Approximation Algorithms for the k-Median Problem

Roberto Solis-Oba

Department of Computer Science,
The University of Western Ontario,
London, Ontario, Canada
solis@csd.uwo.ca

Abstract. The k-median problem is a central problem in Operations Research that has captured the attention of the Algorithms community in recent years. Despite its importance, a non-trivial approximation algorithm for the problem eluded researchers for a long time. Remarkably, a succession of papers with ever improved performance ratios have been written in the last couple of years. We review some of the approaches that have been used to design approximation algorithms for this problem, and we also present some of the known results about the hardness of approximating the optimum solution for the k-median problem.

1 Introduction

The problem of locating a set of facilities so that they can efficiently serve a group of clients has been extensively studied because of its large number of applications and interesting algorithmic aspects. Facility location problems find applications in areas as diverse as Operations Research [18,29], network design [3,19,20], data mining [9], clustering analysis [40], and web access [32,22,44], among others. These problems have been studied for over four decades [29], and many different approaches have been proposed for solving them [5,6,12,14,18,33,42].

Given sets of facilities F and clients D, the cost of servicing a client i by a facility j, denoted as $c(i,j)$, expresses the effort required to serve i from j. So, the service cost might represent the distance from a client to a facility, or the cost of shipping some commodity produced at a facility site to the client site. The most studied facility location problems are the k-median problem and the facility location problem.

In the k-median problem the goal is to select or open a set of at most k facilities (called *centers* or *medians*) that serve the clients D at minimum total cost. Given a set of centers S, a client i is always served by the center $j \in S$ for which the service cost $c(i,j)$ is minimum. The facility location problem assigns a cost $f(j)$ for opening a facility $j \in F$. The problem is to choose a set of facilities and assign clients to facilities so that the total service cost plus the total cost of the facilities is minimized. Numerous variants of these problems have been proposed in the literature, like the capacitated facility location [15], the fault tolerant k-center [21,28], bounded facility location [31], capacitated facility location with client demands [15], k-capacitated facility location [23,5], universal facility location [37], and multilevel facility location [10,49].

Let n denote the number of clients plus the number of facilities. The k-median problem is NP-hard even if the sets of facilities and clients are points on the plane and the service cost is the Euclidean distance from a client to the nearest center [26,36,43]. Moreover, the problem cannot be approximated within any factor $\alpha > 1$ of the optimum unless P=NP, even if the cost matrix is symmetric [33]. For any given value $\alpha > 1$ no polynomial time algorithm that chooses up to $o(\log n)k$ centers can approximate the value of the optimum solution for the k-median problem within a factor α of the optimum, unless P=NP [33].

Due to the difficulty of approximating the solution of the general k-median problem, many constrained versions have been considered. The most studied version of the problem is the *metric* k-median problem. In this version of the problem we interpret the cost function c as measuring the distance between a given client i and facility j. Furthermore, the cost function is extended so that it not only gives the distance between a client and a facility, but it also gives the distance between two clients or between two facilities. In the metric k-median problem, the cost function is assumed to be symmetric, i.e. $c(i,j) = c(j,i)$, and it satisfies the triangle inequality, i.e., $c(i,k) \leq c(i,j)+c(j,k)$ for any $i,j,k, \in D \cup F$. The metric facility location problem is defined analogously, and the best known algorithm for it achieves a performance ratio of 1.52 [38].

Despite the fact that the k-median problem is a central problem in Operations Research, a non-trivial approximation algorithm for it eluded researchers for a long time. The problem restricted to trees, though, has been known to be polynomially solvable for some time [25,47]. It has been fascinating to see how in the span of a few years successively better algorithms have been designed for the problem. The first approximation algorithm for the problem was designed by Lin and Vitter [33] in 1992. They showed that a $(1+\varepsilon)$-approximation to the value of the optimum solution for the k-median problem can be computed in polynomial time if it is allowed to select up to $(1+\frac{1}{\varepsilon})(\ln n + 1)k$ centers, for any value $\varepsilon > 0$. This result is best possible, up to constant factors, unless P=NP. For the metric k-median problem, Lin and Vitter also designed an algorithm that finds a solution of value at most $2(1+\varepsilon)$ times the optimum while selecting at most $(1+\frac{1}{\varepsilon})k$ centers.

In 1998, Korupolu, Plaxton, and Rajaraman [30] designed a simple local search algorithm that, for any value $\varepsilon > 0$, computes a $(1+\varepsilon)$-approximation to the value of the optimum solution for the metric k-median problem that selects no more than $(3+5/\varepsilon)k$ centers. This algorithm can be modified so that it uses only $(3+\varepsilon)k$ centers, but by selecting fewer centers, the value of the solution that it finds can be up to $1+5/\varepsilon$ times the optimum.

The first approximation algorithm for the metric k-median problem that produced a solution with at most k centers was derived in 1996 from the powerful result of Bartal [7,8], that shows how to approximate any finite metric space by a tree space. Combining this result with known algorithms for the k-median problem on trees, Bartal gave a randomized $O(\log n \log\log n)$-approximation algorithm for the metric k-median problem. Shortly after, Charikar, Chekuri, Goel, and Guha [11] showed how to de-randomize Bartal's algorithm, and they also slightly improved it obtaining an $O(\log k \log\log k)$-approximation algorithm.

The next breakthrough came a few months later, in 1998, when Arora, Raghavan, and Rao [4] presented a polynomial time approximation scheme for the metric k-median problem restricted to two-dimensional Euclidean spaces. This means that the clients and facilities are points on the plane and the cost $c(i, j)$ is the Euclidean distance between i and j. A polynomial time approximation scheme is a family of algorithms $\mathcal{A}$ such that for any value $\varepsilon > 0$ an algorithm $A_\varepsilon \in \mathcal{A}$ finds in polynomial time a solution of value at most $(1 + \varepsilon)$ times the optimum.

In 1999, Charikar and Guha, and independently, Tardos and Shmoys, designed the first constant factor approximation algorithm for the metric k-median problem. They combined their results and presented a linear programming based algorithm that finds a solution of value at most $6\frac{2}{3}$ times larger than the optimum. The same year Jain and Vazirani [23] improved this result by designing a primal-dual 6-approximation algorithm. Shortly after, Charikar and Guha [12] refined Jain and Vazirani's approach to obtain a 4-approximation algorithm for the problem.

The latest improvement, to the date when this paper was written, came in 2001, when Arya, Garg, Khandekar, Meyerson, Munagala, and Pandit [5] proposed a very simple local search algorithm that finds a solution of value $(3 + \frac{2}{p} + O(\varepsilon'))$ times larger than the optimum, for any value $0 < \varepsilon' = o(n^{-2})$.. The parameter $p \geq 1$ is an integer value that guides the local search procedure, and the time complexity of the algorithm depends exponentially on it.

Another work that deserves mention is the algorithm of Mettu and Plaxton [39] for a generalization of the k-median problem known as the *online metric median problem*. In this problem the centers are selected one at a time and a facility cannot be de-selected once it has been chosen. Furthermore, the number of facilities to be selected, k, is not known in advance. The algorithm of Mettu and Plaxton finds a solution of value at most a constant factor times the value of an optimum for the k-median problem (also known as the *offline* k-median problem), for any value of k.

In this paper we describe in some detail some of the aforementioned works. In Section 2 we describe some results related to the hardness of approximating the k-median problem. Then, in Section 3, we present two linear programming based algorithms. The first one is the algorithm of Lin and Vitter [33] which approximates the solution of the general k-median problem by using up to $O(\log n)k$ centers. The second algorithm, and the rest of the algorithms that we present here, is for the metric version of the problem. This algorithm by Charikar, Guha, Tardos, and Shmoys [13] is the first algorithm which achieved constant performance ratio (this means that the algorithm approximates the optimum solution within a constant factor). In Section 4 we describe a very interesting primal dual algorithm by Jain and Vazirani [23], which exploits the fact that a Lagrangian relaxation of the k-median problem yields a special instance of the uncapacitated facility location problem. Finally, in Section 5 we present a remarkable algorithm by Arya, Garg, Khandekar, Meyerson, Munagala, and Pandit [5], which uses local search heuristics to yield the best known performance ratio for the problem.

2 Hardness of the k-Median Problem

Any instance of the k-median problem can be modeled with an undirected graph $G = (V, E)$, where each vertex corresponds to either a client or a facility, and every edge (i, j) is labelled with the corresponding cost $c(i, j)$. The number of vertices in V is n and the number of edges in E is m. Using this representation, we can give a simple proof that the k-median problem is NP-hard. Consider an instance $G = (V, E), k$ of the problem in which $V = F = D$, i.e. every vertex is both a client and a facility. Such an instance has a solution of value $n - k$ if and only if G has a dominating set of size k. Since the dominating set problem is NP-hard, the claim follows.

Interestingly, another simple reduction also shows that for any computable function $\alpha(n) > 1$, the optimum solution for the k-median problem cannot be approximated within any factor $\alpha(n)$ of the optimum in polynomial time, unless P=NP.

Theorem 1. *For any computable function $\alpha(n) > 1$, the optimum solution for the k-median problem cannot be approximated within factor $\alpha(n)$ in polynomial time, unless P=NP, even if the cost function is symmetric.*

Proof. Consider an instance $G = (V, E), k$ of the k-median problem in which $V = F = D$. Build a complete graph $G' = (V, E')$ with edge weights defined as follows:

– $c(i, i) = 0$ for all $i \in V$,
– $c(i, j) = 1$ if $(i, j) \in E$, and
– $c(i, j) = \alpha(n)(n - k) + 1$, otherwise.

Assume that there is an $\alpha(n)$-approximation algorithm A for the k-median problem. If G has a dominating set of size k, then the corresponding instance of the k-median problem has a solution of value $n - k$. Note that in this case, algorithm A would choose a set S of k centers with total service cost at most $\alpha(n)(n - k)$. Since every edge $(i, j) \in E' \setminus E$ has cost $\alpha(n)(n - k) + 1$, set S must be a dominating set of size k for G.

Interestingly, the problem is also hard to approximate if we allow the selection of more than k centers. Let S^* be an optimum solution for the k-median problem and let $c(S^*)$ be the cost of this solution.

Theorem 2. *If for any value $\rho > 1$ there is a ρ-approximation algorithm A for the k-median problem that selects a set of at most $o(\log n)k$ centers, then P=NP.*

Proof. Using the approximation preserving reduction described in the previous theorem, it is possible to show that if algorithm A exists, then A is an $o(\log n)$-approximation algorithm for the minimum dominating set problem. Since Raz and Safra [45] proved that unless P = NP, the dominating set problem does not have an $o(\log n)$-approximation algorithm, then the existence of algorithm A would imply that P = NP.

Since by Theorem 1 an optimum solution for the k-median problem cannot be approximated within any factor $\rho > 1$, we mainly focus our attention on constrained versions of the problem. The most studied version of the problem is the *metric* k-median problem, in which the cost function c is symmetric and it obeys the triangle inequality. This version of the problem is still hard to solve, even if we allow the election of more than k centers:

Theorem 3. *If A is an approximation algorithm for the metric k-median problem that finds a set S of at most $o(\log n)k$ centers that serves the clients D with total service cost $c(S) \leq c(S^*)$, then P=NP.*

Proof. Given an unweighted graph $G = (V, E)$ with minimum dominating set of size k, build a complete graph $G' = (V, E')$ with cost function c satisfying:

– $c(i, j) = 1$ if $(i, j) \in E$,
– $c(i, j) =$ length of a shortest path from i to j in G, if $(i, j) \notin E$.

Since G has a dominating set of size k, the corresponding instance of the k-median problem has a solution of value $n - k$. In this case, algorithm A finds a set S with at most $f(n)k$ centers and service cost $c(S) \leq n-k$, for some function $f(n) = o(\log n)$. We observe that the set S' of vertices not adjacent to S in G has size $|S'| \leq f(n)k$. To see this, assume that $|S'| > f(n)k$. Since the vertices in S' are at distance at least 2 from S, then the service cost of S would be

$$\begin{aligned} c(S) &\geq 2|S'| + n - |S| - |S'| \\ &\geq n + |S'| - |S| \\ &\geq n + f(n)k - f(n)k \\ &= n > n - k. \end{aligned}$$

Hence, set $S \cup S'$ is a dominating set for G of size at most $2f(n)k = o(\log n)k$ and, thus, A is an $o(\log n)$-approximation algorithm for the minimum dominating set problem, which by the aforementioned result of Raz and Safra [45], implies that P=NP.

3 Linear Programming Based Algorithms

A powerful technique that has been successfully applied to design approximation algorithms for a variety of NP-hard problems consists in formulating a problem as an integer program, then, solving the linear program obtained by relaxing the integrality constraints and, finally, rounding this fractional solution to get an integer feasible solution for the original problem. In this section we present two approximation algorithms for the k-median problem that use this technique. First, we present the algorithm of Lin and Vitter [33] which introduces an elegant rounding technique known as *filtering*. This algorithm works for the general k-median problem (with an arbitrary cost function). It finds, for any value $\varepsilon > 0$, a solution of value at most $(1+\varepsilon)$ times the optimum, but it selects up to $(1+\frac{1}{\varepsilon})\ln n$ centers. Then, in the next section we describe an algorithm by Charikar, Guha, Tardos, and Shmoys [13], which refines the filtering technique for the metric version of the problem. This was the first algorithm for the metric k-median problem that achieved constant performance ratio using at most k centers.

3.1 Integer Program Formulation

To formulate the k-median problem as an integer program, we first need to define some variables. For each facility j we define a variable y_j which takes value 1 if j is selected as a center, and it takes value 0 otherwise. For every client-facility pair (i, j) we define a variable x_{ij} which takes value 1 if and only if client i is serviced by center j. The k-median problem can be stated as the following integer program.

$$\text{Minimize} \quad \sum_{j\in F}\sum_{i\in D} c(i,j)x_{ij} \tag{1}$$

$$\begin{aligned} &\text{subject to} \\ &\sum_{j\in F} x_{ij} \geq 1 && \text{for all} \;\; i \in D \\ &\sum_{j\in F} y_j \leq k && \qquad (2) \\ &x_{ij} \leq y_j && \text{for all} \;\; i \in D,\; j \in F \\ &x_{ij},\; y_j \in \{0,1\} && \text{for all} \;\; i \in D,\; j \in F \end{aligned}$$

The linear program relaxation of the above formulation relaxes the last set of constraints allowing the variables y_j and x_{ij} to take fractional non-negative values. This linear program can be solved in polynomial time by using, for example, interior point methods [48,27]. Let $\hat{y} = (\hat{y}_1, \hat{y}_2 \ldots, \hat{y}_{|F|})$, $\hat{x} = (\hat{x}_{11}, \hat{x}_{12}, \ldots, \hat{x}_{|D||F|})$ be an optimum solution for the linear program. Before showing how to use $\hat{x}, \hat{y}$ to find an approximate solution for the k-median problem, we note that a solution for the linear program is completely characterized by the values of the $\hat{y}$ variables, since optimum fractional values for the $\hat{x}$ variables can be determined from $\hat{y}$ [2]. The idea is to (fractionally) assign each client i to its closest facilities. The algorithm for computing $\hat{x}$ from $\hat{y}$ is as follows.

Algorithm AssignClients

1. For each client i consider those facilities $i_1, i_2, \ldots, i_{|F|}$ with values $\hat{y}_{i_\ell} > 0$ in non-decreasing order of service cost: $c(i, i_1) \leq c(i, i_2) \leq \cdots \leq c(i, i_{|F|})$.
2. Find the first facility i_p (in this ordering) for which $\sum_{\ell=1}^{p} \hat{y}_{i_\ell} \geq 1$.

3. Set the values of the variables $\hat{x}_{ij}$ in this manner:
   - $\hat{x}_{ii_\ell} \leftarrow \hat{y}_{i_\ell}$ for all $\ell = 1, 2, \ldots, p-1$, and
   - $\hat{x}_{ii_p} \leftarrow 1 - \sum_{\ell=1}^{p-1} \hat{y}_{i_\ell}$.

3.2 Filtering

The algorithm that we describe in this section approximates the value of the optimum solution for the k-median problem within a factor of $(1+\varepsilon)$, for any value $\varepsilon > 0$. The precision parameter ε affects the number of centers that the algorithm chooses, as we describe below.

The idea of filtering is to use the solution of the linear program to discard, or filter out, some of the possible assignments of clients to centers. This filtering has to be done in such a way that the problem is simplified by reducing the possible number of assignments of clients to facilities, but it has to ensure that at least one good assignment of clients to centers remains.

For each client i let $\hat{C}_i = \sum_{j \in F} c(i,j)\hat{x}_{ij}$ be the fractional cost of servicing i. We filter out some possible assignments of i to facilities by allowing client i to be served only by those facilities j such that $c(i,j) \leq (1+\varepsilon)\hat{C}_i$. Let the *neighbourhood* V_i of a client i be the set of facilities that satisfy the above condition, i.e. $V_i = \{j \mid j \in F \text{ and } c(i,j) \leq (1+\varepsilon)\hat{C}_i\}$.

We show that the neighbourhood of every client is non-empty and, therefore, there is a solution for the k-median problem in which every client is assigned to some facility in its neighbourhood.

Lemma 1. *For every client i, its neighbourhood V_i is non-empty and, furthermore,*

$$\sum_{j \in V_i} \hat{y}_j \geq \frac{\varepsilon}{1+\varepsilon}.$$

Proof. We can show that the neighbourhood V_i of a client i is not empty by using a weighted average argument. If V_i is empty, then for every facility $j \in F$, the service cost c_{ij} is larger than $(1+\varepsilon)\hat{C}_i$. Thus, $\hat{C}_i = \sum_{j \in F} c(i,j)\hat{x}_{ij} > (1+\varepsilon)\hat{C}_i \sum_{j \in F} \hat{x}_{ij} \geq (1+\varepsilon)\hat{C}_i$, by the first constraint of the integer program.

The second part of the lemma can also be proven by using a weighted average argument. Assume that $\sum_{j \in V_i} \hat{y}_i \leq \varepsilon/(1+\varepsilon)$. Then, by the third constraint of the integer program, $\sum_{j \in V_i} \hat{x}_{ij} \leq \varepsilon/(1+\varepsilon)$ and $\sum_{i \notin V_i} \hat{x}_{ij} > 1/(1+\varepsilon)$. Multiplying both sides of the last inequality by $\hat{C}_i$, we get: $\hat{C}_i < (1+\varepsilon)\hat{C}_i \sum_{j \notin V_i} \hat{x}_{ij} < \sum_{j \notin V_i} \hat{x}_{ij} c_{ij} < \hat{C}_i$.

The advantage of assigning each client to a center in its neighbourhood is that, then, the cost of servicing a client is close to the cost of servicing the client in an optimum solution for the problem.

Lemma 2. *If every client i is served by a center j in its neighbourhood V_i, then the cost of the solution is at most $(1+\varepsilon)$ times the cost $c(S^*)$ of an optimum solution for the k-median problem.*

Proof. If it is possible to find a solution x, y for the k-median problem in which clients are assigned to centers in their neighborhoods, then, for every client i there is one facility j for which $x_{ij} = 1$. Therefore, the cost of this solution is

$$\sum_{j\in F}\sum_{i\in D} c(i,j)x_{ij} \le (1+\varepsilon)\sum_{i\in D}\hat{C}_i \le (1+\varepsilon)\sum_{j\in F}\sum_{i\in D} c(i,j)\hat{x}_{ij} \le (1+\varepsilon)c(S^*).$$

According to this lemma, the problem is, then, how to select for each client i a center j from its neighbourhood V_i so that the total number of centers selected is small. This latter problem is an instance of the set covering problem in which the ground set is D, the set of clients, and the family of subsets is $\{S_1, S_2, \ldots, S_{|F|}\}$, where $S_j = \{i \mid i \in D \text{ and } c(i,j) \le (1+\varepsilon)\hat{C}_i\}$.

The greedy set cover algorithm [16,24,35], can be used to approximately solve the set covering problem, and it finds a set S of centers of size at most $\bar{s}(\ln n + 1)$, where $\bar{s}$ is the value of a fractional set cover for the above set cover problem. The value of $\bar{s}$ can be easily obtained from the solution $\hat{x}, \hat{y}$ of the linear program since by Lemma 1, for each client i, $\sum_{j\in V_i} \hat{y}_j > \varepsilon/(1+\varepsilon)$. Therefore, setting $\bar{y}_j = (1+\frac{1}{\varepsilon})\hat{y}_j$ for every facility j yields a fractional solution for the set cover problem since, then, for each client i:

$$\sum_{i\in S_j} \bar{y}_j = \sum_{j\in V_i}\left(1+\frac{1}{\varepsilon}\right)\hat{y}_j > 1.$$

The value of this fractional set cover $\bar{y}$ is

$$\sum_{j\in F} \bar{y}_j = \left(1+\frac{1}{\varepsilon}\right)\sum_{j\in F}\hat{y}_j \le \left(1+\frac{1}{\varepsilon}\right)k.$$

The last inequality follows from the second constraint of the integer program. The algorithm is as follows.

Algorithm Filtering

1. Solve the linear program relaxation of the k-median problem to get a fractional solution $\hat{y}, \hat{x}$.
2. For each client i, compute $\hat{C}_i = \sum_{j\in F} c(i,j)\hat{x}_{ij}$.
3. Use the greedy set cover algorithm on the instance with ground set D, and family of subsets $\{S_1, S_2, \ldots, S_{|F|}\}$, where each set S_j is formed by the clients i such that $c(i,j) \le (1+\varepsilon)\hat{C}_i$.
4. Choose as centers those facilities selected by the greedy set cover algorithm and assign each client i to one of its closest centers.

Theorem 4. *For any value $\varepsilon > 0$, the above algorithm finds a solution of value at most $(1+\varepsilon)$ times the value of an optimum solution for the k-median problem, and this solution has at most $(1+\frac{1}{\varepsilon})(\ln n + 1)k$ centers.*

Proof. The theorem follows from Lemma 2 and the above discussion on the greedy set cover algorithm.

3.3 An Algorithm with Constant Performance Ratio

We describe now an algorithm for the metric k-median problem with constant performance ratio. This algorithm was designed by Charikar, Guha, Tardos, and Shmoys [13], and it uses a more sophisticated version of the filtering technique than that described in the previous section. This algorithm works for the more general version of the problem where each one of the clients i has a non-negative demand d_i, and the objective is to minimize the total service cost weighted by the demands: $\min\limits_{S \subseteq F}\{\sum\limits_{i \in D} d_i \times \min\limits_{j \in S}\{c(i,j)\} \mid |S| \leq k\}$.

We describe here the version of the problem when the set of clients D and the set of facilities F are the same. Or in other words, we consider that there is a set $N = D \cup F$ of locations; each location i has some demand d_i and in each location it is possible to build a center. The distance between locations i and j is $c(i,j)$.

The linear program relaxation of the integer program formulation of this problem is as follows.

$$\text{Minimize} \quad \sum_{i,j \in N} d_i c(i,j) x_{ij} \tag{3}$$

$$\begin{aligned}
&\text{subject to} \\
&\sum_{j \in N} x_{ij} \geq 1 && \text{for all } \; i \in N \\
&\sum_{j \in N} y_j \leq k && \qquad (4) \\
&x_{ij} \leq y_j && \text{for all } \; i,j \in N \\
&0 \leq x_{ij}, y_j && \text{for all } \; i,j \in N
\end{aligned}$$

The problem defined by this linear program is known as the *fractional k-median problem with demands.* Given an optimum solution $\hat{x}, \hat{y}$ for this linear program, the fractional service cost of a location i is $\hat{C}_i = \sum_{j \in N} c(i,j)\hat{x}_{ij}$. Let us assume that the locations are indexed non-decreasingly by fractional service cost, so $\hat{C}_1 \leq \hat{C}_2 \leq \cdots \leq \hat{C}_{|N|}$. The algorithm is as follows.

Algorithm CONSOLIDATE

```
1. Find an optimum solution $\hat{x}, \hat{y}$ for linear program (3).
2. Re-assign demands as follows.
   Set $d'_i \leftarrow d_i$ for each location $i$.
   For each location $i$:
       If there is a location $j$ with fractional service cost $\hat{C}_j < \hat{C}_i$ and such that
       $d'_j > 0$ and $c(i,j) \leq 4\hat{C}_j$, then move the demand of $i$ to location $j$:
           $d'_j \leftarrow d'_j + d'_i$,
           $d'_i \leftarrow 0$.
```

3. Let $N'_> = \{i \in N \mid d'_i > 0\}$, be the set of locations with positive demand. We show below that $|N'_>| \leq 2k$. For each location $i \in N$, find the location $s(i) \in N'_>$ closest to it. In case of ties, choose the location with minimum index.
4. Set $x' \leftarrow \hat{x}$ and $y' \leftarrow \hat{y}$.
   For each location i with $y'_i > 0$ and $d'_i = 0$:
   - Set $y'_{s(i)} \leftarrow \min\{1, y'_i + y'_{s(i)}\}$, and set $y'_i \leftarrow 0$.
   - For each location $j \in N$, set $x'_{j\,s(i)} \leftarrow x'_{j\,s(i)} + x'_{ji}$, and set $x'_{ji} \leftarrow 0$.
5. Sort the locations $i \in N'_>$ in non-increasing order of value $d'_i\, c(i, s(i))$.
   Set $\bar{y}_i \leftarrow 1$ for the first $2k - |N'_>|$ locations $i \in N'_>$.
   Set $\bar{y}_i = \frac{1}{2}$ for the remaining $2(|N'_>| - k)$ locations $i \in N'_>$.
   Set $\bar{y}_i \leftarrow 0$ for all locations $i \in N \setminus N'_>$.
   For each location $i \in N$:
   Set $\bar{x}_{ii} \leftarrow \bar{y}_i$, and $\bar{x}_{is(i)} \leftarrow 1 - \bar{y}_i$.
6. Build a graph $H = (V_H, E_H)$ having as vertices the locations $i \in N'_>$.
   For every vertex i with $\bar{y}_i = \frac{1}{2}$ add an edge between $s(i)$ and i. Note that graph H is a forest.
   Find a dominating set I for H containing all vertices i with $\bar{y}_i = 1$, and such that $|I| \leq k$.
7. Select I as the set of centers. Let each client i be served by the center $j \in I$ with minimum service cost $c(i, j)$.

This algorithm first simplifies the problem by moving demands so that there are at most $|N'_>| \leq 2k$ locations with positive demands. These locations are far away from each other, and this allows the possibility of finding a $\frac{1}{2}$-integral solution for the problem (a solution in which every variable has value either 0, $\frac{1}{2}$, or 1). This solution can, then, be transformed into an integral solution by rounding the variables with value $\frac{1}{2}$ up to 1 or down to 0. Note that step 4 of the algorithm is not needed, we include it only to simplify the analysis. We analyze now the algorithm in more detail.

3.4 Analysis

First, we show that the re-assignment of demands does not increase the cost of the solution.

Lemma 3.

$$\sum_{i,j \in N} d'_i\, c(i,j)\hat{x}_{ij} \leq \sum_{i,j \in N} d_i\, c(i,j)\hat{x}_{ij}.$$

Proof. Since $\sum_{i,j \in N} d_i\, c(i,j)\hat{x}_{ij} = \sum_{i \in N} d_i \hat{C}_i$, and in step 2 demand is moved from location i to location j only if $\hat{C}_j \leq \hat{C}_i$, the claim follows.

Re-assigning centers as described in step 4 increases the value of the fractional solution by at most a factor of 2.

Lemma 4.

$$\sum_{i,j \in N} d'_i\, c(i,j)x'_{ij} \leq 2 \sum_{i,j \in N} d'_i\, c(i,j)\hat{x}_{ij}.$$

Proof. Note that x', y' is a feasible solution for linear program (3). Consider some location $i \in N'_>$ that is (partially) served in solution $\hat{x}, \hat{y}$ by some location $j \notin N'_>$. The (fractional) center at j is moved to location $s(j)$ and, hence, the service cost of i increases to $c(i, s(j))\, \hat{x}_{ij} \leq (c(i,j) + c(j, s(j)))\hat{x}_{ij} \leq 2\, c(i,j)\, \hat{x}_{ij}$, since by definition of $s(j)$, $c(j, s(j)) \leq c(j,i) = c(i,j)$. Therefore, the total service cost is increased by at most a factor of 2.

We show that the solution x', y' built in step 4 is such that at least one half of a center is assigned to each location $i \in N'_>$, i.e., $y'_i \geq \frac{1}{2}$. By constraint (4) of the linear program (3), this means that set $N'_>$ has at most $2k$ locations.

Lemma 5. *For each location $i \in N'_>$, $y'_i \geq \frac{1}{2}$.*

Proof. For any two locations $i, j \in N'_>$, $c(i,j) > 4\max\{\hat{C}_i, \hat{C}_j\}$ because otherwise the demands for i and j would have been merged in step 2 of the algorithm. Hence, after re-assigning demands, for each location $i \in N'_>$ all (fractional) centers within distance $2\,\hat{C}_i$ of i are moved to i in step 4. Note that for any $i \in N'_>$,

$$\sum_{j : c(i,j) > 2\hat{C}_i} \hat{x}_{ij} < \frac{1}{2}$$

because, otherwise

$$\sum_{j : c(i,j) > 2\hat{C}_i} c(i,j)\hat{x}_{ij} > 2\hat{C}_i \sum_{j : c(i,j) > 2\hat{C}_i} \hat{x}_{ij} \geq \hat{C}_1$$

which is a contradiction. Since $\sum_{j \in N} \hat{x}_{ij} \geq 1$ and $\hat{x}_{ij} \leq \hat{y}_j$ for each $i \in N$, then

$$\sum_{j : c(i,j) \leq 2\hat{C}_i} \hat{y}_j \geq \sum_{j : c(i,j) \leq 2\hat{C}_i} \hat{x}_{ij} \geq \frac{1}{2}.$$

Therefore, in step 4 of the algorithm the value assigned to each variable y'_i, $i \in N'_>$, is at least $\frac{1}{2}$.

It is easy to verify that the $\frac{1}{2}$-integral solution $\bar{x}, \bar{y}$ built in step 5 is feasible for the linear program (3) with demands d'. As for its cost, we show that it is not larger than the cost of solution x', y'.

Lemma 6.

$$\sum_{i,j \in N} d'_i c(i,j)\bar{x}_{ij} \leq \sum_{i,j \in N} d'_i c(i,j) x'_{ij}.$$

Proof.

$$\begin{aligned} \sum_{i,j \in N} d'_i c(i,j)\bar{x}_{ij} &= \sum_{i \in N'_>} \sum_{j \in N'_>} d'_i c(i,j)\bar{x}_{ij} \\ &= \sum_{i \in N'_>} d'_i c(i, s(i))(1 - \bar{y}_i) \\ &\leq \sum_{i \in N'_>} d'_i c(i, s(i))(1 - y'_i) \qquad (5) \end{aligned}$$

To show that the last inequality is true, let us define N_1' as the set of centers $i \in N_>'$ for which $\bar{y}_i = 1$. Also, let $d_\ell' c(\ell, s(\ell)) = \min_{i \in N_1'} \{d_i' c(i, s(i))\}$. Because of the way in which set N_1' is chosen in step 5 of the algorithm, we know that $d_\ell' c(\ell, s(\ell)) \geq d_i' c(i, s(i))$ for all $i \notin N_1'$. Finally, recall that $y_i' \leq 1$ for all $i \in N$. Hence,

$$\begin{aligned}
&\sum_{i \in N_>'} d_i' c(i, s(i))(1 - \bar{y}_i - (1 - y_i')) \\
&\qquad = \sum_{i \in N_>'} d_i' c(i, s(i))(y_i' - \bar{y}_i) \\
&\qquad = \sum_{i \in N_1'} d_i' c(i, s(i))(y_i' - 1) + \sum_{i \notin N_1'} d_i' c(i, s(i))(y_i' - \frac{1}{2}) \\
&\qquad \leq \sum_{i \in N_1'} d_\ell' c(\ell, s(\ell))(y_i' - 1) + \sum_{i \notin N_1'} d_\ell' c(\ell, s(\ell))(y_i' - \frac{1}{2}) \\
&\qquad = \sum_{i \in N_1'} d_\ell' c(\ell, s(\ell))(y_i' - \bar{y}_i) + \sum_{i \notin N_1'} d_\ell' c(\ell, s(\ell))(y_i' - \bar{y}_i) \\
&\qquad = d_\ell' c(\ell, s(\ell)) \sum_{i \in N_>'} (y_i' - \bar{y}_i) \\
&\qquad = 0
\end{aligned}$$

The last equation follows from the fact that $\sum_{i \in N_>'} y_i' = \sum_{i \in N_>'} \bar{y}_i = k$. To complete the proof, we note that

$$\sum_{\substack{j \in N' \\ j \neq i}} x_{ij}' \geq 1 - x_{ii}' \geq 1 - y_i'.$$

Therefore,

$$\sum_{i \in N_>'} d_i' c(i, s(i))(1 - y_i') \leq \sum_{i \in N_>'} \sum_{j \in N_>'} d_i' c(i, s(i)) x_{ij}' \leq \sum_{i \in N_>'} \sum_{j \in N_>'} d_i' c(i, j) x_{ij}'.$$

Combining this with inequality (5), the claim follows.

Now, we are ready to show that the algorithm has a constant performance ratio.

Theorem 5. *Algorithm* Consolidate *finds a solution of cost at most* 8 *times the cost of an optimum solution for the metric k-median problem.*

Proof. It is not hard to see that the graph H built in step 6 is a forest and, hence, we can efficiently find the required dominating set I as follows. First, add to I all those vertices i with $\bar{y}_i = 1$ and, then, remove such vertices from H. Any isolated vertex in the resulting graph H is dominated by at least one vertex in I and, thus, these vertices are also deleted from H. After these deletions, H is a forest in which every tree contains at least 2 vertices, and the total number of vertices in H is at most $|N'_>| - |I| = 2(|N'_>| - k)$. A minimum dominating set I' of H can be easily found and it includes at most half of the vertices in each tree of H. Thus, $|I'| \leq |N'_>| - k$. We set $I \leftarrow I \cup I'$ to get a dominating set for the original graph H of size $|I| \leq 2k - |N'_>| + |N'_>| - k = k$ as desired.

Consider this solution I. Each vertex $i \in I$ with $\bar{y}_i = 1$ does not contribute to the cost of the solution since $\bar{x}_{ii} = \bar{y}_i = 1$ and $c(i,i) = 0$. For those vertices i with $\bar{y}_i = \frac{1}{2}$, either $i \in I$ or $s(i) \in I$. Hence, the contribution of i to the cost of the solution is at most $d'_i\, c(s(i), i) \leq 2d'_i\, c(s(i), i)\bar{x}_{i\,s(i)}$. Therefore, the cost $c(I)$ of solution I is at most 2 times the cost of the half integral solution $\bar{x}, \bar{y}$. By Lemmas 4 and 6, $c(I)$ is at most 4 times the cost of the fractional solution $\hat{x}, \hat{y}$ with demands d'.

To determine the effect of the re-allocation of demands performed in step 2 on the cost of the solution, consider a location $j \in N$ which has its demand moved to another location $j' \in N'$. For this location $c(j', j) \leq 4\hat{C}_j$. Let j' be served by center i in solution I. If we move back the demand to j and let j be served by center i, this would increase the cost of the solution by at most $d_j\, c(j', j) \leq 4d_j\hat{C}_j$. Therefore, moving all demands back to the original locations increases the cost of the solution by at most 4 times the cost of solution $\hat{x}, \hat{y}$.

By the above arguments, the cost of solution I is at most 8 times the cost of an optimum solution for the k-median problem.

By using a more complex rounding algorithm than the one described in step 6 of the algorithm, Charikar et al. [13] are able to show that the performance ratio of the algorithm can be improved to $6\frac{2}{3}$.

4 A Primal-Dual Algorithm

In this section we describe another linear programming based approximation algorithm for the k-median problem. The approach that we present now differs from the ones presented in the previous section in that it does not need to solve a linear program, but rather it finds primal and dual solutions for the problem using combinatorial methods. This results in a faster algorithm, and, as we show, the performance ratio is better than the performance ratio of the algorithm of Charikar et al. [13].

The primal-dual algorithm that we present here is due to Jain and Vazirani [23]. This algorithm is based on an interesting relationship between the k-median problem and the uncapacitated facility location problem. In order to understand the algorithm we need first to describe a primal-dual approximation algorithm for the uncapacitated facility location problem.

4.1 The Uncapacitated Facility Location Problem

The uncapacitated facility location problem differs from the k-median problem in that facilities have assigned building costs $f(j)$ and there is no bound on the number of facilities that might be selected as centers. The goal is to minimize the total cost for servicing the clients plus the cost of the facilities selected. An integer program defining the uncapacitated facility location problem is very similar to integer program (1), with the differences stated above. A linear program relaxation of this integer program is given below.

$$\text{Minimize} \sum_{j \in F} \sum_{i \in D} c(i,j) x_{ij} + \sum_{j \in F} f(j) y_j \tag{6}$$

$$\begin{aligned} &\text{subject to} \\ &\sum_{j \in F} x_{ij} \geq 1 && \text{for all } i \in D \\ &x_{ij} \leq y_j && \text{for all } i \in D,\ j \in F \\ &0 \leq x_{ij},\ y_j && \text{for all } i \in D,\ j \in F \end{aligned}$$

The dual linear program corresponding to this linear program is the following (for a comprehensive study of linear programming concept the reader is referred to [17,41,46]).

$$\text{Maximize} \sum_{i \in D} \alpha_i \tag{7}$$

$$\begin{aligned} &\text{subject to} \\ &\alpha_i - \beta_{ij} \leq c(i,j) && \text{for all } i \in D,\ j \in F \\ &\sum_{i \in D} \beta_{ij} \leq f(j) && \text{for all } j \in F \\ &\alpha_i,\ \beta_{ij} \geq 0 && \text{for all } i \in D,\ j \in F \end{aligned} \tag{8}$$

There is a nice interpretation for the dual variables α, β. Let us think, as it happens in the real world, that the clients must pay for the service cost and for the cost of building the selected facilities. If client i is served by facility j, then the amount α_i paid by the client must be at least equal to $c(i,j)$. If $\alpha_i > c(i,j)$, the rest of the money paid by the client, i.e., $\beta_{ij} = \alpha_i - c(i,j)$, goes towards paying for the cost of building facility j. By the complementary slackness conditions, the second constraint of the dual linear program is *tight* (which means that $\sum_{i \in D} \beta_{ij} = f(j)$) if facility j is selected. By the above argument, the total amount $\sum_{i \in D} \beta_{ij}$ contributed by the clients served by j, has to be exactly equal to the building cost of facility j.

To solve the dual linear program, we must determine the price that each client must pay. From the dual solution it is easy to determine which facilities are selected. Each client is assigned to that facility with smallest service cost.

An instance of the uncapacitated facility location problem can be modeled with a graph $G = (D \cup F, E)$ having as vertices the clients and facilities, and edges connecting every facility to each client. The weight of an edge (i,j) is the service cost $c(i,j)$. In the above primal-dual context, an edge (i,j) is said to be *tight* if $\alpha_i \geq c(i,j)$ (or in other words, if $\alpha_i = c(i,j) + \beta_{ij}$ and $\beta_{ij} \geq 0$), and a facility j is said to be *paid for* if $\sum_{i \in D} \beta_{ij} = f(j)$. A client i is *marked* if there is a facility j which has been paid for, and edge (i, j) is tight. The algorithm for approximately solving the uncapacitated facility location problem is as follows.

Algorithm PrimalDual

```
0. Initially all clients are un-marked. All values $\alpha_i$ and $\beta_{ij}$ are initialized to 0.
1. Repeat steps 2 and 3 as long as there are un-marked clients.
2. Simultaneously and uniformly (at the same rate) raise the values of, both,
   the dual variables $\alpha_i$ for all un-marked clients $i$ and the variables $\beta_{ij}$ for all
   tight edges $(i, j)$, until either:
     – $\alpha_i = c(i, j)$ for some edge $(i, j)$, or
     – $\sum_{i \in D} \beta_{ij} = f(j)$ for some facility $j$.
   In the first case we label edge $(i, j)$ as tight. In the second case we label
   facility $j$ as paid for.
3. Every un-marked client $i$ with a tight edge $(i, j)$ connecting it to a paid for
   facility $j$ is marked.
4. Build the graph $T = (D \cup F, E')$ containing those edges $(i, j)$ for which
   $\beta_{ij} > 0$.
5. Build the square $T^2$ of graph $T$ by adding an edge between vertices $u$ and $v$
   if there is a path of length at most 2 between $u$ and $v$ in $T$.
6. Build the subgraph $H$ of $T^2$ induced by those facilities that are paid for.
7. Find a maximal independent set $I$ of $H$.
8. Select $I$ as the set of centers and let each client $i$ be served by a center
   $j \in I$ for which the service cost $c(i, j)$ is minimum. If for a client $i$ there are
   2 or more centers with minimum service cost, choose any one of them for
   servicing $i$.
```

In the solution I produced by this algorithm, we say that a client i is *directly connected* to the facility j that serves it if $\beta_{ij} > 0$. Otherwise client i is said to be *indirectly connected.*

4.2 Performance Ratio

We interpret the value of variable α_i as the price that client i has to pay for, both, building a facility and for being serviced by that facility. Let client i be serviced by facility j. Let α_i^f be the price that client i pays for building facility j and let α_i^s be the service cost paid by the client. Clearly, $\alpha_i = \alpha_i^s + \alpha_i^f$. If client i is directly connected to j, then $\alpha_i = c(i, j) + \beta_{ij}$, and so we set $\alpha_i^s \leftarrow c(i, j)$, and $\alpha_i^f \leftarrow \beta_{ij}$. If i is indirectly connected, then we set $\alpha_i^s \leftarrow \alpha_i$ and $\alpha_i^f \leftarrow 0$.

Lemma 7.

$$\sum_{j \in I} f(j) = \sum_{i \in D} \alpha_i^f .$$

Proof. Every facility $j \in I$ is paid for, i.e. $f(j) = \sum_{i \in D} \beta_{ij} = \sum_{i \in D_j} \alpha_i^f$, where D_j is the set of clients directly connected to j. Since sets D_j are disjoint, the claim follows.

Lemma 8. *For every indirectly connected client i, $c(i,j) \leq 3\alpha_i^s$, where $j \in I$ is the facility that serves i.*

Proof. Since the loop in steps 1-3 terminates when all clients are marked, then, i is marked in step 3 because these is a facility j' for which $\alpha_i \geq c(i,j')$. However, $j' \notin I$ since i is indirectly connected. As i is indirectly connected, facility j' was not selected to be in the final solution I computed in step 7. Note that since I is a maximal independent set of H, there has to be at least one facility $k \in I$ such that there is an edge from j' to k in the graph H built in step 6 (see Figure 1). Because clients are assigned to their nearest centers in step 8, then $c(i,k) \geq c(i,j)$.

Observe that k and j' are facilities and so there is no edge (k,j') in the graph $G = (D \cup F, E)$. Since edge (k,j') belongs to H, there must be a client h with edges (h,k) and (h,j') such that $\beta_{hk} > 0$ and $\beta_{hj'} > 0$. This implies that $\alpha_h > c(h,k)$ and $\alpha_h > c(h,j')$.

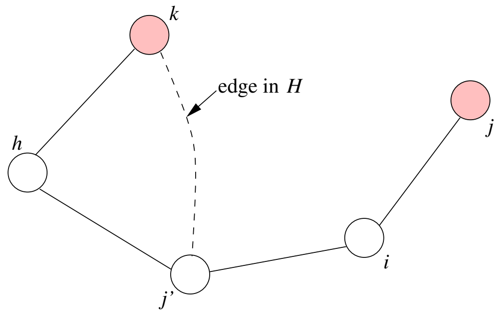

Fig. 1. Service cost of client i. Centers k and j belong to I, but $j' \notin I$.

The algorithm does not rise the value of α_h after facility j' is paid for (and in fact it might stop raising the value of α_h when k or other neighbouring facility of h is paid for). Since, as we assumed above, i is marked when α_i takes value $c(i,i')$, then α_i is raised at least until the time when j' is paid for. Therefore, $\alpha_i > \alpha_h$. By the triangle inequality (see Figure 1), $c(i,j) \leq c(i,k) \leq c(i,j') + c(h,j') + c(h,k) \leq \alpha_i + 2\alpha_h \leq 3\alpha_i$.

Using these two lemmas, we can compute the performance ratio of algorithm PrimalDual.

Theorem 6. *The performance ratio of algorithm* PrimalDual *is* 3.

Proof. Let x, y and α, β be the primal and dual solutions produced by the algorithm. Since for a client i directly connected to a facility j, $\alpha_i^s = c(i,j)$, and by Lemma 8, for each indirectly connected client i, $c(i,j) \leq 3\alpha_i^s$, then

$$\sum_{j \in F} \sum_{i \in D} c(i,j) x_{ij} \leq 3 \sum_{i \in D} \alpha_i^s.$$

From this inequality and Lemma 7, we get

$$\sum_{j\in F}\sum_{i\in D} c(i,j)x_{ij} + 3\sum_{j\in F} f(j)y_j \le 3\sum_{i\in D}(\alpha_i^s + \alpha_i^f) = 3\sum_{i\in D}\alpha_i. \tag{9}$$

Since the value of an optimum solution for the k-median problem is at least $\sum_{i\in D}\alpha_i$, the claim follows.

4.3 Running Time

Steps 1-3 are, perhaps, the most difficult steps of the algorithm to implement and analyze, so we will give some details here as to how these steps can be efficiently implemented. Note that since the values α_i are raised uniformly, the order in which the edges will become tight is consistent with a non-decreasing ordering of the edges by cost. Hence, if we store the edges in a list L in non-decreasing order of cost, we will know the order in which the events $\alpha_i = c(i,j)$ of step 2 will take place. To keep track of the second class of events that take place in step 2, namely $\sum_{i\in D}\beta_{ij} = f(j)$ for some facility j, we need to maintain for each facility j the following 3 variables:

- a variable b_j giving the number of tight edges currently contributing to facility j,
- a variable t_j giving the time when the value of b_j last changed, and
- a variable p_j giving the total contribution made by clients to facility j up to time t_j.

To determine the event that will take place during the following iteration of steps 2-3 we first compute

$$\tau = \min\left\{c(i,j), \min_{j\in F}\left\{\frac{f(j)-p_j}{b_j}\right\}\right\},$$

where (i,j) is the first edge in L, if any. If L is empty, we simply ignore the first term $c(i,j)$. We can efficiently compute τ by using a heap h.

a. If $c(i,j) = \tau$, then edge (i,j) becomes tight in this iteration, so we discard (i,j) from L.
   - If j is not yet paid for, we update $p_j \leftarrow p_j + (\tau - t_j)b_j$ and, then, increase the value of b_j by 1 since in the next iterations the value of β_{ij} will be increased. Finally, we set $t_j \leftarrow \tau$. These steps can be performed in constant time and updating the value $\frac{f(j)-p_j}{b_j}$ associated with j in the heap needs $O(\log n_f)$ time, where n_f is the number of facilities.
   - If j is already paid for, then we mark i. Since from this point on the value α_i and the values $\beta_{ij'}$ for all tight edges (i,j') will not increase any more, then, we need to update the values of the variables for such facilities j'. For each facility j' such that (i,j') is tight we set $p_{j'} \leftarrow p_{j'} + b_{j'}(\tau - t_{j'})$ and, then, we decrease $b_{j'}$ by 1. Finally, we set $t_{j'} \leftarrow \tau$. The total amount of time needed to perform these steps is $O(\text{degree}(i)\log n_f)$, where degree$(i)$ is the degree of client i in the graph $G = (D\cup F, E)$.

b. On the other hand, if $\frac{f(j)-p_j}{b_j} = \tau$, then facility j will next get paid for, so we remove this value $\frac{f(j)-p_j}{b_j}$ from the heap. Furthermore, all clients i with tight edges to j will be marked, and so their α_i and β_{ij} values will stop increasing. For each one of these clients with tight edges (i, j') we need to update the values of $p_{j'}$, $t_{j'}$, and $b_{j'}$ as described above. Let S_j be the set of clients marked in this step. The total time needed to perform the above steps is $O(\sum_{i \in S_j} \text{degree}(i) \log n_f)$.

Since every client is marked only once, then the total time needed to complete steps 1-3 is $O(\sum_{i \in D} \text{degree}(i) \log n_f) = O(m \log n_f)$, where m is the number of edges in the graph $G = (D \cup F, E)$.

4.4 Algorithm for the k-Median Problem

By studying the integer program formulations for the k-median and uncapacitated facility location problems, we note that the Lagrangian relaxation of constraint (2) in the integer program (1) for the k-median problem gives an instance of the uncapacitated facility location problem in which all the facilities have the same cost z, where z is the Lagrange multiplier:

$$\begin{aligned} \text{Minimize} \quad & \sum_{j \in F} \sum_{i \in C} c(i,j) x_{ij} + \sum_{j \in F} z y_j \qquad (10) \\ \text{subject to} \quad & \\ & \sum_{j \in F} x_{ij} \geq 1 \qquad \text{for all } \; i \in D \\ & x_{ij} \leq y_j \qquad \text{for all } \; i \in D, \; j \in F \\ & x_{ij}, \; y_j \in \{0,1\} \qquad \text{for all } \; i \in D, \; j \in F \end{aligned}$$

The value of the Lagrange multiplier z controls the number of facilities selected by the algorithm PrimalDual. As the value of z increases, the number of facilities selected decreases. If it happens that for some value of z the algorithm chooses exactly k facilities, then this would be a good solution for the k-median problem:

Lemma 9. *Suppose that for some value z algorithm* PrimalDual *selects exactly k facilities. The cost of this solution is at most* 3 *times the cost of an optimum solution for the k-median problem.*

Proof. Let x, y and α, β be the primal and dual solutions constructed by the algorithm. Then, by equation (9),

$$\sum_{j \in F} \sum_{i \in D} c(i,j) x_{ij} \leq 3 \left(\sum_{i \in D} \alpha_i - kz \right) \qquad (11)$$

Observe that x, y is a feasible solution for the k-median problem and α, β, z is a feasible solution for the dual linear program for the k-median problem:

$$\begin{aligned}
&\text{Maximize} \quad \sum_{i \in D} \alpha_i - zk \qquad\qquad (12)\\
&\text{subject to}\\
&\quad \alpha_i - \beta_{ij} \le c(i,j) \qquad \text{for all } \; i \in D, \; j \in F\\
&\quad \sum_{i \in D} \beta_{ij} \le z \qquad \text{for all } \; j \in F\\
&\quad \alpha_i, \; \beta_{ij} \ge 0 \qquad \text{for all } \; i \in D, \; j \in F
\end{aligned}$$

Therefore, by the weak duality theorem, x, y is a solution for the k-median problem of cost $\sum_{j \in F} \sum_{i \in D} c(i,j) x_{ij}$ at most 3 times the cost of an optimum solution for the problem.

However, there might not be a value z for which algorithm PrimalDual selects exactly k centers, or it might be the case that finding such a value might take too long[1]. What we can do in that case, is to find two "close"values z_1 and z_2 for the facility costs, such that for the first value the algorithm will choose $k_1 < k$ and for the second one it will select $k_2 > k$ centers, respectively. We combine these solutions to get a fractional solution with exactly k centers. The details are as follows.

Algorithm ConvexCombination

1. Compute $c_{\min} = \min\{c(i,j) \mid i \in D, j \in F\}$ and $c_{\max} = \max\{c(i,j) \mid i \in D, j \in F\}$.
   Set $n_f \leftarrow |F|$.
2. Use algorithm PrimalDual and binary search over the interval $[0, nc_{\max}]$ to find two values, z_1 and z_2, such that $z_1 - z_2 \le c_{\min}/(4n_f^2)$ for which algorithm PrimalDual finds solutions
   – $x^s, y^s, \alpha^s, \beta^s$with $k_1 < k$ centers, and
   – $x^\ell, y^\ell, \alpha^\ell, \beta^\ell$ with $k_2 > k$ centers, respectively.
   Let A be the set of centers chosen in solution x^s, y^s, and let B be the set of centers chosen in solution x^ℓ, y^ℓ.
3. Let $a = \frac{k_2 - k}{k_2 - k_1}$ and $b = \frac{k - k_1}{k_2 - k_1}$. Combine the two above solutions to get a new solution $(\hat{x}, \hat{y}) = a(x^s, y^s) + b(x^\ell, y^\ell)$ that (fractionally) opens exactly k centers.
4. $B' \leftarrow \emptyset$
   For each facility $j \in A$ **do**
   Remove from B the facility j' with minimum $c(j, j')$ value, and include it into set B'.

[1] Recently Archer et al. [1] designed a variant of Jain and Vazirani's algorithm which guarantees the existence of a value k for which exactly k centers are selected. Unfortunately, this algorithm requires the solution of the maximum independent set problem, and so it is not guaranteed to run in polynomial time.

5. Choose a set I of k centers from $A \cup B$ as follows:
   – Select all centers in A with probability a and select all centers in B' with probability b (note that $b = 1 - a$).
   – Randomly select $k - k_1$ centers from B.
6. Select I as the set of centers and let each client i be served by the center $j \in I$ with minimum service cost $c(i, j)$.

Lemma 10. *The cost of the fractional solution $\hat{x}, \hat{y}$ computed in step 3 is at most $(3 + \frac{1}{n_f})$ times the cost of an optimum fractional solution for the k-median problem.*

Proof. Let $\alpha = a\alpha^s + b\alpha^\ell$, and $\beta = a\beta^s + b\beta^\ell$. Note that, α, β, z_1 is a feasible solution for the dual linear program (12) and, thus, its value is a lower bound for the value of the optimum fractional solution for the k-median problem. Let us compute the value of this dual solution. By equation (9),

$$3\left(\sum_{i \in D} \alpha_i^\ell - z_2 k_2\right) \geq \sum_{j \in F} \sum_{i \in D} c(i,j) x_{ij}^\ell \geq c_{\min} \geq (3n_f + 3)(z_1 - z_2)k_2.$$

This last inequality follows since $k_2 \leq n_f$, and $z_1 - z_2 \leq c_{\min}/(4n_f^2)$. Furthermore, we need to assume that $n_f \geq 3$. This is a reasonable assumption since if the number of facilities is less than 3, then the k-median problem can be easily solved by trying the at most $O(n^2)$ subsets of k facilities as possible solutions for the problem. Hence,

$$3\left(\sum_{i \in D} \alpha_i^\ell - z_1 k_2\right) = 3\left(\sum_{i \in D} \alpha_i^\ell - z_2 k_2\right) - 3(z_1 - z_2)k_2 \geq 3n_f(z_1 - z_2)k_2.$$

So, $(z_1 - z_2)k_2 \leq \frac{1}{n_f}\left(\sum_{i \in D} \alpha_i^\ell - z_1 k_2\right)$ and, therefore,

$$3\left(\sum_{i \in D} \alpha_i^\ell - z_2 k_2\right) = 3\left(\sum_{i \in D} \alpha_i^\ell - z_1 k_2\right) + (z_1 - z_2)k_2 \leq \left(3 + \frac{1}{n_f}\right)\left(\sum_{i \in D} \alpha_i^\ell - z_1 k_2\right).$$

Combining the above inequalities, we get

$$\sum_{j \in F} \sum_{i \in D} c(i,j) x_{ij}^\ell \leq \left(3 + \frac{1}{n_f}\right)\left(\sum_{i \in D} \alpha_i^\ell - z_1 k_2\right).$$

Also, by inequality (9),

$$\sum_{j \in F} \sum_{i \in D} c(i,j) x_{ij}^s \leq 3\left(\sum_{i \in D} \alpha_i^s - z_1 k_1\right).$$

From the last two inequalities, we finally get:

$$\begin{aligned}\sum_{j\in F}\sum_{i\in D} c(i,j)\hat{x}_{ij} &\le \left(3+\frac{1}{n_f}\right)\left(a\sum_{i\in D}\alpha_i^s + b\sum_{i\in D}\alpha_i^\ell - z_1 k\right)\\ &= \left(3+\frac{1}{n_f}\right)\left(\sum_{i\in D}\alpha_i - z_1 k\right).\end{aligned}$$

Let x^I, y^I be the solution for integer program (1) corresponding to the solution I computed by algorithm ConvexCombination. The expected cost of this solution is bounded in the following Lemma.

Lemma 11.

$$E[\sum_{j\in F}\sum_{i\in D} c(i,j)x_{ij}^I] \le (1+\max\{a,b\})\sum_{j\in F}\sum_{i\in D} c(i,j)\hat{x}_{ij},$$

where $\hat{x}$ is the fractional solution computed in step 3.

Proof. Consider a client i and the sets of facilities A, B, and B' constructed by the algorithm in steps 2 and 4. Let $j_1 \in A$ and $j_2 \in B$ be the centers that serve i in solutions x^s, y^s and x^ℓ, y^ℓ, respectively. We consider two cases.

1. If $j_2 \in B'$, then either $j_1 \in I$ or $j_2 \in I$. Hence,

$$E[\sum_{j\in F} c(i,j)x_{ij}^I] = a\,c(i,j_1) + b\,c(i,j_2) = \sum_{j\in F} c(i,j)\hat{x}_{ij}.$$

2. If $j_2 \notin B'$, then
   - the probability that $j_2 \in I$ is b,
   - the probability that $j_2 \notin I$ and $j_1 \in I$ is $a(1-b) = a^2$, and
   - the probability that $j_1 \notin I$ and $j_2 \notin I$ is $(1-a)(1-b) = ab$.

   Let $j_3 \in B'$ be the facility that is paired to j_1 in step 4 of the algorithm. Then,

$$E[\sum_{j\in F} c(i,j)x_{ij}^I] \le b\,c(i,j_2) + a^2 c(i,j_1) + ab\,c(i,j_3).$$

   Since j_3 is the center in B closest to j_1, then $c(j_1,j_3) \le c(j_1,j_2)$, and so $c(i,j_3) \le c(i,j_1) + c(j_1,j_3) \le c(i,j_1) + c(j_1,j_2) \le 2c(i,j_1) + c(i,j_2)$. Using this last inequality we get:

$$\begin{aligned}E[\sum_{j\in F} c(i,j)x_{ij}^I] &\le (a^2+2ab)c(i,j_1) + (b+ab)c(i,j_2)\\ &= a(a+2b)c(i,j_1) + b(1+a)c(i,j_2)\\ &= a(1+b)c(i,j_1) + b(1+a)c(i,j_2)\\ &\le (1+\max\{a,b\})(c(i,j_1)a + c(i,j_2)b)\\ &= (1+\max\{a,b\})\sum_{j\in F} c(i,j)\hat{x}_{ij}\end{aligned}$$

We are now ready to determine the performance ratio of the algorithm.

Theorem 7. *There is a deterministic 6-approximation algorithm for the metric k-median problem.*

Proof. Since algorithm ConvexCombination can be easily de-randomized using the method of conditional expectations, we only need to show that the expected cost of the solution x^I, y^I is at most 6 times the cost of an optimum solution for the k-median problem.

By the previous two lemmas, the cost of the solution x^I, y^I is at most $(3 + \frac{1}{n_f})(1 + \max\{a, b\})$ times larger than the optimum. By using basic algebra we can show that, $a = \frac{k_2-k}{k_2-k_1} \leq \frac{n_f-k}{n_f-(k-1)} \leq \frac{n_f-1}{n_f}$, and $b = \frac{k-k_1}{k_2-k_1} \leq \frac{k-1}{k}$. Then $1 + \max\{a, b\} \leq 1 + \frac{n_f-1}{n_f} \leq 2 - \frac{1}{n_f}$. Therefore, the performance ratio of the algorithm is $\left(3 + \frac{1}{n_f}\right)\left(2 - \frac{1}{n_f}\right) \leq 6$.

Charikar and Guha [12] proposed a slight variation of the above algorithm, and using a more complex analysis they were able to show that their algorithm has a performance ratio of 4.

5 A Local Search Algorithm

Local search heuristics have not been widely used to design approximation algorithms, mainly because of the difficulty of proving that a locally optimal solution is within a certain factor of the globally optimal one. Hence, it is surprising that a local search algorithm yields the best known performance ratio for the metric k-median problem. In this section we describe the algorithm of Arya et al. [5] which for any value $\varepsilon > 0$, achieves a performance ratio of $(3+2/p)/(1-\varepsilon n^2) \leq 3+2/p+\varepsilon'$ for $\varepsilon' = 10\varepsilon n^2$, if $\varepsilon \leq 1$ and $n \geq 2$. This algorithm repeatedly improves a solution by swapping p of the centers in the current solution with p facilities not in the solution, where the value of the parameter p is not larger than k.

Given a set S of at most k centers we define the cost of S, denoted as $c(S)$, as the total service costs of the clients, i.e. $c(S) = \sum_{d \in D} c(d, S)$, where $c(d, S) = \min\{c(d, f) \mid f \in S\}$ is the smallest cost of servicing d by one of the centers in S. Let ε be a constant value and $0 < \varepsilon < 1$. The algorithm of Arya et al. is the following.

Algorithm LocalSearch (p, ε)

```
1. $S^\ell \leftarrow$ an arbitrary set of $k$ centers
2. **while** there are sets $T \subseteq F \setminus S^\ell$ and $T' \subseteq S^\ell$ such that $|T| = |T'| \leq p$
           **and** $c((S^\ell \setminus T') \cup T) \leq (1-\varepsilon)c(S^\ell)$, **do**
       $S^\ell \leftarrow (S^\ell \setminus T') \cup T$
3. **return** $S^\ell$.
```

Let S^* be an optimum solution for the k-median problem. If p and ε are constant values, a straightforward implementation of the algorithm checks in every iteration of the while loop all subsets of $F \backslash S^\ell$ and S^ℓ of size at most p. Since there are $O(n^p)$ subsets of $F \setminus S^\ell$ (and of S^ℓ) of size at most p, and since the condition $c((S^\ell \backslash T') \cup T) \leq (1-\varepsilon)c(S^\ell)$ can be easily tested in $O(n)$ time, then each iteration of the while loop can be implemented to run in $O(n^{2p+1})$ time. In every iteration of the while loop the value of the solution decreases by at least a factor of $1-\varepsilon$, and so the maximum number of iterations is $\log_{\frac{1}{1-\varepsilon}} \frac{c(S_0)}{c(S^*)}$, where S_0 is the initial solution. Hence, the algorithm runs in time $O(n^{2p+1} \log(c(S_0)/c(S^*))/\log(\frac{1}{1-\varepsilon}))$.

Theorem 8. *Algorithm* LocalSearch *has a performance ratio of* $(3+2/p)/(1-\varepsilon)$.

In order to ensure that the above algorithm has polynomial running time, we must guarantee that each iteration of the **while** loop decreases the value of the current solution by at least a factor of $1-\varepsilon$. Therefore, the algorithm is not guaranteed to find a locally optimal solution. To prove Theorem 8 we first need to show some properties of the solution S^ℓ computed by the algorithm. For any sets $T \subseteq S^\ell$ and $T' \subseteq F \setminus S^\ell$ such that $|T| = |T'| \leq p$,

$$c((S^\ell \setminus T) \cup T') > (1-\varepsilon)c(S^\ell). \tag{13}$$

Given a feasible solution S for the k-median problem, we denote the set of clients serviced by some subset of centers $A \subseteq S$ as $N_S(A)$. Given a center $s \in S$, the set of clients served by s is denoted as $N_S(s)$.

Given a set of centers $A \subseteq S^\ell$, we say that A *captures* a center o belonging to the optimum solution S^*, if A serves at least half of the clients served by o, or in other words, if $|N_{S^\ell}(A) \cap N_{S^*}(o)| \geq |N_{S^*}(o)|/2$. We define *capture*$(A)$ as the set of centers $o \in S^*$ captured by A. For any two sets $X, Y \subseteq S^\ell$, the following properties hold.

Lemma 12. *If X and Y are disjoint, then* capture(X) *and* capture(Y) *are disjoint. Furthermore, if $X \subseteq Y$ then* capture$(X) \subseteq$ capture(Y).

Proof. To prove the first property we show that every center $o \in S^*$ can be captured by at most one of the sets X, Y. If a center $o \in S^*$ is captured by, say, X then more than half of the clients served by o in the optimum solution are served by centers in X. Therefore, the centers in Y cannot capture o.

The second property is easy to show since every center $o \in S^*$ captured by X has more than half of its clients served also by Y in S^ℓ.

We call a center $s \in S^\ell$ *bad* if it captures at least one center in S^*, and *good* otherwise.

Lemma 13. *The solutions S^ℓ and S^* can be partitioned into sets $A_1, A_2, \ldots, A_r$ and $B_1, B_2, \ldots, B_r$, respectively, where $r-1$ is the number of bad centers in S^ℓ. This partition is such that for all $i = 1, \ldots, r-1$, $|A_i| = |B_i|$, $B_i =$*capture(A_i)*, and A_i has one bad center. Furthermore, $|A_r| = |B_r|$ and A_r has only good centers.*

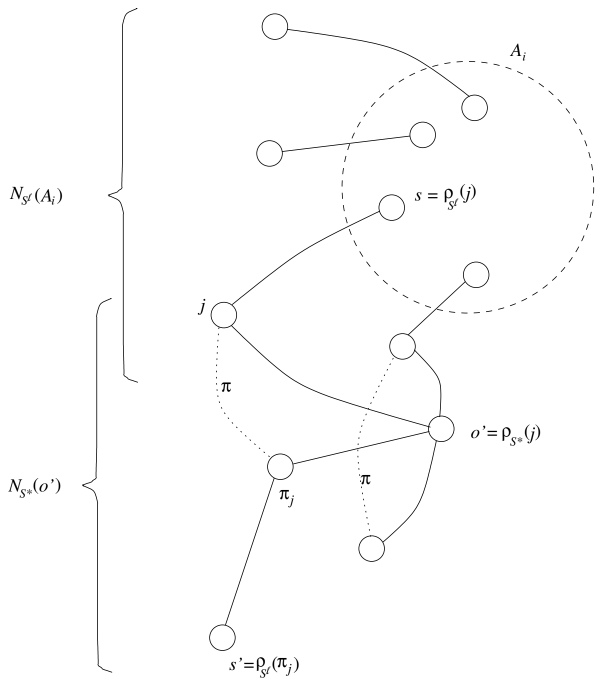

Fig. 2. Service cost of client j after reassigning clients to centers

Proof. We prove the lemma by induction on the number of bad centers. The basis is trivial because if the number of bad centers is 0 then we simply set $A_1 = S^\ell$ and $B_1 = S^*$. For the induction step, let us assume that the partitions exist when the number of bad centers in S^ℓ is b, $b \geq 0$, and prove that such a partition exists when the number of bad centers is $b + 1$. Choose a bad center $s \in S^\ell$, and set $A_1 = \{s\}$. Note that $capture(A_1) \geq 1$. If $|capture(A_1)| > |A_1|$, then repeatedly add good centers to A_1 until the size of A_1 is equal to the size of $capture(A_1)$. We can always do this since every bad center in S^ℓ captures at least one center in S^* and, hence, if $|capture(A_1)| > |A_1|$ there must be at least one good center in $S^\ell \setminus A_1$.

Set $B_1 \leftarrow capture(A_1)$. To complete the proof, we note that by the induction hypothesis it is possible to partition $S^\ell \setminus A_1$ and $S^* \setminus capture(A_1)$ into sets $A_2, \ldots, A_{k+2}$ and $B_2, \ldots, B_{k+2}$ as described in the statement of the lemma. By Lemma 12, the sets B_i are disjoint, so the sets A_i and B_i are partitions of S^ℓ and S^*, respectively. Finally, since $|S^\ell| = |S^*|$, then $|A_r| = |B_r|$.

For each client j let $\rho_{S^\ell}(j)$ be the center that serves j in solution S^ℓ and let $\rho_{S^*}(j)$ be the center that serves j in S^*. Now we are ready to prove Theorem 8.

Proof of Theorem 8. Let $\{A_1, A_2, \ldots, A_r\}$, $\{B_1, B_2, \ldots, B_r\}$ be partitions of S^ℓ and S^* as described above. To compute the performance ratio of the algorithm, let us consider the following sets of swaps involving all the centers in the optimum solution S^*.

i. For each set A_i such that $|A_i| = |B_i| \leq p$, swap the centers in A_i with those in B_i. By inequality (13) we know that

$$c((S^\ell \setminus A_i) \cup B_i) > (1 - \varepsilon)c(S^\ell).$$

We can bound the value of the left hand side of this inequality by reassigning the clients served by S^ℓ to centers in $(S^\ell \setminus A_i) \cup B_i$ as follows. All clients in $N_{S^*}(B_i)$ are assigned to the centers in B_i. For all other clients that are served by A_i we proceed as follows. Consider all clients j served by A_i that are served in S^* by some center $o' = \rho_{S^*}(i) \notin B_i$. Since $o' \notin B_i$, then A_i does not capture o' and, hence, $|N_{S^\ell}(A_i) \cap N_{S^*}(o')| < \frac{1}{2}|N_{S^*}(o')|$. Thus, for each one of these clients j we can associate a unique client $\pi_j \in N_{S^*}(o') \setminus N_{S^\ell}(A_i)$ (see Figure 2). Let π_j be served in S^ℓ by some center $s' = \rho_{S^\ell}(\pi_j) \notin A_i$. Then, by the triangle inequality, reassigning client j to center s' incurs a cost at most $c(\rho_{S^\ell}(\pi_j), j) \leq c(\rho_{S^\ell}(\pi_j), \pi_j) + c(\rho_{S^*}(j), \pi_j) + c(\rho_{S^*}(j), j)$. Hence,

$$\begin{aligned}
(1 - \varepsilon)c(S^\ell) &< c((S^\ell \setminus A_i) \cup B_i) \\
&\leq \sum_{j \in N_{S^*}(B_i)} c(\rho_{S^*}(j), j) + \\
&\quad \sum_{\substack{j \in N_{S^\ell}(A_i) \\ j \notin N_{S^*}(B_i)}} (c(\rho_{S^\ell}(\pi_j), \pi_j) + c(\rho_{S^*}(j), \pi_j) + c(\rho_{S^*}(j), j)) + \\
&\quad \sum_{\substack{j \in D \\ j \notin N_{S^*}(B_i) \cup N_{S^\ell}(A_i)}} c(\rho_{S^\ell}(j), j) \\
&\leq \sum_{j \in D} c(\rho_{S^\ell}(j), j) + \sum_{j \in N_{S^*}(B_i)} (c(\rho_{S^*}(j), j) - c(\rho_{S^\ell}(j), j)) + \\
&\quad \sum_{j \in N_{S^\ell}(A_i)} (c(\rho_{S^\ell}(\pi_j), \pi_j) + \\
&\quad c(\rho_{S^*}(j), \pi_j) + c(\rho_{S^*}(j), j) - c(\rho_{S^\ell}(j), j))
\end{aligned} \tag{14}$$

ii. For each set A_i such that $|A_i| = |B_i| = q > p$, we first select a set of $q - 1$ good centers from A_i. Then, we swap every center $o \in B_i$ with each one of the $q - 1$ selected good centers s from A_i. Proceeding similarly as above, we can show that

$$\begin{aligned}
(1 - \varepsilon)c(S^\ell) &< c((S^\ell \setminus \{s\}) \cup \{o\}) \\
&\leq \sum_{j \in D} c(\rho_{S^\ell}(j), j) + \sum_{j \in N_{S^*}(o)} (c(\rho_{S^*}(j), j) - c(\rho_{S^\ell}(j), j)) + \\
&\quad \sum_{j \in N_{S^\ell}(s)} (c(\rho_{S^\ell}(\pi_j), \pi_j) + c(\rho_{S^*}(j), \pi_j) + \\
&\quad c(\rho_{S^*}(j), j) - c(\rho_{S^\ell}(j), j))
\end{aligned}$$

By adding all inequalities for a center $o \in B_i$, and then dividing the sum by $q-1$ we get

$$\begin{aligned}(1-\varepsilon)c(S^\ell) < \sum_{j\in D} c(\rho_{S^\ell}(j),j) + \sum_{j\in N_{S^*}(o)} (c(\rho_{S^*}(j),j) - c(\rho_{S^\ell}(j),j)) +\\ \frac{1}{q-1}\sum_{j\in N_{S^\ell}(A_i)} (c(\rho_{S^\ell}(\pi_j),\pi_j) + c(\rho_{S^*}(j),\pi_j) +\\ c(\rho_{S^*}(j),j) - c(\rho_{S^\ell}(j),j)) \qquad (15)\end{aligned}$$

Next, we add the inequalities (15) for the q centers $o \in B_i$:

$$\begin{aligned}q(1-\varepsilon)c(S^\ell) < q\sum_{j\in D} c(\rho_{S^\ell}(j),j) + \sum_{j\in N_{S^*}(B_i)} (c(\rho_{S^*}(j),j) - c(\rho_{S^\ell}(j),j)) +\\ \frac{q}{q-1}\sum_{j\in N_{S^\ell}(A_i)} (c(\rho_{S^\ell}(\pi_j),\pi_j) + c(\rho_{S^*}(j),\pi_j) +\\ c(\rho_{S^*}(j),j) - c(\rho_{S^\ell}(j),j)) \qquad (16)\end{aligned}$$

Add the inequalities (14) and (16) for all sets A_i, B_i. Let r_1 be the number of pairs A_i, B_i for which $|A_i| = |B_i| \le p$. Since $\bigcup_{A_i} N_{S^\ell}(A_i) = \bigcup_{B_i} N_{S^*}(B_i) = D$, then by adding the inequalities we get

$$\begin{aligned}(r_1 + q(r-r_1))(1-\varepsilon)c(S^\ell) < (r_1 + q(r-r_1))\sum_{j\in D} c(\rho_{S^\ell}(j),j) +\\ \sum_{j\in D}(c(\rho_{S^*}(j),j) - c(\rho_{S^\ell}(j),j)) +\\ \frac{q}{q-1}\sum_{j\in D}(c(\rho_{S^\ell}(\pi_j),\pi_j) + c(\rho_{S^*}(j),\pi_j) +\\ c(\rho_{S^*}(j),j) - c(\rho_{S^\ell}(j),j)) \qquad (17)\end{aligned}$$

Because of the way in which the mapping π has been defined, it is not hard to see that

$$\sum_{j\in D}(c(\rho_{S^\ell}(\pi_j),\pi_j) + c(\rho_{S^*}(j),\pi_j) + c(\rho_{S^*}(j),j) - c(\rho_{S^\ell}(j),j)) \le 2c(S^*).$$

By using this last inequality in (17), we get

$$\begin{aligned}(r_1 + q(r-r_1))(1-\varepsilon)c(S^\ell) &< (r_1 + q(r-r_1))c(S^\ell) +\\ &\quad (c(S^*) - c(S^\ell)) + \frac{q}{q-1}(2c(S^*))\\ &\le \left(3 + \frac{2}{p}\right)c(S^*) + (r_1 + q(r-r_1) - 1)c(S^\ell)\end{aligned}$$

The last inequality follows from $q/(q-1) \leq (p+1)/p$. Since $r_1 + q(r - r_1) < n^2$, then

$$(1 - \varepsilon n^2)c(S^\ell) < (1 - \varepsilon(r_1 + q(r - r_1)))c(S^\ell) < \left(3 + \frac{2}{p}\right) c(S^*).$$

Thus, $c(S^\ell) \leq (3 + \frac{2}{p})/(1 - \varepsilon n^2)c(S^*)$. □

Acknowledgements

I kindly thank the anonymous referees for their very accurate and insightful comments.

References

1. A. Archer, R. Rajagopalan, and D. Shmoys, Lagrangian relaxation for the k-median problem: new insights and continuity properties, *Proceedings of the 11th Annual European Symposium on Algorithms*, 2003, LNCS 2832, pp. 31–42.
2. S. Ahn, A. Cooper, G. Cornuejols, and A. Frieze, Probabilistic analysis of a relaxation for the k-median problem, *Mathematics of Operations Research*, 13, 1988, pp. 1–31.
3. M. Andrews and L. Zhang, The access network design problem, *Proceedings of the 39th Annual IEEE Symposium on Foundations of Computer Science*, 1998, pp.40–59.
4. S. Arora, P. Raghavan, and S. Rao, Approximation schemes for Euclidean k-medians and related problems, *Proceedings of the 30th Annual ACM Symposium on Theory of Computing*, 1998, pp. 106–113.
5. V. Arya, N. Garg, R. Khandekar, A. Meyerson, K. Munagala, and V. Pandit, Local search heuristics for k-median and facility location problems, *Proceedings of the ACM Symposium on Theory of Computing*, 2001, pp. 21–29.
6. M.L. Balinski, On finding integer solutions to linear programs, *Proceedings of the IBM Scientific Computing Symposium on Combinatorial Problems*, 1966, pp. 225-248.
7. Y. Bartal, Probabilistic approximation of metric spaces and its algorithmic applications, *Proceedings of the 37th Annual IEEE Symposium on Foundations of Computer Science*. 1996, pp. 184–193.
8. Y. Bartal, On approximating arbitrary metrics by tree metrics, *Proceedings of the 30th Annual ACM Symposium on Theory of Computing*, 1998, pp. 161–168.
9. P.S. Bradley, U.M. Fayad, and O.L. Mangasarian, Mathematical programming for data mining: formulations and challenges, Microsoft Technical Report, 1998.
10. A.F. Bumb and W. Kern, A simple dual ascent algorithm for the multilevel facility location problem, *Proceedings of RANDOM-APPROX*, 2001, pp. 55-62.
11. M. Charikar, C. Chekuri, A. Goel, and S. Guha, Rounding via trees: deterministic approximation algorithms for group Steiner trees and k-median, *Proceedings of the 30th Annual ACM Symposium on Theory of Computing*, 1998, pp. 114–123.
12. M. Charikar and S. Guha, Improved combinatorial algorithms for the facility location and k-median problems, *Proceedings of the 40th Annual Symposium on Foundations of Computer Science*, 1999, pp. 378–388.

13. M. Charikar, S. Guha, E. Tardos, and D.B. Shmoys, A constant-factor approximation algorithm for the k-median problem, Proceedings of the Thirty-First Symposium on Theory of Computing, 1999, pp. 1–10.
14. J. Cheriyan and R. Ravi, *Approximation algorithms for network problems*, Lecture Notes, University of Waterloo, 1998.
15. F. Chudak and D. Williamson, Improved approximation algorithms for capacitated facility location problems, *Proceedings of the 7th Conference on Integer Programming and Combinatorial Optimization*, 1999.
16. V. Chvátal, A greedy heuristic for the set covering problem, *Mathematics of Operations Research*, 4, 1979, pp. 233–235.
17. V. Chvátal, *Linear Programming*, W.H. Freeman and Company, 1983.
18. G. Cornuejols, G.L. Nemhauser, and L.A. Wolsey, The uncapacitated facility location problem, in P. Mirchandani and R. Francis, eds. *Discrete Location Theory*, John Wiley and Sons, New York, 1990.
19. S. Guha, A. Meyerson, and K. Munagala, Hierarchical placement and network design problems, *Proceedings of the 14th Annual IEEE Symposium on Foundations of Computer Science*, 2000.
20. S. Guha, A. Meyerson, and K. Munagala, Improved combinatorial algorithms for single sink edge installation problems, Technical Report STAN-CS-TN00-96, Stanford University, 2000.
21. S. Guha, A. Meyerson, and K. Munagala, Improved algorithms for fault tolerant facility location, *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms*, 2001, pp. 636–641.
22. S. Jamin, C. Jin, Y. Jin, D. Raz, Y. Shavitt, and L. Zhang, On the placement of internet instrumentations, *Proceedings of IEEE INFOCOM*, 2000, pp. 26–30.
23. K. Jain and V.V. Vazirani, Approximation algorithms for metric facility location and k-median problems using the primal-dual schema and Lagrangian relaxation, *Proceedings of the 40th Annual Symposium on Foundations of Computer Science*, 1999. Also, *Journal of the ACM*, 48, 2001, pp. 274–296.
24. D.S. Johnson, Approximation algorithms for combinatorial problems, *Journal of Computer and System Sciences*, 9, 1974, pp. 256–278.
25. O. Kariv and S.L. Hakimi, An algorithmic approach to network location problems, Part II: p-medians, *SIAM Journal on Applied Mathematics*, 1979, pp. 539–560.
26. O. Kariv and S.L. Hakimi, An algorithmic approach to network location problems. II: The p-medians, *SIAM Journal on Applied Mathematics*, 1979, pp. 539–560.
27. L. Khachiyan, A polynomial algorithm for linear programming, *Doklady Akad. Nauk USSR*, 244 (5), 1979, pp. 1093–1096.
28. S. Khuller, R. Pless, and Y.J. Sussman, Fault tolerant k-center problems, *Theoretical Computer Science*, 242, 2000, pp. 237–245.
29. A.A. Kuehn and M.J. Hamburger, A heuristic program for locating warehouses, *Management Science*, 9, 1963, pp. 643–666.
30. M. R. Korupolu, C. G. Plaxton, and R. Rajaraman. Analysis of a local search heuristic for facility location problems, *Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms*, 1998, pp. 1-10. Also in *Journal of Algorithms* 37, 2000, pp. 146-188.
31. P. Krysta and R. Solis-Oba, Approximation algorithms for bounded facility location, *Journal of Combinatorial Optimization*, 5, 2001, pp. 2–16.
32. B. Li, M. Golin, G. Italiano, X. Deng, and K. Sohraby, On the optimal placement of web proxies in the internet, *Proceedings of IEEE INFOCOM* 1999, pp. 1282–1290.

33. J. Lin and J.S. Vitter, ϵ-Approximations with minimum packing constraint violation, *Proceedings of the 24th Annual ACM Symposium on Theory of Computing*, 1992, pp. 771–782.
34. J. Lin and J.S. Vitter, Approximation algorithms for geometric median problems, *Information Processing Letters*, 44, 1992, pp. 245–249.
35. L. Lovász, On the ratio of optimal integral and fractional covers, *Discrete Mathematics*, 13, 1975, pp. 383–390.
36. N. Megiddo and K.J. Supowit, On the complexity of some common geometric location problems, *SIAM Journal on Computing*, 13, 1984, pp. 182–196.
37. M. Mahdian and M. Pál, Universal facility location, *Proceedings of the 11th Annual European Symposium on Algorithms*, 2003, LNCS 2832, pp. 409–421.
38. M. Mahdian, Y. Ye, and J. Zhang, Improved approximation algorithms for metric facility location problems, *Proceedings of the 5th International Workshop on Approximation Algorithms for Combinatorial Optimization*, 2001, LNCS 2462, pp. 229–242.
39. R. R. Mettu and C. G. Plaxton, The online median problem, *SIAM Journal on Computing*, 32, 2003, pp. 816-832.
40. J.M. Mulvey and H.L. Crowder, Cluster Analysis: an application of Lagrangian relaxation, *Management Science*, 25, 1979, pp. 329–340.
41. K. Murty, *Linear Programming*, John Wiley & Sons, 1983.
42. G.L. Nemhauser and L.A. Wolsey, *Integer and Combinatorial Optimization*, John Wiley and Sons, New York, 1990.
43. C.H. Papadimitriou, Worst case and probabilistic analysis of a geometric location problem, *SIAM Journal on Computing*, 10 (3), 1981, pp. 542–557.
44. L. Qiu, V.N. Padmanabhan, and G. Voelker, On the placement of web server replicas, *Proceedings of IEEE INFOCOM*, 2001.
45. R.Raz and S. Safra, A sub-constant error-probability low-degree test, and sub-constant error-probability PCP characterization of NP, *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*,1997, pp. 475–484.
46. A. Schrijver, *Theory of Linear and Integer Programming*, John Wiley & Sons, 1986.
47. A. Tamir, An $O(pn^2)$ algorithm for the p-median and related problems on tree graphs, *Operations Research Letters*, 19, 1996, pp. 59–94.
48. Y. Ye, An $O(n^3L)$ potential reduction algorithm for linear programming, *Mathematical Programming*, 50, 1991, 239–258.
49. J. Zhang, Approximating the two-level facility location problem via a quasi-greedy approach, *Proceedings of the 15th Annual ACM-SIAM Symposium on Discrete Algorithms*, 2004, 801-810.