# Approximation schemes for Euclidean $k$-medians and related problems

Sanjeev Arora*
Princeton University

Prabhakar Raghavan†
IBM Research

Satish Rao‡
NEC Research

**Abstract**

In the *k-median* problem we are given a set $S$ of $n$ points in a metric space and a positive integer $k$. We desire to locate $k$ *medians* in space, such that the sum of the distances from each of the points of $S$ to the nearest median is minimized. This paper gives an approximation scheme for the plane that for any $c > 0$ produces a solution of cost at most $1 + 1/c$ times the optimum and runs in time $O(n^{O(c+1)})$. The approximation scheme also generalizes to some problems related to $k$-median.

Our methodology is to extend Arora's [1, 2] techniques for the TSP, which hitherto seemed inapplicable to problems such as the $k$-median problem.

## 1 Introduction

In the *k-median* problem we are given a set $S$ of $n$ points in a metric space and a positive integer $k$. We desire to locate $k$ *medians* in the space, such that the sum of the distances from each of the points of $S$ to the nearest median is minimized. Besides its intrinsic appeal as a cleanly-stated, basic unsolved problem in combinatorial optimization, the $k$-median problem has applications to pattern classification and data mining (see for instance [9, 12] and references therein). In fact it is a prototypical *clustering* problem for $\Re^2$.

Building on methods due to Arora [1, 2] for the traveling salesman problem, we give in this paper the first constant-factor approximation algorithm for the $k$-median problem in the plane; in fact, we give a polynomial-time approximation scheme. We note that this application of Arora's techniques is quite surprising, since he mentions [2] that his techniques seem to apply only to problems in which (a) the objective function is a sum of edge lengths (b) the patching lemma holds (i.e., there is a solution of cost $c$ that crosses a certain length $l$ line segment more than twice, then there is a solution of cost $c + O(l)$ that crosses the line segment only twice). The $k$-median problem satisfies property (a) but not property (b). We develop therefore (in Section 2) a general version of his charging lemma that seems useful for problems for which the patching lemma fails. Further, our techniques extend to some cases where property (a) is also violated. As a by-product of our methods, we derive a polynomial-time approximation scheme for the *facility location problem* in the plane; this is defined below.

Since the Patching Lemma does not hold for most geometric problems, we believe that our delinking of the charging scheme from the Patching Lemma could be an important tool in the design of approximation algorithms for other geometric problems. We note that our charging argument — just like Arora's argument — is non-local (in contrast to many existing geometric algorithms, which are analyzed by charging additional cost to "nearby" edges), and amortizes cost over a very large neighborhood.

**Related prior work:** A problem related to $k$-median is the $k$-center problem, in which we wish to minimize the maximum distance of any point to a facility. This and related *minmax*

*Supported by NSF CAREER award NSF CCR-9502747, an Alfred Sloan Fellowship, and a Packard Fellowship. Email: arora@cs.princeton.edu

†IBM Almaden Research Center, 650 Harry Road, San Jose CA 95120.

‡NEC Research Institute, Princeton, NJ.

*clustering* problems are relatively well-understood. They can be approximated (typically using simple greedy heuristics) within factors close to 2, and achieving a substantially better factor is NP-hard. (See [4] for a survey.)

The *facility-location problem* is somewhat similar to the $k$-median problem: we are given a set $S = \{x_1, \ldots, x_n\}$ of $n$ points again, together with a cost $c_i$ for opening a facility at $x_i$. We are to find a set $F$ of facilities so as to minimize the sum of the distances from each of the points of $S$ to the nearest facility, *plus* the cost of opening the facilities. Hochbaum [7] showed that the greedy algorithm is an $O(\log n)$-approximation algorithm for this problem for the uncapacitated problem. Shmoys, Tardos and Aardal [13] give a 3.16 approximation algorithm for arbitrary metric spaces; this factor has since been improved to 2.41 by Guha and Khuller [6], and more recently to 1.74 by Chudak [5].

Whereas the facility-location problem (where the number of facilities is variable) has succumbed to approximation, the $k$-median problem on the other hand has defied the development of good approximation algorithms, apparently because we are only allowed to use $k$ facilities (and no more). Indeed, the only approximation algorithm we know of for the $k$-median problem stems from a combination of techniques of Bartal [3] together with an (exact) algorithm for the $k$-median problem on trees due to Hochbaum [7], yielding an approximation ratio that is somewhat bigger than $\log n$ for general metrics. Lin and Vitter [10] use an elegant technique called *filtering* for rounding fractional solutions of linear programming relaxations to the $k$-median problem. This results in an algorithm for the $k$-median problem achieving a cost within a factor of $(1+\epsilon)$ of optimal, but using $(1+1/\epsilon)(\ln n + 1)k$ median locations. Lin and Vitter [11] also gave an algorithm achieving a solution that is $2(1+\epsilon)$ times the optimum, but using at most $(1+1/\epsilon)k$ median locations. All of these results together suggest that the $k$-median problem's resistance to approximation stems from our insistence on using $k$ medians and not more.

### 1.1 Clear statement of problems and results

**$k$-median problem:** Given $n$ points $\{x_1, \ldots, x_n\}$ in a metric space with a distance metric $d()$ and a positive integer $k$, find a set of $k$ medians $M = \{m_1, \ldots, m_k\}$ in the space so as to minimize $\sum_{i=1}^{n} \min_{1 \le j \le k} d(x_i, m_j)$.

**Facility location problem:** Given $n$ points $x_1, \ldots, x_n$ in a metric space with a distance metric $d()$, and a *cost* $c_i$ for opening a facility at $x_i$, determine a subset $F$ of $\{x_1, \ldots, x_n\}$ at which to place facilities so as to minimize

$$\sum_{x_\in F} c_i + \sum_{i=1}^{n} \min_{x_j \in F} d(x_i, x_j).$$

A **d dimensional geometric instance** of the $k$-median problem or (facility location problem) is a set of points that correspond to points in $d$-dimensional space with Euclidean distance as a metric.

**Theorem 1** *Given a 2-dimensional geometric instance of the $k$-median problem and a positive real $c$, the algorithm developed in Section 3.3 achieves with probability $1 - o(1)$ a solution of cost at most $(1 + 1/c)$ times the optimal cost in time $n^{O(c+1)}$.*

**Theorem 2** *Given a 2-dimensional geometric instance of the facility location problem and a positive real $c$, the algorithm developed in Section 3.3.2 achieves with probability $1 - o(1)$ a solution of cost at most $(1 + 1/c)$ times the optimal cost in time $n^{O(c+1)}$.*

Our algorithms are randomized, and so the probabilities in all our theorems are over the random choices of the algorithm (and hold for any input). It is easy to restate these results as guaranteeing a $(1+1/c)$-approximate solution in expected time $n^{O(c+1)}$. The algorithm hinges on a charging algorithm that generalizes the techniques of Arora [1, 2], and should be of independent interest for other geometric optimization problems; this argument is developed in Section 2.1.

Also in section 3.3 we describe quasi-polynomial time algorithms for $d$-dimensional geometric instances and for a capacitated versions of the problems.

### 1.2 Our Techniques

A *quadtree* is a geometric division of the plane into a hierarchy of square regions (see Section 2). Arora [2] uses a randomized variant of the quadtree. Using the Patching Lemma, he proves that the optimum TSP tour can be modified —with a $1 + \epsilon$ factor increase in cost— such that the

number of tour edges crossing any region in the quadtree becomes $O(1/\epsilon)$. (Note that this is a surprising result, since *a priori*, the number of crossing edges could be a large function of the number of nodes, say $\sqrt{n}$.) In particular it follows that there *exists* a tour of cost at most $(1+\epsilon)OPT$ that crosses every region of the quadtree at most $O(1/\epsilon)$ times. A simple dynamic programming (whenever you divide a quadtree square into four, "guess" the tour interface between the four subsquares and then recur independently in the subsquares) can find such a tour.

We use a similar quadtree-based algorithm. The main difference is that the Patching Lemma does not hold for our problems here, so we cannot in general reduce the number of edges that cross the boundary of any quadtree square. Nevertheless, we can generalize Arora's charging argument — which does not use the Patching Lemma — to argue that the there exists a $(1+\epsilon)$-approximate solution that has a very restricted "interface" between any adjacent squares of the quadtree. (Specifically, the interface belongs to a set of $n^{O(1/\epsilon)}$ interfaces, all of which are considered by our algorithm.) Thus we can use dynamic programming to find such a solution.

## 2 Quadtree dissection and the charging argument

This section describes the quadtree dissection and an associated charging argument that applies to any set of line segments in the plane. The dissection is the same as in [2], and the charging argument is a general statement of some of the calculations in [2].

Let the *bounding box* of a set of points be the smallest square containing it. Let $L$ denote its length.

A *dissection* of the bounding box is a recursive partitioning into smaller squares. We view it as a 4-ary tree whose root is the bounding box. Each square in the tree is partitioned into four equal squares, which are its children. We stop partitioning a square if it has size $\leq 1$ (and therefore at most one node). Note that there are $O(L^2)$ nodes in the tree and its depth is $\log L$. A *quadtree* is defined similarly, except we stop the recursive partitioning as soon as the square has at most one input point. The quadtree may in general have fewer squares than the dissection; see Figure 1.

If $a, b$ are integers in $[0, L)$, then the $(a, b)$-*shift* of the dissection is defined by shifting the $x$- and $y$- coordinates of all lines by $a$ and $b$ respectively, and then reducing modulo $L$. In other words, the middle vertical line of the dissection is moved from the $x$-coordinate $L/2$ to the $x$-coordinate $a + L/2 \bmod L$, and the middle horizontal line from the $y$-coordinate $L/2$ to the $y$-coordinate $b + L/2 \bmod L$. The rest of the dissection is then "wrapped-around," so that the left edge of the dissection comes to rest at the $x$-coordinate $a$, and the lower edge of the dissection comes to rest at the $y$-coordinate $b$ (see Figure 2). Note that we treat a "wrapped-around" square in the shifted dissection as a single region; this will greatly simplify notation later. The reader can also think of a wrapped-around square as a disjoint union of 2 (or 4) rectangles.

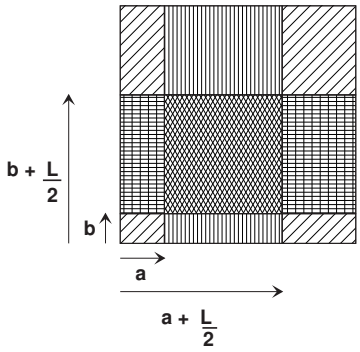
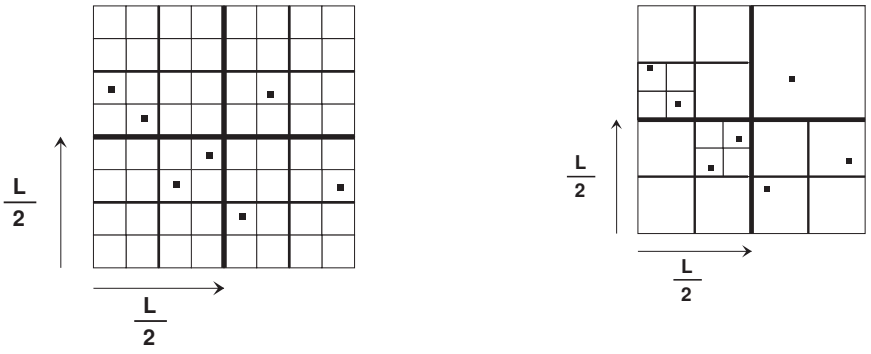



Figure 2: Dissection with shift $(a, b)$. Only the four children of the root are shown; each is identified by a distinctive shading. Note that three of the children are "wrapped-around" squares.

The *quadtree with shift* $(a, b)$ is obtained from the corresponding shifted dissection by cutting off the partitioning at squares that contain only one input point. It is easy to see that in general the shifted quadtree has a very different structure than the original quadtree.

### 2.1 The charging argument

Now we describe our charging argument. Suppose we have a collection $S$ of line segments in the plane. Suppose the minimum length of a segment in $S$ is at least 4 units, and the sum of lengths is $\text{cost}(S)$. Consider a shifted dissection with random shifts $a, b$. The charging argument looks at the number of times $S$ crosses various squares in the dissection, and allows some costs to be charged in proportion to this number. Our main lemma bounds from above the




(a) (b)

Figure 1: (a) The dissection. (b)The corresponding quad tree (Figures from [2].)

total cost thus charged as a fraction of cost$(S)$.

Let us place a unit grid in the plane, and for a grid line $l$ we denote by $t(S, l)$ the number of lines in $S$ that cross $l$. The next lemma follows by averaging.

**Lemma 3 ([2])** *If the minimum length of a segment in $S$ is at least 4 units, then $\sum_l t(S, l) = \Theta(cost(S))$, where the summation is over all lines in the unit grid.*

Consider a shifted dissection with shifts $a, b$. Its squares form a hierarchy, and so have a natural notion of "level"(the bounding box is at level 0, its four children are the squares at level 1, and so on). Thus each edge of the dissection has an associated *level*: simply the level of the square it bounds. We say that an edge is *maximal* if it is not part of any edge higher in the hierarchy.

We also define a notion of level for grid lines. A grid line $l$ has *level* $i$ in the shifted dissection if it contains a level $i$ edge. Note that a level $i$ edge gets subdivided to yield the edges of two level $i + 1$ squares, so a line that is at level $i$ is also at level $j$ for all $j > i$. For each $i \geq 1$ there are $2^i$ horizontal lines and $2^i$ vertical lines at level $i$. The vertical lines have $x$-coordinates $a + p \cdot L/2^i \bmod L$, where $p = 0, \ldots, 2^i - 1$ and the horizontal lines have $y$-coordinates $b + p \cdot L/2^i \bmod L$ , where $p = 0, \ldots, 2^i - 1$. The *maximal level* of a line is the highest level it is at.

Since the horizontal shift $a$ is chosen randomly, we have for each vertical line $l$ in the grid, and each $i \leq \log L$,

$$\Pr_a[l \text{ is at level } i] = \frac{2^i}{L}. \qquad (1)$$

A similar statement is true for horizontal lines.

For any integer $R > 0$, we describe an *$R$-charging process* as follows. For each dissection edge that is maximal, if $k$ is the number of times this edge is crossed by line segments in $S$, then the process is allowed to charge a cost that is

$$\frac{k}{R} \times \text{length of the edge.}$$

**Lemma 4 (Charging Lemma)** *Suppose an $R$-charging process is allowed to charge costs only for edges from the top $i$ levels. Then if the shifts $a, b$ were chosen randomly, the expected total cost (over the random choices $a, b$) charged with shifts $a, b$ is $O(i \ cost(S)/R)$.*

**Proof:** Let $l$ be any line of the unit grid. We show that the expectation (over the random choices $a, b$) of the cost charged to edges of $l$ when the shifts are $a, b$ is at most $it(S, l)/R$, whence the lemma follows by linearity of expectations and Lemma 3.

We show this only when $l$ is a horizontal line (the case of vertical lines is similar). If the maximal level of $l$ turns out to be $j$, then every maximal edge on it has length at most $L/2^j$, so the process can charge at most a cost $(t(S, l)/R)(L/2^j)$. The probability that $l$ is at level $j$ is $2^j/L$, so the expected charge to $l$ is at most

$$\sum_{j \leq i} \frac{2^j}{L} \cdot \frac{t(S, l)}{R} \cdot \frac{L}{2^j} \qquad \leq \ \frac{i}{R} t(S, l).$$

■

Typically, the charging process is allowed to charge for $O(\log n)$ levels. Then we make $R = O(\log n/\epsilon)$, so the total cost charged is bounded

from above by $\epsilon \cdot \text{cost}(S)$. In other words, each time a maximal edge is crossed by $S$, we can charge a cost as high as $\epsilon/\log n$ times the edge length.

## 3 Structure theorem and algorithms

Now we prove theorems for the various problems that show the existence of approximately optimal solutions with a simple structure. The design of algorithms is then easy.

### 3.1 The $k$-medians problem

To begin with we assume that all nodes lie on the unit grid, the minimum nonzero internode distance is $\geq 1$ and the maximum internode distance is $O(n^4)$. (This is without loss of generality; see Section 3.2.3.) This means that the the leaf squares of the dissection contain at most one point, and so can be treated as the (trivial) base case of our dynamic programming algorithm.

We define the notion of an $m$-light solution to facility location. First, we note that a solution consists of collections of line segments, where the lines in each collection each form a star. We will allow the segments forming the spokes of each star to bend and pass through a set of prespecified points called *portals*.

**Definition 1** Let $m$ be a positive integer. An *$m$-regular set of portals* for a shifted dissection is a set of points on the edges of the squares in it. Each square has a portal at each of its 4 corners and $m$ other equally-spaced portals on each edge.

An *$m$-light* solution for the facility location problem is a solution in which whenever an edge passes the boundary of any square of the dissection, it does so through a portal.

**Lemma 5** *Let $m > 1$ be any integer and shifts $0 \leq a, b \leq L$ be picked randomly. Then for any facility location problem, with probability at least $1/2$, there is an $m$-light solution with respect to the dissection with shift$(a, b)$ with cost at most $(1 + O(\log L/m))$ times the optimal solution, whose value we denote by OPT.*

**Proof:** Consider the optimal facility location solution. This consists of a set of line segments (forming a set of stars). To make a solution $m$-light, we need only deflect each edge so that whenever it crosses a side of a square in the dissection, it does so through a portal. We assume wlog that $m$ is a power of 2, so a portal of a square is a portal of every smaller square that shares a boundary with the square. Thus we do not need to deflect the edge in more than one direction for that square.

Note that if the length of the side of the square is $l$ then we need to deflect an edge by at most $l/m$ to make it pass through a portal. We charge this cost to the corresponding side of the square.

Note that we have just described an $m$-charging process! Since the number of levels in the dissection is $L$, Lemma 4 implies that the expected cost charged throughout the process is $O(\log L/m)OPT$. Hence the result is proved. ■

Since $L = O(n^4)$, we can make $m = O(\log n/\epsilon)$ and the lemma then guarantees an $m$-light solution of cost $(1 + \epsilon)OPT$.

### 3.2 The Dynamic Program

In the previous section, we showed that one need only find the optimal facility location solution where each edge is bent so that it passes only through portals.

In this section, we present a dynamic programming approach that approximates the optimal $m$-light solution to a facility location problem to within a factor of $(1 + 1/4m)$. We conclude using Lemma 5, that it is an $(1+O(\log L/2m))$ approximation algorithm for the facility location problem.

In the discussion below, we will refer to edges or portions of edges that are bent to pass through portals as $m$-light paths.

#### 3.2.1 The Tables

We will solve a set of subproblems on each square. A solution to a subproblem on a square is an assignment of the nodes in the square to either facilities in the square or to the portals of the square. The cost of a solution depends on how close each portal is to a facility. Thus, we "guess" (by enumeration during the dynamic program) how far each portal is from a facility.

It is easy to see that it suffices to "guess" the distance to the nearest facility only approximately, i.e., to within an $(1 + 1/4m)$ factor. Furthermore, the distance to the nearest facility is "not too different" among neighboring portals, so we can represent some of the distances as offsets from the previous distance. All of these ideas put together allows us to make

do with "guessing" only $O(m)$ bits of distance per square, so the dynamic programming can go through all guesses in $2^{O(m)}$ time.

Formally, an instance of the problem is specified by the following inputs.

- A nonempty square in the shifted quadtree.
- An integer $f \in [0, k]$
- If $f \neq 0$, an assignment *inside* of numbers $[1, ...4m]$ to the portals of the box, such that $|inside(p) - inside(p')| \leq 1$ if portal $p$ and $p'$ are successive portals along the bounding box.
- If $f < k$, an assignment *closest* of numbers $[1, ...4m]$ to the portals of the box, such that $|closest(p) - closest(p')| \leq 1$ if portal $p$ and $p'$ are successive portals along the bounding box, and $closest(p) < inside(p)$.

When $f \neq 0$, the goal of the subproblem for a box of side length $s$, is to identify a set of $f$ facilities and an $m$-light path connecting each node to either a facility or to a portal satisfying the following properties.

- If $f > 0$, each portal $p$ is within distance $inside(p)(s/m)$ of a facility.
- For a portal $p$, let $n(p)$ be the number of nodes that are connected to $p$ by the solution. The total length of the $m$-light paths plus

$$\sum_{portals\ p} closest(p)n(p)(s/m)$$

  is minimum.

The dynamic programming builds a lookup table containing the costs of the optimal solutions to all instances of the problem above arising in the quadtree.

If there is an $m$-light solution with cost $C$, then we argue that the dynamic program finds a solution $\mathcal{S}$ of cost $(1 + 1/4m)C$. The solution of cost $C$ gives rise to a single "interface" for each non-empty square. That is, each non-empty square gets a corresponding *closest* (and possibly *inside*) function. In trying all possible interfaces the algorithm will hit upon an interface in which all distances are guessed correctly to within a factor $(1 + 1/4m)$. Thus, the table entry of this subsquare is able to assign the facilities in this subsquare with cost at most $(1 + 1/4m)$ times the cost of assigning these facilities in the optimum. Arguing similarly about all squares we conclude that the dynamic program will produce a solution of cost at most $(1 + 1/4m)C$ to the $m$-light facility location problem.

The number of entries in the lookup table is just the number of different instances of the subproblem in the shifted quadtree. For each square the number of entries is bounded by $T = k((4m)(3)^{4m})^2$, since there are $k$ possible choices for the number of facilities in the square, and there are $(4m)3^{4m}$ possible choices for the inside and outside functions.

### 3.2.2 Computing the table

For a square $S$, the solution to the subproblem for $f = 0$, and *closest* corresponds to assigning each node $u$ in $S$ to the portal $p$ with minimum $dist(u, p) + closest(p)(s/m)$. (Technically, $dist(u, p)$ should be defined according to the shortest $m$-light path between $u$ and $p$. The algorithm works either way.)

The remainder of the table is built up in a bottom-up fashion. At the leaves, make a table entry with $f = 1$ with cost zero and with the inside function computed on the portals according to the distance to the single facility inside. The entries for other inside functions are undefined.

Inductively, suppose the algorithm has solved all the subproblems at depth $i + 1$ and let $S$ be any other square at depth $i$ with side length $s$. Let $S_1, S_2, S_3, S_4$ be its four children in the quadtree.

For every choice of (b) with $f > 0$, (c) and (d) for $S$, the algorithm enumerates all combinations of the defined table entries for its children such that

- the sum of the value of $f$ over the four subproblems equals the value of $f$ for this entry for $S$,
- for each portal $p$ of $S$, there is a portal $p'$ of one of its children where $dist(p, p') + inside(p')(s/2m) \leq inside(p)(s/m)$,
- and for each portal $p$ of $S_1, S_2, S_3, S_4$ there is
  - either a portal $p'$ of $S_1, S_2, S_3, S_4$ such that $dist(p, p') + (s/2m)inside(p') \leq closest(p)s/2m$

– or a portal $p'$ of $S$ such that $dist(p, p')+(s/m)closest(p') \leq closest(p)(s/2m)$.

It then chooses the one with minimum cost as the entry. If no such way is defined, then the entry is left undefined. With this description, each entry requires the examination of at most all four-tuples of table entries for the four sub squares. Thus, the time to fill in all the table entries for a square is $O(T^5)$. (One could easily improve this time, but this is not our focus here.) Since the total number of nonempty squares is $N \log N$, the total running time is $O(N \log N T^5)$.

By choosing $m = O(\log L/c)$, we obtain an algorithm that finds an $(1+1/c)$ times optimal solution in time $N^{O(1)}N^{O(c)}$ since $T = N^{O(c)}$ if $L$ is polynomial in $N$.

#### 3.2.3 Initial Perturbation

Now we indicate why the input can be assumed to be on integral points of an $L \times L$ grid, where $L$ is polynomial in $n$. We can ensure this as follows. First we use the algorithm for minmax clustering [4] to find a facility assignment that minimizes the *maximum* distance to a facility to within a factor of two, say this distance is $D$. We know that the optimal facility location value is at least $D/2$ and at most $Dn$. Using arguments similar to those in [2], it is easy to see how to decompose the problem into problems that can be enclosed in disjoint squares of size $2Dn^2$.

Then we round each coordinate of each point to the nearest multiple of $D/n^2$. This increases the cost of the resulting solution by at most $O(D/n)$, which is a small fraction of the optimal. By scaling, we obtain problems where the points are on integer coordinates and that are encoded by boxes with side length $O(n^4)$.

### 3.3 Related Facility Problems.

#### 3.3.1 Higher Dimensions

For $d$-dimensional instances of the $k$-medians problem we can provide quasi-polynomial time $(1+\epsilon)$-approximation algorithm as follows. The quadtree decomposition is easily generalized so that the space is recursively divided into hyperboxes by hyperplanes. We can then show that there exists a $(1+\epsilon)$-approximate solution that is $m^{d-1}$-light with respect to any hyperplane that is used in the quadtree decomposition for $m = O(\log n/\epsilon)$.

We proceed with a dynamic program analogous to that in section 3.2 that requires $n^O(m^{d-1})$ time.

The running time of the DP is $n^{O((\log n/\epsilon)^{d-1})}$.

#### 3.3.2 Facility Location

We can find a $(1+\epsilon)$ times optimal solution to the facility location problem using our approach as follows. Consider an optimal solution to the facility location problem. Say the facility cost of the solution is $F$ and the service cost is $C$.

We then use a dynamic programming approach that follows the same structure as our $k$-median algorithm and produces a solution with facility cost $(1+1/N)F$ and service cost $(1+1/c)C$ in $N^{O(1/\epsilon)}$ time with probability $1/2$.

#### 3.3.3 Algorithms for few medians.

A alternative dynamic program can be based on approximating the position of the facilities rather than the distances between portals and facilities. For this approach, each subproblem consists of choosing $k$ positions from an $m \times m$ regular spaced grid placed inside the square. This results in dynamic programs with table size $\binom{m^2}{k}$.

We can thus find an $(1+\epsilon)$ approximation algorithm for the $k$-median algorithm with running time $N(O(\log N/\epsilon))^{2k}$. This is better than our previous algorithms when $k$ is small.

#### 3.3.4 Extension to Capacitated $k$-medians

The capacitated $k$-medians problem is a $k$-medians problem where we are given an integer $B$ such that at most $B$ points can be assigned to any facility. The structure lemma 5 for the $k$-medians problem applies directly to this problem. The dynamic programming approach above, however, does not ensure that the capacity bounds are met. We can use an alternative dynamic programming formulation where the subproblems on a square enumerate partitions of excess capacity available at approximate locations throughout the region. As in the previous subsection the approximate locations are specified by positions in an $m \times m$ grid for each square where $m$ is $O(\log n/\epsilon)$.

This leads to an $N^{O(\log n/\epsilon)^2}$ time algorithm for finding an $(1+\epsilon)$-approximate solution for this problem.

## 4 Further work

One of our contributions here has been the decoupling of the charging scheme from the Patching lemma of Arora [1]. This raises the prospect of applying these methods to other difficult geometric optimization problems such as the *Minimum-weight Steiner triangulation* problem: given $n$ points in the plane, form a triangulation of minimum total length whose vertices include the $n$ given points and possibly additional "Steiner" points. Intriguingly, for this problem we do not even know whether there is a near-optimal solution with a small (even polynomially-bounded number of Steiner points).

## References


[1] S. Arora. Polynomial-time approximation schemes for Euclidean TSP and other geometric problems. *Proceedings of 37th IEEE Symposium on Foundations of Computer Science*, pp. 2-12, 1996.

[2] S. Arora. Nearly linear time approximation schemes for Euclidean TSP and other geometric problems. *Proceedings of 38th IEEE Symposium on Foundations of Computer Science*, pp. 554–563, 1997.

[3] Y. Bartal. Probabilistic approximation of metric spaces and its algorithmic applications. *Prof. 37th IEEE FOCS,* 1996.

[4] M. Bern and D. Eppstein. Approximation algorithms for geometric problems. In [8].

[5] F. Chudak. Improved approximation algorithms for uncapacitated facility location. Submitted for publication.

[6] S. Guha and S. Khuller. Greedy Strikes Back: Improved Facility Location Algorithms. *Proceedings of the Ninth ACM-SIAM Symposium on Discrete Algorithms*, 1998.

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

[8] D. Hochbaum, ed. Approximation Algorithms for NP-hard problems. PWS Publishing, Boston, 1996.

[9] A. K. Jain, R. C. Dubes, *Algorithms for Clustering Data*, Prentice-Hall, 1981.

[10] J-H. Lin and J. S. Vitter. $\epsilon$-approximations with minimum packing constraint violation. *Proc. 24th ACM STOC*, 771–782, 1992.

[11] J.-H. Lin and J.S. Vitter. Approximation algorithms for the geometric median problems. *Information Proc. Letters* 44, 148–162, 1992.

[12] O. L. Mangasarian. Mathematical programming in data mining. *Data mining and knowledge discovery*, 1(2), 1997.

[13] D. B. Shmoys, E. Tardos and K. Aardal. Approximation algorithms for facility location problems (extended abstract). *Proc. 29th ACM STOC*, 265–274, 1997.