SIAM J. COMPUT.
Vol. 41, No. 1, pp. 47–60

ON CLUSTERING TO MINIMIZE THE SUM OF RADII*

MATT GIBSON†, GAURAV KANADE‡, ERIK KROHN§, IMRAN A. PIRWANI¶, AND KASTURI VARADARAJAN‡

Abstract. Let P be a set of n points in the plane. Consider the problem of finding k disks, each centered at a point in P, whose union covers P with the objective of minimizing the sum of the radii of the disks. We present an exact algorithm for this well-studied problem with polynomial running time, under the assumption that two candidate solutions can be compared efficiently. The algorithm generalizes in a straightforward manner to any fixed dimension and to some other related problems.

Key words. k-clustering, min-cost k-cover, minimum sum of radii cover

AMS subject classifications. 68W25, 68U05, 68W40, 68Q25

DOI. 10.1137/100798144

1. Introduction.

Given a metric d defined on a set V of points (a metric space), we define the ball $\mathrm{B}(v,r)$ centered at $v \in V$ and having radius $r \geq 0$ to be the set $\{q \in V | d(v,q) \leq r\}$. In this work, we consider the problem of computing a minimum cost k-cover for a given set $P \subseteq V$ of n points, where $k > 0$ is some given integer which is also part of the input. For $\kappa \geq 0$, a κ-cover for subset $Q \subseteq P$ is a set of at most κ balls, each centered at a point in P, whose union covers (contains) Q. The cost of a set $\mathcal{D}$ of balls, denoted $\mathrm{cost}(\mathcal{D})$, is the sum of the radii of those balls.

In the metric version of the min-cost k-cover problem, we are given P and k and the distance d between every pair of points in P. In the Euclidean version, P is given as a set of points in some fixed dimensional Euclidean space $\mathbb{R}^l$, and d is then the standard Euclidean distance. Both the metric and the Euclidean version of the problem have been well examined, motivated by applications in clustering and base-station coverage [8, 6, 11, 5, 2].

Doddi et al. [8] consider the metric min-cost k-cover problem and the closely related problem of partitioning P into a set of k clusters so as to minimize the sum of the cluster diameters. Following their terminology, we will call the latter problem *clustering to minimize the sum of diameters*. They present a bicriteria poly-time algorithm that returns $O(k)$ clusters whose cost is within a multiplicative factor $O(\log(n/k))$ of the optimal. They also show that the existence of a polynomial time algorithm that returns k clusters whose cost (sum of diameters) is strictly within 2 of the optimal

*Received by the editors June 10, 2010; accepted for publication (in revised form) October 23, 2011; published electronically January 5, 2012. A preliminary version of this paper appeared in the 18th ACM-SIAM Symposium on Discrete Algorithms (SODA), San Francisco, CA, 2008, pp. 819–825. The work of the first, second, third, and fifth authors was partially supported by NSF CAREER award CCR 0237431.

http://www.siam.org/journals/sicomp/41-1/79814.html

†Department of Electrical and Computer Engineering, University of Iowa, Iowa City, IA 52242 (matthew-gibson@uiowa.edu).

‡Department of Computer Science, University of Iowa, Iowa City, IA 52242 (gaurav.kanade@gmail.com, kasturi-varadarajan@uiowa.edu).

§Department of Computer Science, University of Wisconsin - Oshkosh, Oshkosh, WI 54901 (krohne@uwosh.edu).

¶Department of Computing Science, University of Alberta, Edmonton, AB T6G 2E8, Canada (imran.pirwani@gmail.com). This author's work was partially supported by Alberta Ingenuity.

would imply that $P = NP$. Charikar and Panigrahy [6] give a polynomial time algorithm based on the primal-dual method that gives a constant factor approximation—around 3.504—for the k-cover problem and thus also a constant factor approximation for clustering to minimize the sum of diameters.

Proietti and Widmayer [13] consider a problem closely related to the metric min-cost k-cover problem, which they call the k-radius problem. Here, the input is a weighted graph $G = (V, E)$ and an integer k, and the goal is to partition the vertex set into k nonempty subsets $V_1, \ldots, V_k$ so that the sum of the radii of the induced subgraphs of the V_i is minimized. The radius of the induced subgraph $G[V_i]$ is the smallest ball containing V_i in the metric space given by the shortest path distances in $G[V_i]$. They prove that the problem is NP-hard, but for fixed k they give polynomial time algorithms.

Lev-Tov and Peleg [11] give a polynomial time approximation scheme for a problem that is closely related to the geometric min-cost k-cover problem. Here we are given a finite set $F \subseteq \mathbb{R}^2$ of servers or facilities and a finite set $C \subseteq \mathbb{R}^2$ of clients, and the goal is to cover the clients by a min-cost set of disks centered at the facilities. They call this the *minimum sum of radii cover* (MSRC) problem. One significant difference from the k-cover problem is that here the number of disks that can be used is not bounded. Bilo et al. [5] give polynomial time approximation schemes for generalizations of the MSRC problem. Specifically, they also give such approximation schemes for a generalization of the MSRC problem where we want to minimize the αth power of the radii of the balls, where $\alpha \geq 1$. They also show that for $\alpha \geq 2$ such a generalization min-cost k-cover problem in the plane and the MSRC problem are NP-hard. Alt et al. [2] show that the NP-hardness result for the MSRC problem can be extended to any $\alpha > 1$. They give fast constant-factor approximation algorithms for the MSRC problem and also consider various related problems.

The well-known k-center problem is a variant of the k-cover problem where the cost of a set of balls is defined to be the maximum radius of any ball in the set. The metric version of this problem has a 2-approximation, and even the geometric version is hard to approximate to within a certain constant factor (see the survey by Bern and Eppstein [4]).

1.1. Our results and techniques. We show that the Euclidean min-cost k-cover problem is solvable exactly in polynomial time, under the assumption that the cost of any two candidate covers can be compared in polynomial time; this assumption is considered in more detail shortly. Our result also extends to variants of the problem such as the MSRC problem considered by Lev-Tov and Peleg [11]. This should be contrasted with the NP-hardness established by Alt et al. [2] for the versions of the MSRC with any $\alpha > 1$.

The result is enabled by a simple observation about the structure possessed by optimal solutions—for the planar case, this observation says that there always exists a vertical line that separates the input in a useful way and intersects $O(1)$ disks of the optimal solution. This is stated more precisely in section 2.

This observation opens up the possibility of computing an optimal k-cover efficiently via dynamic programming, as exemplified by [1, 12]. Some additional ideas are, however, needed to achieve a polynomial time algorithm—we have to deal with the accumulation of the number of disks we need to guess in a recursive application of Lemma 2.1, and also with some complexity that arises because the aspect ratio of the input point set P is not assumed to be bounded by a polynomial in n.

Our polynomial time exact algorithm is obtained under an assumption about the

model of computation, which we now describe. In the geometric min-cost k-cover problem, the optimal solution is a set of k disks, each of which is centered at some input point and each of which has a radius that is the distance between two input points. If the input points have integer coordinates, the cost of such a solution is the sum of square roots of integers. Our algorithm requires the ability to determine, given two such candidate solutions, the one that has lower cost. Our assumption, not unusual in such contexts, is that this can be done in polynomial time. For example, an essentially equivalent assumption is explicitly made by Vaidya [15] in the context of a fast algorithm for computing a matching in the plane that minimizes the sum of the Euclidean lengths of the matching edges.

Notice that for variants of the problem such as the one in which the underlying distance metric is the L_1 or the L_∞ metric rather than the Euclidean metric, we can indeed compare the costs of two candidate solutions in polynomial time. Our approach readily yields a polynomial time algorithm for such variants without any assumptions on the model of computation. In the context of the Euclidean metric, however, whether two sums of square roots of integers can be compared in polynomial time [7, 14] is a longstanding open problem.

We therefore translate our exact algorithm for the Euclidean metric into an approximation algorithm that does not make the above assumption on the computational model. We obtain an approximation algorithm that for any parameter $0 < \epsilon < 1$ runs in time polynomial in the input size and $\log \frac{1}{\epsilon}$ and returns a solution whose cost is at most $(1 + \epsilon)$ times the optimal. It is worth emphasizing that the dependence on ϵ is a polynomial in $\log \frac{1}{\epsilon}$ rather than in $\frac{1}{\epsilon}$.

Organization of the paper. The rest of this article is organized as follows. In section 2, we establish the separability of an optimal solution and other basic observations used subsequently. In section 3, we present the exact polynomial time algorithm for the min-cost k-cover problem in the plane. We then remove our assumption about the model of computation and obtain an approximation algorithm. In section 4, we describe how the results of section 3 can be extended to the MSRC problem in the plane. Section 5 concludes with remarks on extending the planar results to any fixed dimension. In the rest of this section, we discuss the work in [10] that applies the insights gleaned here to the metric min-cost k-cover problem.

Metric min-cost k-cover. The fact that optimal solutions are eminently separable holds in some appropriate way even for the metric min-cost k-cover problem. In [10], we show that if the *aspect ratio* of the input point set P, that is, the ratio of the maximum to minimum interpoint distance within P, is bounded by Δ, then a randomized algorithm that runs in $n^{O(\log n \cdot \log \Delta)}$ time returns an optimal k-cover of P with high probability. Thus when Δ is bounded by a polynomial in n, we obtain quasi-polynomial running time. The main tool in this extension to the metric case is the use of probabilistic partitions [3, 9] in place of the line separators above.

A natural question is whether there is a quasi-polynomial time algorithm for the case where the input metric has unbounded aspect ratio. This is unlikely to be the case because, as we show, the metric min-cost k-cover problem is NP-hard even for metrics induced by weighted planar graphs [10]. The construction here is remarkably similar to that of Proietti and Widmayer [13] for the NP-hardness of the closely related k-radius problem discussed above. More surprisingly, also in [10], we show that the min-cost k-cover problem is NP-hard for metrics of constant doubling dimension for a large enough constant. The results contained herein for fixed dimensional normed spaces are somewhat surprising when contrasted with this negative result on doubling metrics—algorithmic results for fixed dimensional normed spaces often

generalize to metrics of constant doubling dimension. The aspect ratio in the NP-hardness construction for both results is about 2^n [10]. Since the problem admits an exact, quasi-polynomial time algorithm for metrics having aspect ratio that is polynomially bounded in n, the existence of an exact, polynomial time algorithm for metrics induced by unweighted graphs has not been ruled out [10]; it is an interesting open problem to devise an exact algorithm for this case. It is also not known whether an exact algorithm that runs in time polynomially bounded in the aspect ratio and in the number of points exists. Using standard techniques, however, we obtain a quasi-polynomial time approximation scheme for the general, metric min-cost k-cover problem [10]. Thus, the general problem is unlikely to be APX-hard.

2. Preliminaries. We begin with the key observation that any optimal solution to the k-cover problem is eminently separable. To state this formally, we focus on the planar case where the input to the problem is a finite set $P \subseteq \mathbb{R}^2$ of points and an integer k. We define the *length* of a rectangle as the length of its longer side. If the rectangle is a square, the longer side is defined arbitrarily. The *width* of the rectangle is the length of its shorter side. A *separator* for rectangle R is any line s that is perpendicular to the longer side (length) of R, intersects R, and whose distance from each of the two shorter sides of R is at least a third of the length of R.

LEMMA 2.1. *Consider an optimal κ-cover $\mathcal{O}$ for some set $Q \subseteq P$ of points contained in a rectangle R. The rectangle R has a separator that intersects at most* 12 *disks in $\mathcal{O}$.*

Proof. Let l denote the length of R. Choose a separator uniformly at random from the set of separators for R. The probability that it intersects a disk $D \in \mathcal{O}$ is bounded by $\frac{2 \cdot \text{radius}(D)}{l/3}$. It follows that the expected number of disks in $\mathcal{O}$ intersected by a separator is at most $\frac{6}{l} \sum_{D \in \mathcal{O}} \text{radius}(D) = \frac{6 \cdot \text{cost}(\mathcal{O})}{l}$. Now $\text{cost}(\mathcal{O})$, the sum of the radii of disks in $\mathcal{O}$, is at most $2l$. This is because one disk of radius $2l$ centered at any point in Q covers the point set Q. We conclude that the expected number of disks in $\mathcal{O}$ intersected by a random separator is at most 12. ∎

The next lemma states a structural property of an optimal solution in the planar case. It is similar to observations made by Lev-Tov and Peleg [11], and its proof is sketched here for completeness.

LEMMA 2.2. *Let OPT denote an optimal k-cover for $P \subseteq \mathbb{R}^2$. Let R be a rectangle of length $a > 0$. The number of disks in OPT of radius at least a that intersect R is at most* 40.

Proof. Let $\text{OPT}' \subseteq \text{OPT}$ be the disks whose radius is at least a. The rectangle R can be enclosed by a disk B of radius a. We will show that the number of disks in OPT' that intersect B is at most 40.

A basic property of the optimal solution OPT is its *admissibility*, the fact that no disk in OPT contains in its interior the center of another disk in OPT of positive radius. This implies that the centers of any two disks in OPT' are at least a apart.

Consider the disk B' concentric with B and with radius $2a$. Consider those centers of disks in OPT' that lie within B'. Consider the balls of radius $\frac{a}{2}$ at each of these centers. The interiors of any two of these balls are disjoint, and all the balls are contained in a ball of radius $\frac{5a}{2}$. Thus the number of these balls is at most $\frac{\pi(5a/2)^2}{\pi(a/2)^2} = 25$. We conclude that there are at most 25 disks in OPT' whose centers lie within the disk B'.

We now bound the number of disks in OPT' with center outside B' that also intersect B. We partition the plane radially into 15 sectors with angle $2\pi/15$ and

centered at the center of B. Consider two points p and q that are outside B' but within the same sector, and assume that p is closer to B than q. It is easy to verify that the distance $|qp|$ is smaller than the distance of q from B. Thus, if a ball centered at q intersects B, then it must contain p in its interior. It follows from the admissibility of OPT that for each sector there can be at most one disk in OPT (and thus also OPT$'$) that is centered in the sector outside B' and that intersects B. See Figure 2.1.

Thus, the number of disks in OPT that intersect B and have radius at least a is at most $25 + 15 = 40$, and the lemma is established. Note that in the fairly standard packing argument we have used here, there is a trade-off between the radius of B' and how large the sector angles have to be. While we have optimized this trade-off to an extent, the bound of 40 still seems rather loose, as is evident from the slack in the very first step of our proof, where we enclose the rectangle R with length a in the ball B of radius a. □

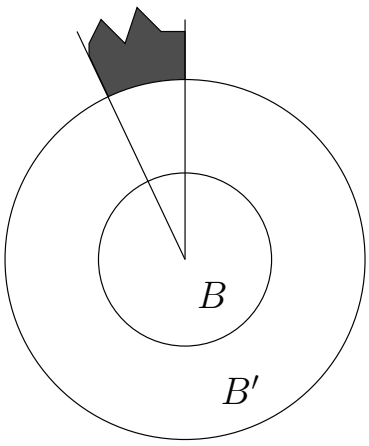

FIG. 2.1. *At most one disk from OPT can have its center in the shaded area and intersect B.*

3. The algorithm in the planar case. In this section, we describe a polynomial time algorithm to compute an optimal k-cover for a set P of n points in the plane, assuming that the costs of two covers can be efficiently compared. We will assume that $k < n$, since otherwise the optimal k-cover is trivially computed. Let $\mathcal{D}$ denote the set of those disks whose center is some $p \in P$ and whose radius is $|pq|$ for some $q \in P$. Note that the set $\mathcal{D}$ includes the disks of radius 0 as well, and thus $|\mathcal{D}| = n^2$. We start by observing that there is an optimal k-cover of P whose disks are chosen from the set $\mathcal{D}$. In the rest of this section we will reserve $\mathcal{D}$ to denote this set of disks.

Canonical and critical lines. We say that a vertical (resp., horizontal) line is *critical* if it passes through a point in P or a point of vertical (resp., horizontal) tangency of some disk in $\mathcal{D}$. (See Figure 3.1.) Note that if l and l' are two vertical (resp., horizontal) noncritical lines with no vertical (resp., horizontal) critical line between them, then l and l' intersect the same set of disks in $\mathcal{D}$. Motivated by this, we define a set of $\Theta(n^2)$ *canonical* vertical lines that includes exactly one arbitrarily chosen vertical line between every two consecutive vertical critical lines, one vertical line to the left of the leftmost critical line and one vertical line to the right of the rightmost critical line. This is illustrated in Figure 3.1. We define a set of $\Theta(n^2)$ canonical horizontal lines analogously.

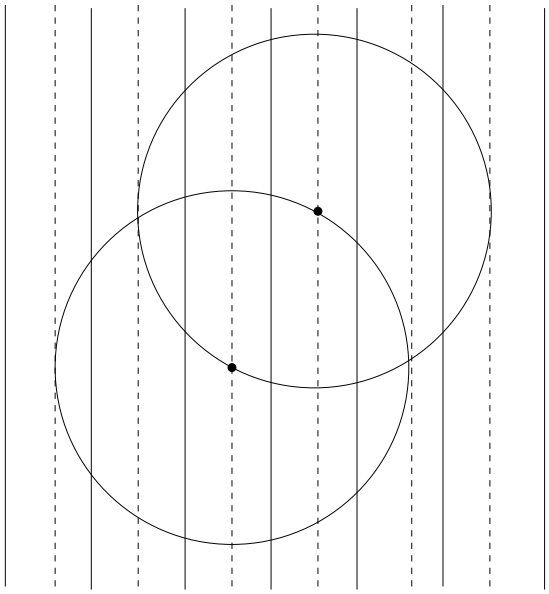

FIG. 3.1. *Critical (dashed) and canonical (solid) lines.*

Balanced rectangles and compression. A rectangle is said to be *balanced* if its width is at least a third of its length. We describe a procedure compress(R) that takes as input a balanced rectangle R that contains at least two of the points in P and returns a balanced rectangle R' such that (a) R' is contained in R, (b) R' contains $P \cap R$, and (c) for any separator for R', there are points of $P \cap R$ in both of the open halfspaces that it bounds (and consequently, any separator for R' partitions $P \cap R$ into two nonempty subsets).

To describe the procedure compress(R), we assume without loss of generality that the separators of R are vertical (its length is horizontal). The procedure first checks whether there is a point in $P \cap R$ that is strictly to the left of the leftmost separator of R, and whether there is a point in $P \cap R$ that is strictly to the right of the rightmost separator of R. If so, the procedure returns $R' = R$.

Otherwise, let R' be the minimal rectangle enclosing $P \cap R$. Let l' be its length and w' its width. Let l and w denote the length and width, respectively, of R. If $w' \geq l'/3$, we are done and we simply return R'.

If $w' < l'/3$, we have to make R' wider while still keeping it enclosed in R. There are two cases to consider, depending on whether the longer side of R' is parallel to the longer side of R.

The first case involves the longer side of R' being parallel to the longer side of R, and is illustrated in Figure 3.2. In this case, it must be that $l' \leq 2l/3$. The width of R' needs to be expanded to $l'/3$. Since $l'/3 \leq 2l/9 \leq l/3 \leq w$, there is enough room for this expansion; that is, we can enlarge R' so that its width is now $l'/3$, its length and in fact its x-projection remain the same, and it is still contained in R.

In the second case, the longer side of R' is parallel to the shorter side of R. Note that $l' \leq w$. The width of R' needs to be expanded to $l'/3$. Since $l'/3 \leq w/3 \leq l/3 \leq l$, there is enough room for this expansion.

The algorithm. We describe a procedure $\texttt{DC}(R, \kappa, T)$ that takes as input a balanced

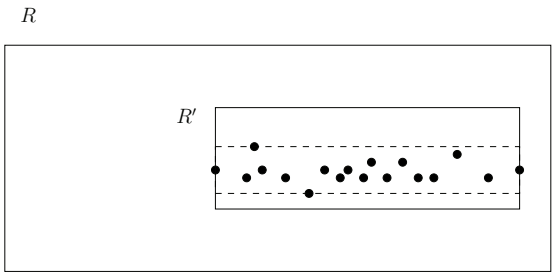

FIG. 3.2. *R and $R' = compress(R)$. (Dashed is R' before expansion.)*

rectangle R, an integer $\kappa \geq 0$, and a subset $T \subseteq \mathcal{D}$. It returns a κ-cover of the set

$$Q = \{q \in P \cap R \mid q \text{ is not covered by } T\}.$$

We will later argue that $\texttt{DC}(S, k, \emptyset)$ computes an optimal k-cover for P when S is any balanced rectangle containing P.

The result of a call to procedure $\texttt{DC}(R, \kappa, T)$ is stored in a table entry indexed by $P \cap R, \kappa, T$; the entry Table$(P \cap R, \kappa, T)$ stores a κ-cover for the set Q defined above. It will be useful for the description of the algorithm to define a special "disk" I whose cost is ∞. If an entry Table$(P \cap R, \kappa, T)$ stores a set that contains I, this means that the algorithm has determined that there is no κ-cover for Q. The table entry will store such a set only in case $\kappa = 0$.

See Algorithm 1 below for the description of the procedure $\texttt{DC}(R, \kappa, T)$.

A couple of remarks about the algorithm are in order. Observe that partitioning a balanced rectangle by a separator results in two balanced rectangles. It follows that R_1 and R_2 are balanced, given that R' is balanced.

No separator that the algorithm considers on step 6 passes through a point in P. We therefore do not have to worry about breaking ties because input points land on a separator.

Suppose that the procedure $\texttt{DC}(R, \kappa, T)$ finds that in step 1 the entry Table$(P \cap R, \kappa, T)$ has not been created. It then proceeds to create this entry. In between the times that this entry is created and subsequently filled in by this procedure, no subproblem will need the entry Table$(P \cap R, \kappa, T)$. This is because for any call $\texttt{DC}(\hat{R}, \hat{\kappa}, \hat{T})$ that is nested within $\texttt{DC}(R, \kappa, T)$, $P \cap \hat{R}$ is a strict subset of $P \cap R$. Such a call $\texttt{DC}(\hat{R}, \hat{\kappa}, \hat{T})$ will not enquire as to whether the entry Table$(P \cap R, \kappa, T)$ has been created, nor will it seek to know the content of this entry.

Running time. We now bound the overall running time of a call to $\texttt{DC}(S, k, \emptyset)$, where S is some balanced rectangle containing P. Note that each table entry is indexed by a set of points $P \cap R$ for some balanced rectangle R, a $\kappa \leq k$, and a set $T \subseteq \mathcal{D}$ such that $|T| \leq \beta = 424$. The number of such $P \cap R$ is $O(n^4)$, the number of such κ is $O(n)$, and the number of such T is $O(n^{2\beta})$. The number of table entries is therefore bounded by $O(n^{853})$. We use this to bound the total number of calls $\texttt{DC}(R, \kappa, T)$ made.

The number of calls of the form $\texttt{DC}(R, \kappa, T)$ that create a corresponding table entry Table$(P \cap R, \kappa, T)$ is also bounded by $O(n^{853})$, since each table entry is created only once. Any call of the form $\texttt{DC}(R, \kappa, T)$ that does not create the table entry Table$(P \cap R, \kappa, T)$ does not make any recursive calls and takes $O(1)$ time. We charge this to the parent instance of $\texttt{DC}()$ that makes the recursive call $\texttt{DC}(R, \kappa, T)$. Note that

ALGORITHM 1. $\texttt{DC}(R, \kappa, T)$.

1: If Table$(P \cap R, \kappa, T)$ has already been created, return. Otherwise, create table entry Table$(P \cap R, \kappa, T)$, which will be assigned below.
2: Let $Q = \{q \in P \cap R \mid q \text{ is not covered by } T\}$. If $Q = \emptyset$, let Table$(P \cap R, \kappa, T) \leftarrow \emptyset$, and return.
3: If $\kappa = 0$, let Table$(P \cap R, \kappa, T) \leftarrow \{I\}$ (covering is infeasible), and return.
4: If $|Q| = 1$, assign to Table$(P \cap R, \kappa, T)$ the set containing the trivial disk consisting of the one point in Q, and return.
5: If we are in this step, the set $P \cap R$ has at least two points in it. Call compress(R) to obtain a balanced rectangle R' containing $P \cap R$. Let us assume for the purposes of exposition that the separators for R' are vertical. Let $L(R')$ be the set consisting of those vertical canonical lines that are separators for R'. Also include in $L(R')$ the leftmost (resp., rightmost) separator for R', provided that it is not critical. Initialize a cover $\mathcal{D}' \leftarrow \{I\}$.
6: **for all** choices of a separator $l \in L(R')$ **do**
7: **for all** choices of a set $\mathcal{D}_0 \subseteq \mathcal{D}$ of at most 12 disks that intersects l **do**
8: **for all** choice of $\kappa_1, \kappa_2 \geq 0$ such that $\kappa_1 + \kappa_2 + |\mathcal{D}_0| \leq \kappa$ **do**
9: Let R_1 and R_2 be two rectangles into which l partitions R'. Let $T_1 = \{D \in T \cup \mathcal{D}_0 \mid D \text{ intersects } R_1\}$. Let $T_2 = \{D \in T \cup \mathcal{D}_0 \mid D \text{ intersects } R_2\}$.
10: **if** $|T_1| \leq \beta \equiv 424$ and $|T_2| \leq \beta$, **then**
11: Recursively call $\texttt{DC}(R_1, \kappa_1, T_1)$ and $\texttt{DC}(R_2, \kappa_2, T_2)$.
12: If cost$(\mathcal{D}_0 \cup \text{Table}(P \cap R_1, \kappa_1, T_1) \cup \text{Table}(P \cap R_2, \kappa_2, T_2)) < \text{cost}(\mathcal{D}')$, then update $\mathcal{D}' \leftarrow \mathcal{D}_0 \cup \text{Table}(P \cap R_1, \kappa_1, T_1) \cup \text{Table}(P \cap R_2, \kappa_2, T_2)$.
13: Assign Table$(P \cap R, \kappa, T) \leftarrow \mathcal{D}'$, and return.

this parent instance creates its corresponding table entry. Any instance of DC() makes $O(n^{28})$ direct recursive calls, since there are $O(n^2)$ choices for separator l in line 6, $O(n^{24})$ choices for $\mathcal{D}_0$ in line 7, and $O(n)$ choices for κ_1 and κ_2 in line 8. We can conclude using our charging argument that the number of calls of the form $\texttt{DC}(R, \kappa, T)$ that do not create a corresponding table entry Table$(P \cap R, \kappa, T)$ is bounded by $O(n^{881})$.

Putting everything together, we can somewhat loosely upper bound the overall running time of $\texttt{DC}(S, k, \emptyset)$ by $O(n^{881} \cdot T(n))$, where $T(n)$ is an upper bound on the time needed to compare the costs of two subsets of $\mathcal{D}$, each of size at most n.

Correctness. Establishing that $\texttt{DC}(S, k, \emptyset)$ returns an optimal k-cover of P is more involved than showing that it runs in polynomial running time, mainly because of the pruning step that ensures that instance $\texttt{DC}(R, \kappa, T)$ that is called has $|T|$ bounded by β. We begin by showing the following technical fact about the algorithm.

LEMMA 3.1. *Let $\texttt{DC}(G_t, \cdot, \cdot)$ be a recursive call made inside a sequence of nested recursive calls that involved $\texttt{DC}(G_1, k, \emptyset), \texttt{DC}(G_2, \cdot, \cdot), \ldots, \texttt{DC}(G_{t-1}, \cdot, \cdot)$, where $G_1 = S$. Suppose that $|G_t \cap P| \geq 2$. Let G'_j be the result obtained by a call to* compress(G_j). *Let $l(G'_j)$ be the separator chosen for G'_j for each $1 \leq j \leq t-1$, so that G_{j+1} is one of the two rectangles into which $l(G'_j)$ partitions G'_j. Furthermore, let $l(G_t)$ be any separator for G'_t, and let R_1 be one of the two rectangles into which $l(G'_t)$ partitions G'_t. Denote by a the length of R_1. Then the horizontal distance between any two distinct vertical separators $l(G'_i)$ and $l(G'_j)$ is at least $a/3$, and the vertical distance between any two distinct horizontal separators $l(G'_i)$ and $l(G'_j)$ is at least $a/3$.*

Proof. Since $R_1 \subseteq G'_t \subseteq G_t \subseteq G'_{t-1} \subseteq G_{t-1} \subseteq \cdots \subseteq G'_1 \subseteq G_1$, the length of any

G'_i is at least a. Let G'_i and G'_j be such that $i < j$ and $l(G'_i)$ and $l(G'_j)$ are vertical. Note that G'_j is contained in one of the two rectangles into which $l(G'_i)$ partitions G'_i. Thus $l(G'_i)$ is either to the left of G'_j or to its right. Now $l(G'_j)$ lies between the two vertical sides of G'_j at a distance of at least a third of the length of G'_j from either side. Since the length of G'_j is at least a, it follows that the horizontal distance between $l(G'_i)$ and $l(G'_j)$ is at least $a/3$. See Figure 3.3 for an illustration.

The case where $l(G'_i)$ and $l(G'_j)$ are horizontal is reasoned similarly. □

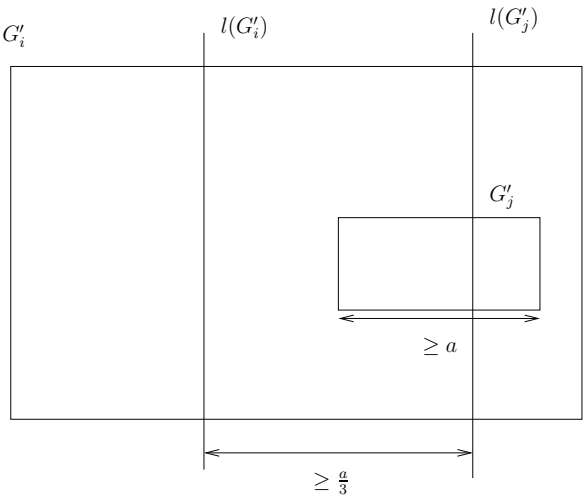

FIG. 3.3. *Illustration for Lemma* 3.1.

The correctness of the algorithm follows from the following lemma; let us fix an optimal k-cover $\mathrm{OPT} \subseteq \mathcal{D}$ of P.

LEMMA 3.2. *Suppose the recursive instance* $\mathtt{DC}(R, \kappa, T)$ *is called (at some recursion depth) by the top-level invocation to* $\mathtt{DC}(S, k, \emptyset)$. *Suppose also that* $T \subseteq OPT$ *and* T *contains every disk in* OPT *that contains a point in* $P \cap R$ *as well as a point in* $P \setminus R$. *(*T *possibly contains other disks in* OPT *as well.) Let* $Q = \{q \in P \cap R \mid q \text{ is not covered by } T\}$, *and let* OPT' *denote the set of disks in* OPT *that contain points in* Q. *Suppose further that* $\kappa \geq |OPT'|$. *Then after the call to* $\mathtt{DC}(R, \kappa, T)$ *completes,* $Table(P \cap R, \kappa, T)$ *contains a* κ*-cover for* Q *whose cost is at most* $cost(OPT')$.

Proof. The proof is by induction on $|P \cap R|$. We may assume that $\mathrm{Table}(P \cap R, \kappa, T)$ has not been created when $\mathtt{DC}(R, \kappa, T)$ is called; otherwise, this table entry was created by a call $\mathtt{DC}(\hat{R}, \kappa, T)$ for some rectangle $\hat{R}$ such that $\hat{R} \cap P = R \cap P$. We check that the suppositions made for T and κ hold with $\hat{R}$ in the place of R, and also that Q and OPT' are the same in both cases. Further, we can use the arguments below to conclude that after the call to $\mathtt{DC}(\hat{R}, \kappa, T)$ completes, $\mathrm{Table}(P \cap R = P \cap \hat{R}, \kappa, T)$ contains a κ-cover for Q whose cost is at most $\mathrm{cost}(\mathrm{OPT}')$. The subsequent call to $\mathtt{DC}(R, \kappa, T)$ returns immediately without changing $\mathrm{Table}(P \cap R = P \cap \hat{R}, \kappa, T)$, which therefore continues to contain a κ-cover for Q whose cost is at most $\mathrm{cost}(\mathrm{OPT}')$.

The cases where the call to $\mathtt{DC}(R, \kappa, T)$ is returned via steps 2 or 4 form the base cases of the induction and are easily established. Note that the call cannot return via step 3, because if $Q \neq \emptyset$, then $\kappa \geq |\mathrm{OPT}'|$ must be at least 1.

The inductive case is when the call returns in step 13. Note that $|P \cap R|, |Q| \geq 2$ in this case. We will assume without loss of generality that the separators of R' are vertical.

We first observe that OPT' is an optimal $|\mathrm{OPT}'|$-cover for Q. This is because points in $(P \cap R) \setminus Q$ are covered by T, and OPT' does not cover any point in $P \setminus R$, so the only role that OPT' plays is to cover Q, and it therefore must do this optimally. From Lemma 2.1, we conclude that R' has a separator l' that intersects at most 12 disks in OPT'. It is easy to see that we can move l' to one of the lines in $L(R')$ (computed in step 5) without adding to the set of disks in $\mathcal{D}$ that it intersects. We therefore consider the choice of a separator $l \in L(R')$ that intersects a set $\hat{\mathcal{D}}_0$ of at most 12 disks from OPT'. Consider the body of the innermost for loop where l is chosen as above, $\mathcal{D}_0$ is chosen to be $\hat{\mathcal{D}}_0$, κ_1 is set to be the number of disks in OPT' to the left of l, and κ_2 is set to be the number of disks in OPT' to the right of l. Let OPT_1 and OPT_2 denote the disks in OPT' to the left and right of l, respectively. Of course, $|\mathrm{OPT}_1| = \kappa_1$ and $|\mathrm{OPT}_2| = \kappa_2$.

Let R_1, R_2, T_1, and T_2 be exactly as in the algorithm. Note that $|P \cap R_1| < |P \cap R|$ and $|P \cap R_2| < |P \cap R|$. Notice further that $T_1 \subseteq \mathrm{OPT}$ and T_1 contains every disk in OPT that contains a point in $P \cap R_1$ as well as a point in $P \setminus R_1$. Furthermore, OPT_1 is the set of disks in OPT that contain points in $Q_1 = \{q \in P \cap R_1 \mid q \text{ is not covered by } T_1\}$, and $k_1 = |\mathrm{OPT}_1|$. We can invoke the induction hypothesis to claim that *if the call* $\mathtt{DC}(R_1, \kappa_1, T_1)$ *is made*, then after the call, $\mathrm{Table}(R_1 \cap P, \kappa_1, T_1)$ contains a κ_1-cover of Q_1 whose cost is at most $\mathrm{cost}(\mathrm{OPT}_1)$. Similarly, we can invoke the induction hypothesis to claim that if the call $\mathtt{DC}(R_2, \kappa_2, T_2)$ is made, then after the call, $\mathrm{Table}(R_2 \cap P, \kappa_2, T_2)$ contains a κ_2-cover of $Q_2 = \{q \in P \cap R_2 \mid q \text{ is not covered by } T_2\}$ whose cost is at most $\mathrm{cost}(\mathrm{OPT}_2)$. We will show that $|T_1|$ and $|T_2|$ are bounded by β and that therefore these calls are indeed made. It follows that in step 12, $\mathcal{D}_0 \cup \mathrm{Table}(P \cap R_1, \kappa_1, T_1) \cup \mathrm{Table}(P \cap R_2, \kappa_2, T_2)$ is a κ-cover for Q whose cost is at most $\mathrm{cost}(\mathcal{D}_0) + \mathrm{cost}(\mathrm{OPT}_1) + \mathrm{cost}(\mathrm{OPT}_2) = \mathrm{cost}(\mathrm{OPT}')$. Thus in step 13, $\mathrm{Table}(P \cap R, \kappa, T)$ is assigned a κ-cover of Q with cost at most $\mathrm{cost}(\mathrm{OPT}')$.

To complete the proof we show that $|T_1| \leq \beta = 424$; the proof for T_2 is similar.

Suppose that we arrived at the (R, κ, T) instance by proceeding through a sequence of nested recursive calls that involved $\mathtt{DC}(G_1, k, \emptyset), \mathtt{DC}(G_2, \cdot, \cdot), \ldots, \mathtt{DC}(G_{t-1}, \cdot, \cdot)$, where $G_1 = S$. Let $G_t = R$. Let G'_j be the result obtained by a call to $\mathrm{compress}(G_j)$. Let $l(G'_j)$ be the separator chosen for G'_j so that for each $1 \leq j \leq t-1$, G_{j+1} is one of the two rectangles into which $l(G'_j)$ partitions G'_j. Finally, let $l(G'_t) = l$ and note that $G'_t = R'$. Thus R_1 is one of the two rectangles into which $l(G'_t)$ partitions G'_t. Furthermore let $\mathcal{D}_0(G_j)$ denote the corresponding choice of $\mathcal{D}_0$ made in the call $\mathtt{DC}(G_j, \cdot, \cdot)$ for $1 \leq j \leq t-1$, and let $\mathcal{D}_0(G_t)$ equal the $\mathcal{D}_0$ above. Note that $T_1 \subseteq \cup_{j=1}^{t} \mathcal{D}_0(G_j)$.

Let $\bar{R}$ be a square of side $5a$ that is centered at the center of R_1, where a is the length of R_1. Lemma 3.1 implies that the number of j for which $l(G'_j)$ intersects $\bar{R}$ is bounded by a constant. In particular, the number of such vertical (resp., horizontal) $l(G'_j)$ is at most 16, since any two of them are apart by at least $a/3$. So the constant can be taken to be 32. It follows that the number of disks in T_1 that belong to $\mathcal{D}_0(G_j)$ for such j is bounded by $32 \times 12 = 384$, since $|\mathcal{D}_0(G_j)| \leq 12$.

Now consider the disks in T_1 that belong to $\mathcal{D}_0(G_j)$ for j such that $l(G'_j)$ does not intersect $\bar{R}$. Any such disk intersects $l(G'_j)$ as well as R_1, so it must have radius at least a. It follows from Lemma 2.2 that the number of such disks is bounded by 40.

We conclude that $|T_1| \leq \beta = 424$. ∎

It is immediate from Lemma 3.2 that after the call $\mathtt{DC}(S, k, \emptyset)$ completes, Table $(P, k, \emptyset)$ contains a k-cover for P whose cost is at most $\mathrm{cost}(\mathrm{OPT})$, that is, an optimal k-cover.

THEOREM 3.3. *There is an algorithm that, given a set P of n points in the plane*

and an integer $k \geq 1$, runs in $O(n^{881} \cdot T(n))$ time and returns an optimal k-cover of P. Here, $T(n) \geq 1$ is an upper bound on the time needed to compare the costs of two subsets of $\mathcal{D}$, each of size at most n, and $\mathcal{D}$ is the set of n^2 disks whose center is some $p \in P$ and whose radius is $|pq|$ for some $q \in P$.

3.1. Relaxing the assumption on the computational model. Theorem 3.3 implies that an optimal k-cover of a point set P in the plane can be computed in polynomial time assuming that the costs of two subsets of $\mathcal{D}$ can be compared in polynomial time. We now obtain an approximation algorithm for the problem without making such an assumption. Let OPT continue to denote an optimal k-cover for the point set P. Suppose that for each disk $D \in \mathcal{D}$ we are given a *proxy cost* $\text{pcost}(D) \geq 0$, which is some rational number. Define the proxy cost $\text{pcost}(\mathcal{D}')$ of a set $\mathcal{D}' \subseteq \mathcal{D}$ of disks to be the sum of the proxy costs of the disks in $\mathcal{D}'$.

Let us modify the procedure $\texttt{DC}(R, \kappa, T)$ so that in step 12 it compares proxy costs rather than actual costs. That is, the modified predicate checks whether

$$\text{pcost}(\mathcal{D}_0 \cup \text{Table}(P \cap R_1, \kappa_1, T_1) \cup \text{Table}(P \cap R_2, \kappa_2, T_2))$$

is less than $\text{pcost}(\mathcal{D}')$. We also consider the proxy cost of the special disk I to be infinity. Arguing along the lines of the previous section, we conclude the following.

LEMMA 3.4. *After the call to the modified procedure $\texttt{DC}(S, k, \emptyset)$ completes, Table$(P, k, \emptyset)$ contains a k-cover for P whose proxy cost is at most pcost(OPT), the proxy cost of OPT.*

Suppose that the coordinates of the input points P are integers whose binary encoding takes at most L bits. Given our tolerance parameter $\epsilon > 0$, we compute for each disk $D \in \mathcal{D}$ a rational number that lies in the interval $[\text{radius}(D), (1+\epsilon)\text{radius}(D)]$ and set $\text{pcost}(D)$ to be this number. This computation is readily accomplished in time polynomial in L and $\log \frac{1}{\epsilon}$ for a single disk D. With the proxy costs set in this manner, we conclude from Lemma 3.4 that after the call to the modified procedure $\texttt{DC}(S, k, \emptyset)$ completes, $\text{Table}(P, k, \emptyset)$ contains a k-cover for P whose cost is at most $(1 + \epsilon)\text{cost}(\text{OPT})$.

Finally, it is straightforward to ensure that the computation of the canonical lines and the lines bounding the rectangles that are encountered as arguments of our recursive algorithm takes time that is polynomial in the input size of the problem (and independent of ϵ). Also, predicates such as testing whether a disk $D \in \mathcal{D}$ contains a point $p \in P$, or testing whether a disk $D \in \mathcal{D}$ intersects a rectangle that is encountered in the course of the algorithm, are also readily implemented in polynomial time.

THEOREM 3.5. *There is an algorithm that, given a set P of points in the plane, an integer $k \geq 1$, and a parameter $0 < \epsilon < 1$, runs in time polynomial in the input size and $\log \frac{1}{\epsilon}$ and returns a k-cover of P whose cost is at most $(1 + \epsilon)$ times the cost of an optimal k-cover.*

4. The MSRC problem. We now outline an extension of the results in section 3 to the MSRC problem. Here, we are given a set C of clients and a set F of facilities in the plane, and the goal is to cover the clients by a minimum cost set of disks centered at the facilities. One difference from the k-cover problem is that here there is no upper bound on the number of disks that the solution can use. While this arguably makes the task of designing an efficient algorithm easier, the situation is somewhat complicated by the fact that the solution can use only disks centered at the facilities and not the clients. The purpose of this section is to point out that this complication can indeed be handled by the techniques we have developed.

We may assume without loss of generality that no point is both a client and a facility. Let $P = C \cup F$, and let $n = |P|$. The key observation in Lemma 2.1 is adapted to the context of MSRC as follows.

LEMMA 4.1. *Consider any optimal solution to an instance of the MSRC consisting of a set $Q \subseteq C$ of clients and a nonempty set $F' \subseteq F$ of facilities, both contained in a rectangle R. The rectangle R has a separator that intersects at most 12 disks in this optimal solution.*

For the rest of this section, let $\mathcal{D}$ denote the set of $O(n^2)$ disks, each centered at some facility $p \in F$ and having radius $|pq|$ for some client $q \in C$. We observe that the MSRC instance with F and C can be solved by finding the minimum cost subset of $\mathcal{D}$ that covers C. The canonical and critical lines are defined as before, except that P and $\mathcal{D}$ mean what they do in the present context.

The algorithm. We describe a recursive procedure $\texttt{DC}(R, T)$ that takes as input a balanced rectangle R and a subset $T \subseteq \mathcal{D}$ of disks. It returns a cover (a subset of $\mathcal{D}$) for the set $Q = \{q \in C \cap R \mid q \text{ is not covered by } T\}$. The input instance of MSRC is solved by calling $\texttt{DC}(S, \emptyset)$, where S is a balanced rectangle containing $P = C \cup F$. The eventual guarantee is that this call $\texttt{DC}(S, \emptyset)$ computes an optimal cover for C.

The result of a call to procedure $\texttt{DC}(R, T)$ is stored in a table entry indexed by $P \cap R$ and T; the entry $\text{Table}(P \cap R, T)$ stores a cover for the set Q defined above.

See Algorithm 2 below for the description of the procedure $\texttt{DC}(R, T)$. As before, the running time of the overall algorithm, that is, the call to $\texttt{DC}(S, \emptyset)$, runs in time that is bounded by $n^{O(1)} \cdot T(n)$, where $T(n) \geq 1$ is the time needed to compare the costs of two subsets of $\mathcal{D}$.

ALGORITHM 2. $\texttt{DC}(R, T)$.

1: If $\text{Table}(P \cap R, T)$ has already been created, return. Otherwise, create table entry $\text{Table}(P \cap R, T)$, which will be assigned below.
2: Let $Q = \{q \in C \cap R \mid q \text{ is not covered by } T\}$. If $Q = \emptyset$, let $\text{Table}(P \cap R, T) \leftarrow \emptyset$, and return.
3: If $|Q| = 1$, assign to $\text{Table}(P \cap R, \kappa, T)$ the set containing the smallest radius disk in $\mathcal{D}$ that contains the only point in Q, and return.
4: If we are in this step, the set $P \cap R$ has at least two points in it, since it includes at least two clients. Call compress(R) to obtain a balanced rectangle R' containing $P \cap R$. Let us assume for the purposes of exposition that the separators for R' are vertical. Let $L(R')$ be the set consisting of those vertical canonical lines that are separators for R'. Also include in $L(R')$ the leftmost (resp., rightmost) separator for R', provided that it is not critical. Initialize a cover $\mathcal{D}'$ to be an arbitrary subset of $\mathcal{D}$ that covers Q.
5: **for all** choices of a separator $l \in L(R')$ **do**
6: **for all** choices of a set $\mathcal{D}_0 \subseteq \mathcal{D}$ of at most 12 disks that intersects l **do**
7: Let R_1 and R_2 be two rectangles into which l partitions R'. Let $T_1 = \{D \in T \cup \mathcal{D}_0 \mid D \text{ intersects } R_1\}$. Let $T_2 = \{D \in T \cup \mathcal{D}_0 \mid D \text{ intersects } R_2\}$.
8: **if** $|T_1| \leq \beta$ and $|T_2| \leq \beta$, where β is a sufficiently large constant, **then**
9: Recursively call $\texttt{DC}(R_1, T_1)$ and $\texttt{DC}(R_2, T_2)$.
10: If $\text{cost}(\mathcal{D}_0 \cup \text{Table}(P \cap R_1, T_1) \cup \text{Table}(P \cap R_2, T_2)) < \text{cost}(\mathcal{D}')$, then update $\mathcal{D}' \leftarrow \mathcal{D}_0 \cup \text{Table}(P \cap R_1, T_1) \cup \text{Table}(P \cap R_2, T_2)$.
11: Assign $\text{Table}(P \cap R, T) \leftarrow \mathcal{D}'$, and return.

Correctness. Let $\text{OPT} \subseteq \mathcal{D}$ denote an optimal cover for the set C of clients. The following lemma is analogous to Lemma 3.2 and establishes the correctness of the above algorithm.

LEMMA 4.2. *Suppose that the recursive instance* $\texttt{DC}(R,T)$ *is called (at some recursion depth) by the top-level invocation to* $\texttt{DC}(S,\emptyset)$. *Suppose also that* $T \subseteq OPT$ *and* T *contains every disk in* OPT *that contains a point in* $P \cap R$ *as well as a point in* $P \setminus R$. *(*T *possibly contains other disks in* OPT *as well.) Let* $Q = \{q \in C \cap R \mid q \text{ is not covered by } T\}$, *and let* OPT' *denote the set of disks in* OPT *that contain points in* Q. *Then after the call to* $\texttt{DC}(R,T)$ *completes,* $\text{Table}(P \cap R, T)$ *contains a cover for* Q *whose cost is at most* $\text{cost}(OPT')$.

The proof appeals to Lemma 4.1, where the proof of Lemma 3.2 appeals to Lemma 2.1, and is very similar otherwise. It is immediate from Lemma 4.2 that after the call to $\texttt{DC}(S,\emptyset)$ completes, $\text{Table}(P,\emptyset)$ contains a cover for C whose cost is at most cost(OPT).

THEOREM 4.3. *There is an algorithm that, given an MSRC instance with a set* C *of clients and a set* F *of facilities that are points in* $\mathbb{R}^2$, *runs in time* $n^{O(1)} \cdot T(n)$ *and returns an optimal solution. Here,* $n = |F \cup C|$, *and* $T(n) \geq 1$ *is an upper bound on the time required to compare any two subsets of* $\mathcal{D}$; $\mathcal{D}$ *consists of the set of* $O(n^2)$ *disks with center at some* $p \in F$ *and radius* $|pq|$ *for some* $q \in C$.

Modifying the algorithm to use proxy costs as done in section 3, we obtain the following approximation algorithm.

THEOREM 4.4. *There is an algorithm that, given an MSRC instance with a set* C *of clients and a set* F *of facilities that are points in* $\mathbb{R}^2$, *and a parameter* $0 < \epsilon < 1$, *runs in time polynomial in the input size and* $\log \frac{1}{\epsilon}$ *and returns a solution with cost at most* $(1+\epsilon)$ *times that of an optimal solution.*

5. Concluding remarks. The results of sections 3 and 4 generalize to any fixed dimension. We conclude by defining the needed generalization of the notion of a separator and a balanced rectangle. The analogue of a rectangle in $\mathbb{R}^l$ for $l \geq 3$ is an axes-parallel box: the Cartesian product, $[a_1,b_1] \times [a_2,b_2] \times \cdots \times [a_l,b_l]$, where the interval $[a_i,b_i]$ is of course the projection of the box onto the ith coordinate axis. We define the *length* of such a box as the length of the longest of the l intervals, and the *width* to be the length of the shortest of the l intervals.

Suppose that the length of such a box B is defined by its projection $[a_i,b_i]$ on the ith coordinate axis; let us break the tie arbitrarily if more than one of the projections has this length. A separator for B is any hyperplane of the form $x_i = c$, where c falls in the middle third of the $[a_i,b_i]$ interval.

We say that the box B is balanced if its width is at least a third of its length. Let P be a finite set of points. Just as in section 3, we can describe a procedure compress(B) that takes as input any balanced rectangle B that contains at least two of the points in P and returns a balanced rectangle B' such that (a) B' is contained in B, (b) B' contains $P \cap B$, and (c) for any separator for B', there are points of $P \cap B$ in both of the open halfspaces that it bounds.

With these ideas fixed, the algorithms and results of sections 3 and 4 generalize in a very natural way to $\mathbb{R}^l$, yielding polynomial running times as before. The exponents of the polynomials will of course depend on the dimension l.

Acknowledgments. We thank Chandra Chekuri for valuable suggestions and the anonymous referees for their feedback.

REFERENCES

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

[2] H. Alt, E. Arkin, H. Bronnimann, J. Erickson, S. Fekete, C. Knauer, J. Lenchner, J. Mitchell, and K. Whittlesey, *Minimum-cost coverage of points by disks*, in Proceedings of the Annual Symposium on Computational Geometry, 2006, pp. 449–458.

[3] Y. Bartal, *Probabilistic approximations of metric spaces and its algorithmic applications*, in Proceedings of the IEEE Symposium on the Foundations of Computer Science, 1996, pp. 184–193.

[4] M. Bern and D. Eppstein, *Approximation algorithms for geometric problems*, in Approximation Algorithms for NP-Hard Problems, D. Hochbaum, ed., PWS Publishing Company, Boston, 1997, pp. 296–345.

[5] V. Bilo, I. Caragiannis, C. Kaklamanis, and P. Kanellopoulos, *Geometric clustering to minimize the sum of cluster sizes*, in Proceedings of the European Symposium on Algorithms, Lecture Notes in Comput. Sci. 3669, Springer, New York, 2005, pp. 460–471.

[6] M. Charikar and R. Panigrahy, *Clustering to minimize the sum of cluster diameters*, J. Comput. Systems Sci., 68 (2004), pp. 417–441.

[7] E. D. Demaine, J. S. B. Mitchell, and J. O'Rourke, *Problem* 33: *Sum of Square Roots*, The Open Problems Project, http://maven.smith.edu/∼orourke/TOPP/P33.html.

[8] S. R. Doddi, M. V. Marathe, S. S. Ravi, D. S. Taylor, and P. Widmayer, *Approximation algorithms for clustering to minimize the sum of diameters*, Nordic J. Comput., 7 (2000), pp. 185–203.

[9] J. Fakcharoenphol, S. Rao, and K. Talwar, *A tight bound on approximating arbitrary metrics by tree metrics*, in Proceedings of the ACM Symposium on Theory of Computing, 2003, pp. 448–455.

[10] M. Gibson, G. Kanade, E. Krohn, I. Pirwani, and K. Varadarajan, *On metric clustering to minimize the sum of radii*, Algorithmica, 57 (2010), pp. 484–498.

[11] N. Lev-Tov and D. Peleg, *Polynomial time approximation schemes for base station coverage with minimum total radii*, Computer Networks, 47 (2005), pp. 489–501.

[12] J. S. B. Mitchell, *Guillotine subdivisions approximate polygonal subdivisions: A simple polynomial-time approximation scheme for geometric TSP, k-MST, and related problems*, SIAM J. Comput., 28 (1999), pp. 1298–1309.

[13] G. Proietti and P. Widmayer, *Partitioning the nodes of a graph to minimize the sum of subgraph radii*, in Proceedings of the International Symposium on Algorithms and Computation (ISAAC), 2005, pp. 578–587.

[14] J. Qian and C. A. Wang, *How much precision is needed to compare two sums of square roots of integers?*, Inform. Process. Lett., 100 (2006), pp. 194–198.

[15] P. M. Vaidya, *Geometry helps in matching*, SIAM J. Comput., 18 (1989), pp. 1201–1225.