# On Clustering to Minimize the Sum of Radii[*]

Matt Gibson Gaurav Kanade Erik Krohn Imran A. Pirwani
Kasturi Varadarajan

## 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 $\kappa$-cover for subset $Q \subseteq P$ is a set of at most $\kappa$ 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 $\Re^l$, 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 would imply that $P = NP$. Charikar and Panigrahy [6] give a poly-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.

Lev-Tov and Peleg [11] give a polynomial time approximation scheme for a problem that is closely related problem to the geometric min-cost $k$-cover problem. Here we are given a finite set $S \subseteq \Re^2$ of servers and a finite set $C \subseteq \Re^2$ of clients, and the goal is to cover the clients by a min-cost set of disks centered at the servers. 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. They also give such approximation schemes when the cost of a set of balls is defined to be the sum of the $\alpha$-th power of their radii, where $\alpha \geq 1$. They also show that the min-cost $k$-cover problem in the plane and the MSRC problem are NP-hard with this generalized definition of cost for every $\alpha \geq 2$. 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]).

**Our Results and Techniques.** We show that the Euclidean min-cost $k$-cover problem ($\alpha = 1$) is solvable exactly in polynomial time. This result also extends to variants of the problem such as the MSRC problem considered by LevTov and Peleg [11] and the generalizations studied by Bilo et al. [5] for $\alpha = 1$.

The result is enabled by a simple observation about the structure possessed by optimal solutions – they are eminently separable. To state this formally, we focus on the planar case and 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*

[*]All the authors are at the Department of Computer Science at the University of Iowa. Work by the first, second, third, and fifth authors was partially supported by NSF CAREER award CCR 0237431. Email: {mrgibson,gkanade,eakrohn,pirwani,kvaradar}@cs.uiowa.edu

for $R$ 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 1.1. *Consider an optimal $\kappa$-cover $\mathcal{D}$ 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 $\mathcal{D}$.*

*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 $D \in \mathcal{D}$ is bounded by $\frac{2 \cdot \text{radius}(D)}{l/3}$. It follows that the expected number of disks in $\mathcal{D}$ intersected by a separator is at most $\frac{6}{l} \sum_{D \in \mathcal{D}} \text{radius}(D) = \frac{6 \cdot \text{cost}(\mathcal{D})}{l}$. Now $\text{cost}(\mathcal{D})$, the sum of the radii of disks in $\mathcal{D}$, is at most $2l$. This is because one disk of radius $2l$ centered at any point in $Q$ covers $Q$. We conclude that the expected number of disks in $\mathcal{D}$ intersected by a random separator is at most 12. $\square$

This observation opens up the possibility of computing an optimal $k$-cover efficiently via dynamic programming, as exemplified by [1, 12]. Some additional play is 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 1.1, 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 result as stated 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. We make the assumption, not unusual in such contexts, that this can be done in polynomial time.

Notice that for variants of the problem such as the one in which the underlying distance metric is the $L_1$ 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, it is a longstanding open problem as to whether two sums of square roots of integers can be compared in polynomial time [7, 13].

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.

**Organization of the paper.** The rest of this article is organized as follows. In Section 2, we establish basic observations used subsequently. In Section 3, we present the exact polynomial time algorithm for the min-cost $k$-cover problem in the plane. The extensions to any fixed dimensional Euclidean space and to related problems such as MSRC are straightforward and are omitted from this version. We then remove our assumption about the model of computation and obtain an approximation algorithm. In the rest of this section, we outline extensions of the above results to the metric case.

**Metric Min-Cost $k$-cover.** The fact that optimal solutions are eminently separable holds true 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 $\Delta$, 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 $\Delta$ 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 quasipolynomial 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 general metric min-cost $k$-cover problem is NP-hard.

Using standard techniques, however, we obtain a quasi-polynomial time approximation scheme for the metric min-cost $k$-cover problem. We also investigate the problem for interesting special cases such as metrics of constant doubling dimension.

## 2 Preliminaries

The following lemma states a structural property of the 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 2.1. *Let $OPT$ denote an optimal $k$-cover for $P \subseteq \Re^2$. Let $R$ be a rectangle of length $a > 0$. The number of disks in $OPT$ of radius at least $a$ that intersect $R$ is bounded by a constant.*

*Proof.* Let $\mathrm{OPT}' \subseteq \mathrm{OPT}$ 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 $\mathrm{OPT}'$ that intersect $B$ is $O(1)$.

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 $\mathrm{OPT}'$ are at least $a$ apart. It follows that there are $O(1)$ disks in $\mathrm{OPT}'$ whose centers lie within the disk $B'$ concentric with $B$ and with radius $10a$.

To bound the number of disks in $\mathrm{OPT}'$ with center outside $B'$ that also intersect $B$, we partition the plane radially into 8 sectors with angle $\pi/4$ and centered at the center of $B$. (See Figure 1.) It is an easy consequence of admissibility that for each sector there can be at most one disk in OPT (and thus also $\mathrm{OPT}'$) that is centered in the sector outside $B'$ and that intersects $B$. $\square$

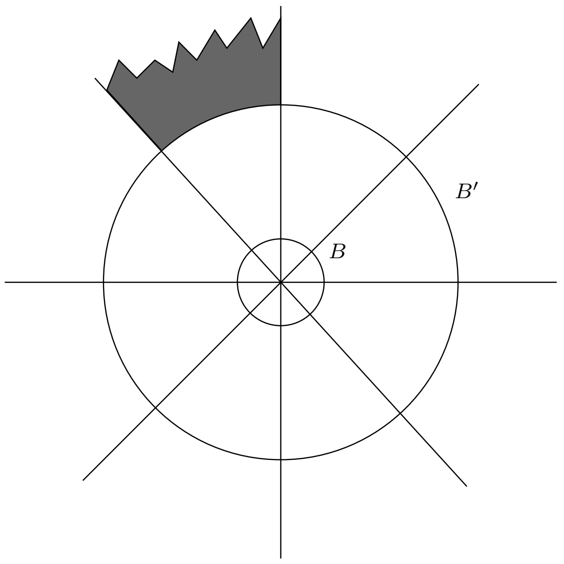

Figure 1: At most one disk from OPT can have its center in the shaded area and intersect B.

## 3 The Algorithm in the Planar Case

We describe a polynomial time algorithm to compute an optimal $k$-cover for a set $P$ of $n$ points in the plane. We will assume that $k \leq n$, since otherwise the optimal $k$-cover is trivially computed. We start by observing that there is an optimal $k$-cover of $P$ whose disks are chosen from the set $\mathcal{D}$ of disks whose center is some $p \in P$ and whose radius is $|pq|$ for some $q \in P$. Note that $|\mathcal{D}| = n^2$. In the rest of the article 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 2.) 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 any one 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. We define a set of $\Theta(n^2)$ canonical horizontal lines analogously.

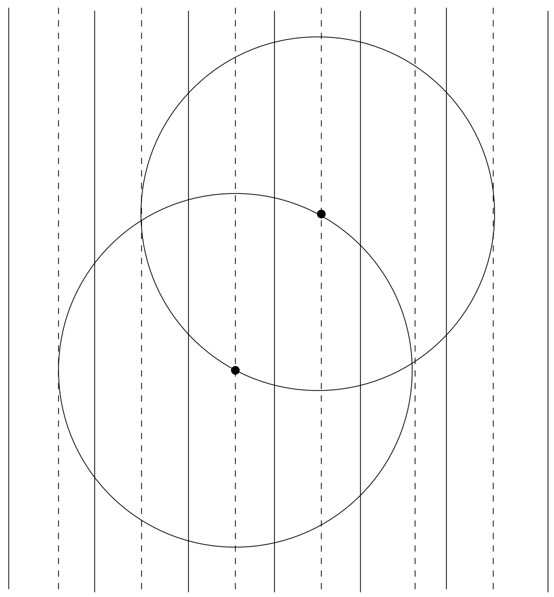

Figure 2: Critical (dashed) and canonical (solid) lines.

**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 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 if there is a point in $P \cap R$ that is strictly to the left of the leftmost separator of $R$, and if 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 of $R$, respectively. 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 $R$. If the longer side of $R'$ is parallel to the shorter side of $R$, we know that $l' \leq w$. We also know that $l \geq w \geq l/3$. The width of $R'$ needs to be expanded to $l'/3$. Since $l'/3 \leq w/3 \leq l/3$, there is enough room for this expansion.

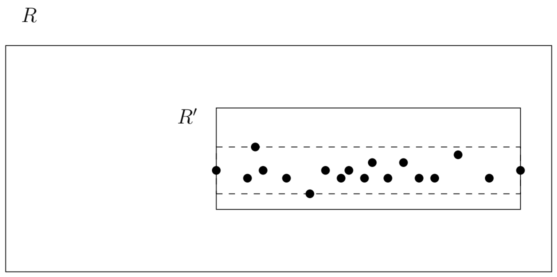

Figure 3: $R$ and $R' = \text{compress}(R)$. (Dashed is $R'$ before expansion.)

The second case involves the longer side of $R'$ being parallel to the longer side of $R$. (See Figure 3.) 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.

**The Algorithm.** We describe a procedure $\texttt{DC}(R, \kappa, T)$ that takes as input a balanced rectangle $R$, an integer $\kappa \geq 0$, and a subset $T \subseteq \mathcal{D}$. It returns a $\kappa$ 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 $\text{Table}(P \cap R, \kappa, T)$ stores a $\kappa$-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 $\infty$. If an entry $\text{Table}(P \cap R, \kappa, T)$ stores a set that contains $I$, this means that the algorithm has determined that there is no $\kappa$-cover for $Q$.

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 $\text{Table}(P \cap R, \kappa, T)$ has not been created. It then proceeds to create this entry. In between the time this entry is created and subsequently filled in by this procedure, no subproblem will need the entry $\text{Table}(P \cap R, \kappa, T)$. 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 $\text{Table}(P \cap R, \kappa, T)$ 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 $|T| \leq \beta$. The number of such $P \cap R$ is $O(n^4)$, the number of such $\kappa$ is $O(n)$ and the number of such $T$ is $O(n^{2\beta})$. Assuming $\beta$ is a constant, the number of table entries is therefore bounded by a polynomial in $n$. 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 $\text{Table}(P \cap R, \kappa, T)$ is polynomially bounded, since the number of table entries is polynomially bounded and each table entry is created only once. Any call of the form $\texttt{DC}(R, \kappa, T)$ that does not create the table entry $\text{Table}(P \cap R, \kappa, T)$ 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 this parent instance creates its corresponding table entry. Since any instance of $\texttt{DC}()$ makes only a polynomial number of direct recursive calls, 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 $\text{Table}(P \cap R, \kappa, T)$ is also bounded by a polynomial.

Since any instance of $\texttt{DC}()$ takes polynomial time in $n$ not counting the time for the recursive calls, we can conclude that the overall running time of $\texttt{DC}(S, k, \emptyset)$ is bounded by a polynomial in $n$.

**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 $\beta$. We begin by showing the following fact about the algorithm.

LEMMA 3.1. *Let $\texttt{DC}(R_1, \kappa_1, T_1)$ be a recursive call made*

**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 $Q = \{q \in P \cap R \mid q$ 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 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 $T_1 = \{D \in T \cup \mathcal{D}_0 \mid D$ intersects $R_1\}$. Let $T_2 = \{D \in T \cup \mathcal{D}_0 \mid D$ intersects $R_2\}$.
10:      if $|T_1| \leq \beta$ and $|T_2| \leq \beta$ ($\beta$ is a large enough constant) then
11:        Recursively call $\texttt{DC}(R_1, \kappa_1, T_1)$ and $\texttt{DC}(R_2, \kappa_2, T_2)$.
12:        If cost($\mathcal{D}_0 \cup$ Table($P \cap R_1, \kappa_1, T_1$) $\cup$ Table($P \cap R_2, \kappa_2, T_2$)) $<$ cost($\mathcal{D}'$), then update $\mathcal{D}' \leftarrow \mathcal{D}_0 \cup$ Table($P \cap R_1, \kappa_1, T_1$) $\cup$ Table($P \cap R_2, \kappa_2, T_2$).
13: Assign Table($P \cap R, \kappa, T$) $\leftarrow \mathcal{D}'$, and return.
```

*inside a sequence of nested recursive calls that involved* $\texttt{DC}(G_1, k, \emptyset), \texttt{DC}(G_2, \cdot, \cdot), \ldots, \texttt{DC}(G_t, \cdot, \cdot)$, *where* $G_1 = S$. *Let* $G'_j$ *be the result obtained by a call to compress*$(G_j)$. *Let* $l(G'_j)$ *be the separator chosen for* $G'_j$ *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* $G'_j$, *and* $R_1$ *is 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 4.)

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

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

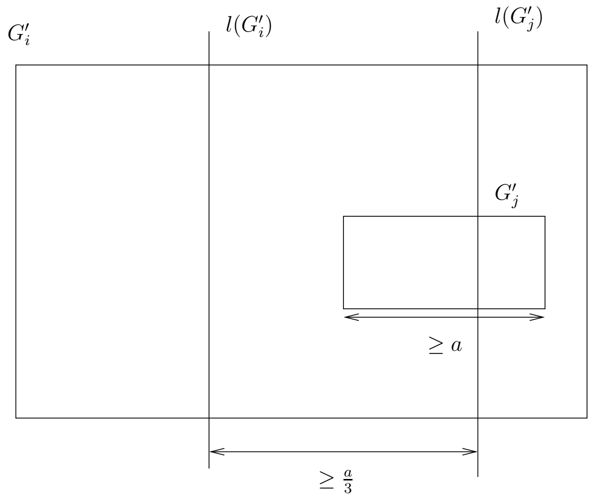

Figure 4: Illustration for Lemma 3.1

Lemma 3.2. *Suppose the recursive instance* $\texttt{DC}(R, \kappa, T)$ *is called (at some recursion depth) by the top-level invocation to* $\texttt{DC}(S, k, \emptyset)$. *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* $Q = \{q \in P \cap R \mid q$ *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* $\texttt{DC}(R, \kappa, T)$ *completes, Table*$(P \cap R, \kappa, T)$ *contains a*

*$\kappa$-cover for $Q$ whose cost is at most cost($OPT'$).*

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

The cases where the call to $\texttt{DC}(R, \kappa, T)$ 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 $\mathrm{OPT}'$ is an optimal $|\mathrm{OPT}'|$-cover for $Q$. This is because points in $(P \cap R) \setminus Q$ are covered by $T$, and $\mathrm{OPT}'$ does not cover any point in $P \setminus R$ – so the only role that $\mathrm{OPT}'$ plays is to cover $Q$, and it therefore must do this optimally. From Lemma 1.1, we conclude that $R'$ has a separator $l'$ that interesects at most 12 disks in $\mathrm{OPT}'$. Let $\hat{\mathcal{D}}_0$ denote this set of disks. Since the set of critical lines is finite, the argument of Lemma 1.1 tells us that we can assume $l'$ to be a line that is not critical. It is easy to see that we can move $l'$ to one of the lines in $L(R')$ (computed in step 5) without changing the set of disks in $\mathcal{D}$ that it intersects. We therefore consider the choice of a separator $l \in L(R')$ whose intersection with $\mathrm{OPT}'$ is exactly $\hat{\mathcal{D}}_0$. Consider the body of the innermost for loop where $l$ is chosen as above, $\mathcal{D}_0$ is chosen to be $\hat{\mathcal{D}}_0$, $\kappa_1$ is set to be the number of disks in $\mathrm{OPT}'$ to the left of $l$ and $\kappa_2$ is set to be the number of disks in $\mathrm{OPT}'$ to the right of $l$. Let $\mathrm{OPT}_1$ and $\mathrm{OPT}_2$ denote the disks in $\mathrm{OPT}'$ 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, $\mathrm{OPT}_1$ 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* $\texttt{DC}(R_1, \kappa_1, T_1)$ *is made*, then after the call, Table$(R_1 \cap P, \kappa_1, T_1)$ contains a $\kappa_1$-cover of $Q_1$ whose cost is at most cost$(\mathrm{OPT}_1)$. Similarly, we can invoke the induction hypothesis to claim that if the call $\texttt{DC}(R_2, \kappa_2, T_2)$ is made, then after the call, Table$(R_2 \cap P, \kappa_2, T_2)$ contains a $\kappa_2$-cover of $Q_2 = \{q \in P \cap R_2 \mid q \text{ is not covered by } T_2\}$ whose cost is at most cost$(\mathrm{OPT}_2)$. We will show that $|T_1|$ and $|T_2|$ are bounded by $\beta$ and that therefore these calls are indeed made. It follows that in step 12, $\mathcal{D}_0 \cup$ Table$(P \cap R_1, \kappa_1, T_1) \cup$ Table$(P \cap R_2, \kappa_2, T_2)$ is a $\kappa$-cover for $Q$ whose cost is at most cost$(\mathcal{D}_0)$ + cost$(\mathrm{OPT}_1)$ + cost$(\mathrm{OPT}_2)$ = cost$(\mathrm{OPT}')$. Thus in step 13, Table$(P \cap R, \kappa, T)$ is assigned a $\kappa$-cover of $Q$ with cost at most cost$(\mathrm{OPT}')$.

To complete the proof we show that $|T_1| \leq \beta$, where $\beta$ is a sufficiently large constant. The proof for $T_2$ is similar.

Suppose that we arrived at the $(R, \kappa, T)$ instance by proceeding through 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 $G_1 = S$. Let $G_t = R$. Let $G'_j$ be the result obtained by a call to compress$(G_j)$. Let $l(G'_j)$ be the separator chosen for $G'_j$ 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 $\texttt{DC}(G_j, \cdot, \cdot)$, 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 $10a$ 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. It follows that the number of disks in $T_1$ that belong to $\mathcal{D}_0(G_j)$ for such $j$ is bounded by a constant, 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 2.1 that the number of such disks is also bounded by a constant.

We conclude that $|T_1| \leq \beta$ for a large enough constant $\beta$. $\square$

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

THEOREM 3.1. *There is a polynomial time algorithm that, given a set $P$ of points in the plane and an integer $k \geq 1$, returns an optimal $k$-cover of $P$.*

**3.1 Relaxing the Assumption on the Computational Model.** We now remove the assumption made in deriving Theorem 3.1 that the costs of two candidate solutions can be compared in polynomial time, and obtain an approximation algorithm. 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 if

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

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:

Lemma 3.3. *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 3.3 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 $\epsilon$). 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 3.2. *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.*

## Acknowledgements

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.* Proceedings of the IEEE Symposium on Foundations of Computer Science, 1996, 2–12.

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

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

[4] M. Bern and D. Eppstein. *Approximation algorithms for geometric problems.* In D. Hochbaum (Ed.), Approximation algorithms for NP-hard problems, PWS Publishing Company, 1997, pages 296–345.

[5] V. Bilo, I. Caragiannis, C. Kaklamanis, and P. Kanellopoulos. *Geometric clustering to minimize the sum of cluster sizes.* Proceedings of the European Symposium on Algorithms, LNCS vol 3669, 460–471, 2005.

[6] M. Charikar and R. Panigrahy. *Clustering to minimize the sum of cluster diameters.* Journal of Computer and Systems Sciences, vol. 68 (2), 417–441, 2004.

[7] E. D. Demaine, J. S. B. Mitchell, and J. O'Rourke. *The open problems project, Problem 33: Sum of square roots*, `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 Journal of Computing, Vol 7(3), 185–203, 2000.

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

[10] M. Gibson, G. Kanade, E. Krohn, I. Pirwani, and K. Varadarajan. *On metric clustering to minimize the sum of radii.* Manuscript, 2007.

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

[12] J. S. B. Mitchell. *Guillotine subdivisions aprpoximate polygonal subdivisions: A simple polynomial-time approximation scheme for geometric TSP, k-MST, and related problems.* SIAM Journal on Computing, 28(4): 1298–1309.

[13] J. Qian and C. A. Wang. *How much precision is needed to compare two sums of square roots of integers?*, Information Processing Letters 100 (2006): 194–198.