k$ locations with positive modified demand. If we open a center at each such location, we would get a solution (for the modified input) of cost 0. This implies that the original input has a solution using at most $2k$ centers that costs at most $4\bar{C}_{LP}$.

Next, we obtain a $\{\frac{1}{2}, 1\}$-integral solution for the modified instance. We begin by examining the structure of any $\frac{1}{2}$-restricted solution. For any $\frac{1}{2}$-restricted solution with fractional center values y, it is easy to express the corresponding optimal fractional assignment x. Consider a location $j \in N'$. The optimal linear programming solution will have $x_{jj} = y_j$, i.e., each location in N' will use its own partially open center to the maximum extent possible. Since $y_j \geq 1/2$, this leaves less than half of j left to be assigned, and so, for any $k \in N'$, we can set $x_{kj} = 1 - x_{jj} (= 1 - y_j)$, and be sure that the resulting solution is feasible (i.e., $x_{kj} \leq 1/2 \leq y_k$). The best alternative is to let k be the location in N' that is closest to j (other than j itself). For each $j \in N'$, let $s(j)$ be the closest location to j in N' (other than j), where ties are broken by choosing the location with smallest index. We have just proved the following lemma.

Lemma 7 *The minimum cost of a $\frac{1}{2}$-restricted solution (x', y') is $\sum_{j \in N'} d'_j (1 - y'_j) c_{s(j)j}$.* $\square$

This lemma implies that we can view the cost of a $\frac{1}{2}$-restricted solution (x', y') as a function solely of the values y'. Using this lemma, we are ready to prove the following theorem.

Theorem 8 *For any $\frac{1}{2}$-restricted solution (x', y'), there exists a $\{\frac{1}{2}, 1\}$-integral solution of no greater cost.*

Proof: By Lemma 7, the cost of the $\frac{1}{2}$-restricted solution (x', y') can be expressed as

$$\sum_{j \in N'} d'_j c_{s(j)j} - \sum_{j \in N'} d'_j c_{s(j)j} y'_j. \tag{22}$$

We will construct a $\frac{1}{2}$-restricted solution whose cost is minimum possible. The first term of (22) is a constant independent of y'. The second term is maximized if the values y'_j are as large as possible for locations $j \in N'$ with the largest values $d'_j c_{s(j)j}$. The $\frac{1}{2}$-restricted solution of minimum cost is obtained as follows. Let $n' = |N'|$. Sort the locations $j \in N'$ in decreasing order of their weight $d'_j c_{s(j)j}$. Set $\hat{y}_j = 1$ for the first $2k - n'$ locations, and set $\hat{y}_j = 1/2$ for the remaining $2(n' - k)$ locations. The solution $(\hat{x}, \hat{y})$ is a $\{\frac{1}{2}, 1\}$-integral solution. By the above discussion, the cost of $(\hat{x}, \hat{y})$ is at most the cost of the $\frac{1}{2}$-restricted solution (x', y'). $\square$

Thus, Step 2 produces a $\{\frac{1}{2}, 1\}$-integral solution $\hat{y}_j$, $j \in N'$, so that each location $j \in N'$ is assigned to itself and at most one other location $s(j)$. Of course, if $\hat{y}_j = 1$, then there is no need to consider $s(j)$ at all; we will adopt the convention that if $\hat{y}_j = 1$, then $s(j) = j$.

5 Step 3: Rounding a $\{\frac{1}{2}, 1\}$-integral solution to an integral one

Finally, we show how to convert the $\{\frac{1}{2}, 1\}$-integral solution obtained in the previous section to an integral solution. We will first show a simple method that increases the cost incurred by a factor of at most 2. Then we shall explain how to convert to an integer solution while increasing the cost incurred by a factor of at most $\frac{4}{3}$.

Consider the $\{\frac{1}{2}, 1\}$-integral solution $(\hat{x}, \hat{y})$ to the modified instance. Recall the structure of an optimal $\{\frac{1}{2}, 1\}$-integral solution. For each location $i \in N'$, the optimal assignment is to set $\hat{x}_{ii} = \hat{y}_i$, and if $\hat{y}_i = \frac{1}{2}$, to then set $\hat{x}_{ki} = \frac{1}{2}$, where k is defined to be the closest other location, $s(i)$, in N'. We use these $(i, s(i))$ pairs as edges to obtain a collection of trees spanning the vertices in N'. We then use the collection of trees to obtain an integral solution.

More precisely, we build a collection of trees as follows. For each node $i \in N'$ with $\hat{y}_i = \frac{1}{2}$, draw a directed edge from i to $s(i)$. By the definition of $s(i)$, each component of this graph contains at most one cycle; such a cycle must be of length 2, and corresponds to the closest pair of nodes in the graph. For each cycle, we choose one of the two vertices as a root and delete the directed edge from the root to the other vertex. This will result in a collection of rooted trees. We say that $s(i)$ is the parent of i, if we have a directed edge from i to $s(i)$.

We first describe a simple rounding scheme that produces an integral solution of cost at most twice the cost of the $\{\frac{1}{2}, 1\}$-integral solution. Define the *level* of any node to be the number of edges on the path from the node to the root.

The locations of the centers are chosen as follows. We build a center at each node i such that $\hat{y}_i = 1$. We partition the nodes $\{i \in N' | \hat{y}_i = \frac{1}{2}\}$ into two subsets corresponding to the odd and even levels and build a center at each node in the smaller of the two subsets. This ensures that the number of centers built is at most $\sum \hat{y}_i \leq k$.

It is easy to argue that this rounding at most doubles the cost of the solution. We build a center at each node i for which $\hat{y}_i = 1$, and hence the contribution of node i to the cost of the solution is 0. For each node i with $\hat{y}_i = \frac{1}{2}$, either i or $s(i)$ is chosen as a center (since one of i and $s(i)$ is at an odd level and the other is at an even level). Hence the contribution of i to the cost of the solution is at most $d'_i c_{s(i)i}$, which is exactly twice its contribution to the cost of the $\{\frac{1}{2}, 1\}$-integral solution.

Thus, the cost of the integral solution produced is at most twice the cost of the $\{\frac{1}{2}, 1\}$-integral solution. If we substitute this bound into the proof of Theorem 1, we get an 8-approximation algorithm for the k-median problem.

Next we describe a procedure for rounding the $\{\frac{1}{2}, 1\}$-integral solution for which one can prove a sharper performance guarantee. Although we have been considering costs c_{ij} derived from an arbitrary metric, the k-median problem has been studied with respect to special classes of metrics. In particular, for a given set of locations N, if there is a tree $T = (N, E)$, and the distance between any pair of locations $i, j \in N$ is defined as the length of the path between i and j in T, then the resulting metric is said to be a *tree metric*. The k-median problem for tree metrics is known to be solvable in polynomial time (see, for example, the work of Tamir[30]); furthermore, the ratio between the optimal integer and fractional solutions to (1)–(6) is known to be at most $2 - \frac{2}{k+1}$ [32, 31]. We shall consider a slightly more general class of metrics, in which there is a forest and the distance between a pair of points in the same component is defined to be the length in that tree, whereas the distance between two points in different components is infinity (or any suitably large constant). It is straightforward to see that the polynomial-time optimization algorithm for the k-median problem with respect to tree metrics can be generalized to this *forest metric* setting as well.

We gave an efficient rounding procedure to convert the $\{\frac{1}{2}, 1\}$-integral solution to an integral one of at most twice the cost. We shall argue next that even a non-constructive proof of the fact that, for any forest metric, the ratio between the costs of the optimal integer and $\{\frac{1}{2}, 1\}$-integral solutions is at most ρ, yields a polynomial-time algorithm that applies to *any* metric with the same guarantee. The new algorithm works as follows. Use the $\{\frac{1}{2}, 1\}$-integral solution $(\hat{x}, \hat{y})$ to generate a forest as was done in the previous rounding algorithm. Use this forest to define a forest metric on the nodes in N', and then compute the optimal k-median solution with respect to this modified metric.

We shall argue next that the modified algorithm finds an integer solution of cost at most ρ times the cost of the $\{\frac{1}{2}, 1\}$-integral solution (both with respect to the original metric). Let z be the cost of the $\{\frac{1}{2}, 1\}$-integral solution in the original metric. For each pair of nodes (i, j), the cost with respect to the forest metric is at least the original cost, and for each edge in the forest, the two costs are the same. Since the $\{\frac{1}{2}, 1\}$-integral solution $(\hat{x}, \hat{y})$ only has (positive) assignments corresponding to edges, the cost of this solution with respect to the forest metric is also z. Hence there is an integer solution of cost of at most ρz with respect to the forest metric. Furthermore, the cost of the optimal solution found by the forest metric algorithm is at most ρz, and so the cost with respect to the original metric of this solution is also at most ρz.

We have already shown that $\rho \leq 2$. In Section 7, we will show that any $\{\frac{1}{2}, 1\}$-integral solution can be converted to an integer solution of cost at most 4/3 times as much. At this point we will make a slight detour and review the algorithm, and then return to the proof of the upper bound of the integrality gap of $\{\frac{1}{2}, 1\}$-integral solutions.

6 The algorithm in retrospect

Reflecting on the algorithm and the analysis we presented, it turns out that the underlying algorithm is much simpler than it might appear from our description. The entire algorithm can be viewed as building a collection of trees from the LP solution and then solving the problem optimally on this collection. Here is the complete algorithm stated slightly differently.

Step 1$'$: Solve the LP and perform (a modified version of) Step 1 as in Section 3. However, instead of modifying the demands, we build a collection of stars on the set of locations N. We have one star in the collection for every location $j \in N'$; this star consists of edges from j to each location i such that the demand of i was moved to j in the original Step 1.

Step 2$'$: Obtain a collection of trees by connecting each vertex $j \in N'$ to its nearest neighbor $s(j) \in N'$ (where ties and cycles of length 2 are broken as before).

Step 3$'$: Solve the modified instance consisting of the original demands where the underlying metric is the forest metric (as defined in the previous section) induced by the collection of trees produced in Step 2.

The analysis of the algorithm presented in Steps 1-3 also proves that the performance guarantee of Steps 1$'$-3$'$ is at most $6\frac{2}{3}$ (modulo the proof still promised for rounding the $\{\frac{1}{2}, 1\}$-integral solution).

7 Bounding the integrality gap of a $\{\frac{1}{2}, 1\}$-integral solution

We will show that for any metric, there exists an integral solution of cost at most $\frac{4}{3}$ times the cost of an optimal $\{\frac{1}{2}, 1\}$-integral solution. That is, we shall show that the integrality gap is bounded by $\frac{4}{3}$. It is interesting to note that, in constrast to the discussion above, we shall not need to use the fact that the metric is a forest metric. This bound is tight, as is shown by the example of a tree with unit-length edges, no demand at the root, but demand 1 at each of its 3 children, and $k = 2$. The optimal integer solution is of cost 2, whereas the optimal $\{\frac{1}{2}, 1\}$-integral solution is of cost 3/2.

The basic idea of the upper bound on the integrality gap is to produce a probability distribution on integral solutions of expected cost at most $\frac{4}{3}$ times the cost of the $\{\frac{1}{2}, 1\}$-integral solution. Since there must be an integer solution of cost no greater than this expectation, this proves the claimed bound.

7.1 The Outline of the Proof

We start by presenting the framework used to bound the integrality gap. As mentioned above, we will construct a probability distribution over integral solutions. We will decompose the tree into small neighborhoods (Step A) and create distributions over these smaller trees (Step B). We first state some properties ensured by these two steps and show how the expected cost of an integral solution can be bounded using these properties. Later, we will describe the individual steps in detail and prove that the stated properties are satisfied.

Step A: Creating 3-level trees.

We are given a set of rooted trees Γ corresponding to a $\{\frac{1}{2}, 1\}$-integral solution $(\hat{x}, \hat{y})$. Recall that the nearest neighbor of i is $s(i)$; if $\hat{y}_i = \frac{1}{2}$, then $s(i)$ is the parent of i in Γ, unless i is a root of a tree in Γ (and then $s(i)$ is its child). We will construct a set of trees Γ_3 that contains all nodes in Γ such that

- (P0) Each tree has at most three levels. We say that the level of a root of a tree is 0, and refer to its children as level-1 nodes, and their children as level-2 nodes.
- (P1) A node i is a parent of j in a tree in Γ_3 only if $i = s(j)$. Thus, each tree in Γ_3 is a subgraph of the graph defined by trees in Γ.
- (P2) Each node i with $\hat{y}_i = \frac{1}{2}$ belongs to a tree of size 2 or more in Γ_3.
- (P3) If i is a root of a tree in Γ_3 with $\hat{y}_i = \frac{1}{2}$, then the distance from node i to its nearest child is at most $2c_{is(i)}$, or $s(i)$ is a level-1 node in some other tree in Γ_3.
- (P4) For each node i in a tree in Γ_3 that is not level-0 or level-1, we have that $c_{is(i)} \geq 2c_{s(i)s(s(i))}$.

Step B: Distribution over 3-level trees.

Given a set of 3-level trees we construct a probability distribution over integral solutions. We will choose k nodes such that the following conditions are met:

- (D0) If $\hat{y}_i = 1$, then node i is chosen.
- (D1) If a root is not chosen, then all level-1 nodes in that tree are chosen.
- (D2) Each root is chosen with probability at least $\frac{2}{3}$.
- (D3) Each level-1 node is chosen with probability at least $\frac{1}{3}$.
- (D4) Each level-2 node is chosen with probability at least $\frac{1}{2}$.

(D5) For each level-2 node i, conditioned on the fact that i is not chosen, its parent $s(i)$ is chosen with probability with at least $\frac{1}{3}$.

Notice that if the choice of i and $s(i)$ were independent, then the last condition would follow from (D3). Using the above properties, we can prove the following lemma.

Lemma 9 *If properties (P0)-(P4) and (D0)-(D5) are true, then for each node j with $\hat{y}_j = \frac{1}{2}$, the expected distance to its nearest center is at most $\frac{2}{3}c_{js(j)}$.*

Proof: Suppose that j is a root node. By property (D2), j is chosen with probability at least $\frac{2}{3}$. By property (D1), if j is not chosen, all of its children are chosen. From property (P2), it has at least one child. If the distance from node j to its nearest child is at most $2c_{js(j)}$, then by property (D1), node j has an open center at most $2c_{js(j)}$ away from it. Otherwise, (if its nearest child were farther away) by property (P3), we have that $s(j)$ is a level-1 node in some other tree. By another application of property (D1), either $s(j)$ or $s(s(j))$ is chosen. In either case, node j is at most $2c_{js(j)}$ away from a center. Therefore, the expected distance from node j to its nearest center is at most $\frac{2}{3}c_{js(j)}$.

Suppose that node j is a level-1 node. By property ($D3$), node j is chosen with probability at least $\frac{1}{3}$. If j is not chosen, by property ($D1$), its parent $s(j)$ (the root) must be chosen. Thus, the expected distance to its nearest center is at most $\frac{2}{3}c_{js(j)}$.

The remaining case is when j is a level-2 node. Then, by property (D4), node j is chosen with probability at least $\frac{1}{2}$. By property (D5), if we condition on node j not being chosen, then $s(j)$ is chosen with probability at least $\frac{1}{3}$. If $s(j)$ is not chosen, then $s(s(j))$ must be chosen (by property (D1)). Thus, the expected distance from node j to its nearest center is at most

$$\frac{1}{2}\left(\frac{1}{3}c_{js(j)} + \frac{2}{3}(c_{js(j)} + c_{s(j)s(s(j))})\right) \leq \frac{1}{2}c_{js(j)} + \frac{1}{3}c_{s(j)s(s(j))}.$$

By property (P4), the right-hand side is at most $\frac{2}{3}c_{js(j)}$. □

The following theorem is an immediate consequence of Lemma 9.

Theorem 10 *For any metric, the ratio between the cost of a given $\{\frac{1}{2}, 1\}$-integral solution and the optimal integral solution is at most $\frac{4}{3}$.*

Proof: We compare the cost of the given $\{\frac{1}{2}, 1\}$-integral solution to the expected cost of the integral solution found above. From (D0), each node with $\hat{y}_j = 1$ is chosen as a center in the integral solution. Thus, the assignment cost for each such node j is equal to 0, both for the integral solution found, and for the given $\{\frac{1}{2}, 1\}$-integral solution. For a node j with $\hat{y}_j = \frac{1}{2}$, its contribution to the objective function value of the $\{\frac{1}{2}, 1\}$-integral solution is $\frac{1}{2}d'_j c_{js(j)}$. By Lemma 9, the expected contribution of node j to the objective function value for the integer solution is at most $\frac{2}{3}d'_j c_{js(j)}$. Therefore, the total expected cost of the integer solution is at most $\frac{4}{3}$ times the cost of the $\{\frac{1}{2}, 1\}$-integral solution. □

We now present the details of Steps A and B.

7.2 Step A: Creating 3-level trees

Let Γ denote the set of trees corresponding to a $\{\frac{1}{2}, 1\}$-integral solution $(\hat{x}, \hat{y})$. We will create the 3-level trees in a two phase process. We will create trees that are subgraphs of trees belonging to Γ, and property (P1) will follow immediately.

Phase 1: In the first phase, we will convert a tree $T \in \Gamma$ to a sub-collection of trees that satisfy properties (P4) and (P2). They need not have 3 or fewer levels. For each tree $T \in \Gamma$, we will visit its nodes by visiting all descendants of a node before visiting the node itself (i.e., post-order). Furthermore, we require that we visit the nearest child of the root of T as the last node visited, before visiting the root of T. This latter condition need not be maintained for visiting other nodes in T.

Throughout this construction, we shall be modifying the tree T. Assume we are visiting a node i that is neither the root, nor the nearest child of the root. Let T_i denote the subtree of the current tree T that is rooted at i. We will prune the tree T by removing the subtree T_i rooted at node i; the tree T_i is added to the collection Γ', if both of the following conditions are satisfied:

(a) the node i has a child,

(b) the distance from i to its nearest child j is less than twice the distance to its parent $s(i)$, i.e., $c_{ij} < 2c_{is(i)}$.

Suppose that we are visiting the nearest child i of a root node j. We check to see if either of the following cases apply:

(i) $\hat{y}_j = \frac{1}{2}$ and i is the only remaining child of j;

(ii) $\hat{y}_j = \frac{1}{2}$ and $c_{i'j} \geq 2c_{ij}$, where i' is the second nearest remaining child of j.

If neither (i) nor (ii) applies, we prune T_i if both conditions (a) and (b) are satisfied. If either (i) or (ii) holds, we modify the tree T by making j the child of i. (Note that any remaining children of the root j, other than i, continue to be children of j.) The modified tree T is added to Γ'.

If we do visit the root, then we add the remaining tree T to Γ', and phase 1 ends. Before proceeding with phase 2 to finish the construction, we prove the following lemma:

Lemma 11 *At the end of phase 1, properties (P1)-(P4) hold, and for each node i' in a tree in Γ' that is at level 2 or more, $c_{i's(i')} > 2c_{s(i')s(s(i'))}$.*

Proof: It is easiest to verify that property (P2) is true. All trees added to Γ' while processing nodes other than the root have size at least 2. The only potential singleton tree is one consisting of the original root itself. Suppose that the root j has $\hat{y}_j = \frac{1}{2}$. If the root has no remaining child other than its nearest child, then the root would be removed when its nearest child was processed and will not create a singleton tree.

Property (P1) is also easy to verify. We do not change the parent-child ordering in the pruning process, except possibly when we visited i, the nearest child of a root j with $\hat{y}_j = \frac{1}{2}$. In this case, node j became a child of i, and indeed $s(j) = i$.

Condition (b) ensures that property (P3) is true for each pruned tree T_i that is added to Γ' while visiting a node i that is not the root node. If the root j becomes the child of i, where i was the nearest child of j originally, then property (P3) holds, since the nearest child of the (new) root i is j, and it is at a distance $c_{is(i)}$. Therefore, the only case not yet covered is when we visit the original root j of the tree and add the remaining tree T to Γ'. If $\hat{y}_j = 1$, then property (P3) is vacuously true. If $\hat{y}_j = \frac{1}{2}$, note that $s(j)$ is node j's nearest child i. Property (P3) is true if the nearest child i is still remaining in T when j is visited. If the nearest child i was pruned away, then it must be the case that both conditions (i) and (ii) did not hold when i was visited. Thus, j must have a child i' remaining in T other than its nearest child i, and $c_{i'j} < 2c_{ij} = 2c_{js(j)}$. Hence, property (P3) holds for the tree rooted at j that is added to Γ'.

To see that property (P4) holds, consider a node i' which is at level $\ell \geq 2$ in some tree in Γ'. Suppose, for a contradiction, that (P4) does not hold, i.e., $c_{i's(i')} < 2c_{s(i')s(s(i'))}$. Let the parent of i' be i, and let us consider why the subtree rooted at i was not pruned and added to Γ' when i was visited. Notice that both conditions (a) and (b) are satisfied. It must be the case that i was the nearest child of the original root j or the original root. If i was the nearest child of j, then a subtree rooted at i must have been added to Γ' contradicting that i' is at level 2 or more. (If either (i) or (ii) holds, such a subtree is added with the root j being made a child of i. If neither holds, the subtree rooted at i is pruned away, since both (a) and (b) hold.) Consider the case when the parent i of i' is itself the original root in Γ. If i' was the nearest child of the root i, then i' is always either a level-0 or a level-1 node in some tree in Γ'. Suppose i' was not the nearest child of the root i. If we added the tree rooted at i to Γ', we would have i' as a level-1 node in some tree in Γ'. Finally, consider the case when the the root i is added to the subtree rooted at its nearest child. In this case, when the nearest child was visited, (i) did not apply since i had another child i'. Hence, (ii) must have applied and in this case $c_{i's(i')} \geq 2c_{s(i')s(s(i'))}$. (Note that $s(s(i')) = s(i)$ is the nearest child of root i in this case.) This proves that property (P4) holds. $\square$

Phase 2: This phase ensures that (P0) is satisfied, and that the properties (P1)-(P4) remain true. We perform the following steps. Initially, Γ_3 is empty.

(a) For each tree $T' \in \Gamma'$, remove all the edges between the levels $2p+1$ and $2p+2$ for all $p \geq 0$. (Edges between levels $2p$ and $2p+1$ are undisturbed.)

(b) The previous step yields a decomposition into trees having at most two levels (i.e., stars). For the singleton nodes created in this process, (i.e., nodes that are in level $2p+2$ and have no children), attach them to their parent in T'. We add the set of trees created to Γ_3.

It is immediate that property (P0) is true. Property (P4) is true, since we do not create any new nodes that are not level-0 or level-1. Property (P2) is true, since we do not create singleton trees in the second phase. Property (P1) is true, since we do not create any new parent-child relationships. Property (P3) is true, since the parent of each new root created is a level-1 node. Therefore, all of the properties (P0)–(P4) are true for the collection of trees, Γ_3.

7.3 Step B: Distribution over 3-level trees

In this section, we specify a probability distribution of integer solutions, given the 3-level trees, and show that the distribution satisfies the desired properties.

We start by introducing some additional definitions. A tree $T \in \Gamma_3$ is said to be an *even* tree if $\sum_{j\in T} \hat{y}_j$ is integral; otherwise, it is said to be an *odd* tree. Clearly the number of odd trees will be even. An even tree T with $\sum_{j\in T} \hat{y}_j = p$ will be assigned p centers. For each odd tree, we have that $\sum_{j\in T} \hat{y}_j = p + \frac{1}{2}$, for some $p \geq 1$: notice that this follows from property (P2). For half of the odd trees, we will round up this fractional value to obtain the number of centers opened, whereas in the other half, we will round this fractional value down. If we round up the number of centers (from $p+\frac{1}{2}$ to $p+1$), then we will say that the tree is a *1-tree*. Otherwise, if we round down (from $p+\frac{1}{2}$ to p), then we will say that the tree is a *0-tree*.

We will choose half of the odd trees to be 1-trees uniformly at random. The other odd trees will be 0-trees. It is immediate that we will open exactly k centers in total.

There are two cases, depending on whether the root i of a tree in Γ_3 has $\hat{y}_i$ equal to 1 or $\frac{1}{2}$. We present the former, and simpler, case first. Throughout this discussion, by saying that we *uniformly choose* p elements from a given set, we will mean that we choose a subset of p elements uniformly at random from amongst all subsets of size p. The selections for different trees are made independently of each other.

Case 1: Suppose that the root i of a tree has $\hat{y}_i = 1$. The root is picked as a center. The remaining centers are picked from the following distribution:

(i) If the root has $2p$ descendants (level-1 and level-2 nodes), uniformly choose p of them to build a center.

(ii) If the root has $2p+1$ descendants, uniformly choose $p+1$ of them if the tree is a 1-tree, and otherwise, uniformly choose p of them.

Before we proceed to Case 2, we can easily show that

Lemma 12 *If root j of a tree in Γ_3 has $\hat{y}_j = 1$, then the above distribution satisfies (D0)-(D5).*

Proof: Since the root is chosen (D0)-(D2) is satisfied immediately. The probability of choosing any other node is exactly $\frac{1}{2}$ in case (i). The probability of choosing a node in case (ii) is also $\frac{1}{2}$. With probability $\frac{1}{2}$ the tree is a 1-tree in which case the probability of a node being chosen is $(p+1)/(2p+1)$. Otherwise with probability $\frac{1}{2}$ the tree is a 0-tree with the probability that a node is chosen is $p/(2p+1)$. Thus the probability that a node is chosen is $\frac{1}{2}$. This proves (D3) and (D4).

The proof of (D5) is straightforward. If we condition on the event that a level-2 node i is not chosen, the probability of choosing its parent $s(i)$ can only increase. Since $s(i)$ was chosen with probability $\frac{1}{2}$ overall, the conditional probability of choosing $s(i)$ is still at least $\frac{1}{2}$. $\square$

Case 2. We are now ready to specify our distribution for any tree in Γ_3, rooted at a node j for which $\hat{y}_j = 1/2$. We will further subdivide this case into two subcases: first, where the number of level-1 nodes is odd or the number of level-2 nodes is even, and the second, where both these conditions do not hold.

Case 2.a: The number of level-1 nodes is odd or the number of level-2 nodes is even. The distribution will be specified by giving two independent distributions: there will be one distribution specifying the selection of level-0 (i.e., the level-1 nodes and another distribution that specifies the selection of the level-2 nodes. We will describe the distribution over level-2 nodes first.

(i) If there are $2r$ level-2 nodes, uniformly choose r to open as centers. If there are $2r+1$ nodes in level 2, (and so the number of level-1 nodes is odd, and this is an odd tree), uniformly choose $r+1$ of them if this is a 1-tree, and uniformly choose r of them if it is 0-tree.

(ii) If there is a single level-1 node, then open a center at the root with probability $\frac{2}{3}$ and at the single level-1 node with probability $\frac{1}{3}$.

(iii) If the number of level-1 nodes is $2t+1$ ($t \geq 1$), open a center at the root and uniformly choose t level-1 vertices as centers.

(iv) If the number of level-1 nodes is 2 (and so the number of level-2 nodes is even and this is an odd tree), then, if the tree is a 1-tree, build a center at both level-1 nodes with probability $\frac{2}{3}$, and build a center at the root and at one of the level-1 nodes (arbitrarily selected), with probability $\frac{1}{3}$. If the tree is a 0-tree, build a center at the root.

(v) If the number of level-1 nodes is $2t$ ($t > 1$), open a center at the root and uniformly choose t level-1 nodes if the tree is a 1-tree. Otherwise, open a center at the root and uniformly choose $t-1$ level-1 nodes as centers.

Lemma 13 *For any tree in Γ_3, rooted at a node j for which $\hat{y}_j = 1/2$ where the number of level-1 nodes is odd or the number of level-2 nodes is even, the above distribution satisfies the properties (D0)-(D5).*

Proof: Property (D0) is true vacuously. Property (D1) is true by construction. Property (D2) is true since the only cases in which the root is not chosen are (ii) and (iv) (only when the tree is chosen as a 1-tree) and in both cases, the root is chosen with probability $\frac{2}{3}$ (taking into account in (iv) that it is a 0-tree with probability $\frac{1}{2}$).

In case (ii), property (D3) follows directly from the construction. In case (iii), it holds since $t/(2t+1) \geq \frac{1}{3}$, for $t \geq 1$. In case (iv), the tree is chosen to be a 1-tree with probability $\frac{1}{2}$, and in that case a level-1 node is chosen with probability $\frac{2}{3}$; thus, the probability of selecting a level-1 node is at least $\frac{1}{3}$, even ignoring the other case. In case (v), the probability of choosing a level-1 node is $\frac{1}{2}$ if the tree is a 1-tree and is $(t-1)/(2t)$ if the tree is a 0-tree; once again, since the probability that the tree is a 1-tree is exactly $\frac{1}{2}$, the probability of choosing a level-1 node is at least $\frac{1}{4} + (t-1)/(4t) \geq \frac{3}{8} > \frac{1}{3}$.

Property (D4) holds by the same argument as in Lemma 12: That is, if the number of level-2 nodes is even, we are done. If it is odd, the tree is odd as was pointed out in the construction. In that case, with probability $\frac{1}{2}$, the tree is a 1-tree and we choose each level-2 node with probability $(r+1)/(2r+1)$, and with probability $\frac{1}{2}$, the tree is a 0-tree and we choose each level-2 node with probability $r/(2r+1)$. Thus, the overall probability of choosing a level-2 node is $\frac{1}{2}$.

Property (D5) holds, since for any level-2 node i, the probability distributions governing the choice of i and $s(i)$ are independent, and we have already proved that (D3) holds. □

Case 2.b: If there are an even number of level-1 nodes and an odd number of level-2 nodes, we consider a different distribution:

(i) If there are 2 level-1 nodes and $2r+1$ level-2 nodes ($r \geq 0$), then, with probability $\frac{2}{3}$, open centers at the root and $r+1$ uniformly chosen level-2 nodes; and, with probability $\frac{1}{3}$, open centers at the 2 level-1 nodes and r uniformly chosen level-2 nodes.

(ii) If there are $2t$ level-1 nodes ($t > 1$) and $2r+1$ level-2 nodes ($r \geq 0$), always open a center at the root. With probability $\frac{1}{2}$, uniformly choose $t-1$ level-1 nodes and independently, uniformly choose $r+1$ level-2 nodes. With probability $\frac{1}{2}$, uniformly choose t level-1 nodes and independently, uniformly choose r level-2 nodes.

Lemma 14 *The above distribution satisfies the properties (D0)-(D5).*

Proof: Property (D0) is true vacuously. Property (D1) follows directly from the construction. To see that property (D2) is also true, note that the only case in which the root is not chosen is (i), and in that case, the root is chosen with probability $\frac{2}{3}$.

It is easy to check that property (D3) is true in case (i). In case (ii), a level-1 node is chosen with overall probability
$$\frac{1}{2} \cdot \frac{t-1}{2t} + \frac{1}{2} \cdot \frac{t}{2t} = \frac{2t-1}{4t}.$$
Since $t > 1$, it follows that property (D3) holds in this case as well.

In case (ii), property (D4) is true, since the overall probability of choosing a level-2 node is

$$\frac{1}{2} \cdot \frac{r+1}{2r+1} + \frac{1}{2} \cdot \frac{r}{2r+1} = \frac{1}{2}.$$

In case (i), the probability of choosing a level-2 node is

$$\frac{2}{3} \cdot \frac{r+1}{2r+1} + \frac{1}{3} \cdot \frac{r}{2r+1} = \frac{3r+2}{6r+3} > \frac{1}{2}.$$

To verify that (D5) is true, for a level-2 node i, we will compute the probability that $s(i)$ is chosen given that i is not chosen. This can be done using the fact that $Prob[A|B] = Prob[A \text{ and } B]/Prob[B]$. In case (i), the probability is:

$$\frac{1 \cdot (r+1)}{2 \cdot r + 1 \cdot (r+1)} \geq \frac{1}{3}.$$

In case (ii), the probability is

$$\frac{t-1}{2t} \cdot \frac{r}{2r+1} + \frac{t}{2t} \cdot \frac{r+1}{2r+1} \geq \frac{3r/4+1/2}{2r+1} \geq \frac{1}{3}.$$

□

Thus, we have created a distribution satisfying properties (D0)-(D5) on a collection of 3-level trees Γ_3 that satisfies the properties (P0)-(P4).

Acknowledgments This work is a combined version of two earlier papers, by Charikar & Guha, and by Tardos & Shmoys, which independently obtained nearly identical results. We would like to thank David Williamson for helping to facilitate the merger, and Chandra Chekuri, Fabian Chudak, Ashish Goel, and Rajeev Motwani for several useful discussions at the starting point of this research. Part of this work was done while the first two authors were visiting IBM T. J. Watson Research Center in Yorktown Heights.

References

[1] S. Arora, P. Raghavan, and S. Rao. Approximation schemes for Euclidean k-medians and related problems. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing*, pages 106–113, 1998.

[2] V. Arya, N. Garg, R. Khandekar, A. Meyerson, K. Mungala, and V. Pandit. Local search heuristics for k-median and facility location problems. In *Proceedings of the 33rd Annual ACM Symposium on the Theory of Computing*, pages 21–29, 2001.

[3] J. Bar-Ilan, G. Kortsarz, and D. Peleg. How to allocate network centers. *J. Algorithms*, 15:385–415, 1993.

[4] Y. Bartal. Probabilistic approximation of metric spaces and its algorithmic applications. In *Proceedings of the 37th Annual IEEE Symposium on Foundations of Computer Science*, pages 184–193, 1996.

[5] Y. Bartal. On approximating arbitrary metrics by tree metrics. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing*, pages 161–168, 1998.

[6] M. Charikar. *Algorithms for Clustering Problems.* Ph.D. Dissertation, Department of Computer Science, Stanford University, 2000.

[7] M. Charikar, C. Chekuri, A. Goel, and S. Guha. Rounding via trees: deterministic approximation algorithms for group Steiner trees and k-median. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing*, pages 114–123, 1998.

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

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

[10] F. Chudak and D. Shmoys. Improved approximation algorithms for the uncapacitated facility location problem. Unpublished manuscript, 1998.

[11] F. A. Chudak and D. B. Shmoys. Improved approximation algorithms for the capacitated facility location problem. In *Proceedings of the 10th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages S875–S876, 1999.

[12] G. Cornuéjols, G. L. Nemhauser, and L. A. Wolsey. The uncapacitated facility location problem. In P. Mirchandani and R. Francis, editors, *Discrete Location Theory*, pages 119–171. John Wiley and Sons, Inc., New York, 1990.

[13] M. E. Dyer and A. M. Frieze. A simple heuristic for the p-center problem. *Oper. Res. Lett.*, 3:285–288, 1985.

[14] U. Feige. Personal Communication, August 1997.

[15] T. F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoret. Comput. Sci.*, 38:293–306, 1985.

[16] S. Guha. *Approximation algorithms for facility location problems.* Ph.D. Dissertation, Department of Computer Science, Stanford University, 2000.

[17] S. Guha and S. Khuller. Greedy strikes back: Improved facility location algorithms. In *Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 649–657, 1998.

[18] D. S. Hochbaum and D. B. Shmoys. A best possible approximation algorithm for the k-center problem. *Math. Oper. Res.*, 10:180–184, 1985.

[19] K. Jain, M. Mahdian, and A. Saberi. A new greedy approach for facililty location problems. Proceedings of the 34th Annual ACM Symposium on the Theory of Computing, pages 731–740, 2002.

[20] 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*, pages 2–13, 1999.

[21] O. Kariv and S. L. Hakimi. An algorithmic approach to network location problems, Part II: p-medians. *SIAM J. Appl. Math.*, 37:539–560, 1979.

[22] S. Khuller and Y. J. Sussmann. The capacitated k-center problem. In *Proceedings of the 4th Annual European Symposium on Algorithms, Lecture Notes in Computer Science 1136*, pages 152–166, Berlin, 1996. Springer.

[23] 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*, pages 1–10, 1998.

[24] J.-H. Lin and J. S. Vitter. Approximation algorithms for geometric median problems. *Inform. Proc. Lett.*, 44:245–249, 1992.

[25] J.-H. Lin and J. S. Vitter. ϵ-approximations with minimum packing constraint violation. In *Proceedings of the 24th Annual ACM Symposium on Theory of Computing*, pages 771–782, 1992.

[26] R. R. Mettu and C. G. Plaxton. The online median problem. In *Proceedings of the 41st Annual IEEE Symposium on Foundations of Computer Science*, 2000, 339–348.

[27] D. B. Shmoys and E. Tardos. An Approximation Algorithm for the Generalized Assignment Problem. *Math. Programming*, 62:461–474, 1993.

[28] D. B. Shmoys, É. Tardos, and K. I. Aardal. Approximation algorithms for facility location problems. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, pages 265–274, 1997.

[29] M. Sviridenko. Personal communication, July, 1998.

[30] A. Tamir. An $O(pn^2)$ algorithm for the p-median and related problems on tree graphs. *Oper. Res. Lett.*, 19:59–94, 1996.

[31] S. de Vries, M. Posner, and R. Vohra. The K-median Problem on a Tree. *Working paper*, Ohio State University, Oct. 1998.

[32] J. Ward, R. T. Wong, P. Lemke, and A. Oudjit. Properties of the tree K-median linear programming relaxation. Unpublished manuscript, 1994.