On Minimum Sum of Radii and Diameters Clustering

Babak Behsaz* Mohammad R. Salavatipour†

February 21, 2014

Abstract

Given a metric (V, d) and an integer k, we consider the problem of partitioning the points of V into at most k clusters so as to minimize the sum of radii or the sum of diameters of these clusters. The former problem is called the Minimum Sum of Radii (MSR) problem and the latter is the Minimum Sum of Diameters (MSD) problem. The current best polynomial time algorithms for these problems have approximation ratios 3.504 and 7.008, respectively [4]. We call a cluster containing a single point, a *singleton* cluster. For the MSR problem when singleton clusters are not allowed, we give an exact algorithm for metrics induced by unweighted graphs. In addition, we show that in this case, a solution consisting of the best single cluster for each connected component of the graph is a $\frac{3}{2}$-approximation algorithm. For the MSD problem on the plane with Euclidean distances, we present a polynomial time approximation scheme. In addition, we settle the open problem of complexity of the MSD problem with constant k by giving a polynomial time exact algorithm in this case. The previously best known approximation algorithms for MSD on the plane or for MSD with constant k have both ratio 2.

1 Introduction

Clustering is one of the fundamental techniques in information technology, which has been used in a wide variety of application areas such as data mining, bioinformatics, and information retrieval. The main goal of this technique is to partition a set of objects into a number of homogeneous subsets, called *clusters*. In any clustering method, we need to define a distance measure between each pair of objects to determine how similar those objects are. In most clustering algorithms, we try to find a clustering that optimizes an objective function based on these distances. The well known k-center problem is the clustering problem with the objective of minimizing the maximum cluster radius (see [12] for a tight approximation algorithm).

Unfortunately, in some applications, using k-center objective function produces a *dissection effect*. This effect causes objects that should be placed in the same cluster to be assigned to different clusters [11]. To avoid this effect, it is proposed to minimize the sum of cluster radii or diameters. This leads to the Minimum Sum of Radii (MSR) and the Minimum Sum of Diameters (MSD) problems, respectively. In each of these problems, one is given a set of points V in a metric space d and the goal is to partition V into k clusters so as to minimize the sum of radii of clusters (in MSR) or the sum of diameters of the clusters (in MSD). We can consider these points as the vertices of a graph with a metric cost function on the edges. More formally, we are given a graph $G = (V, E)$ with edge costs (or distances) $d : E \to \mathbb{R}^+$ that satisfy triangle inequality. The goal is

*Department of Computing Science, University of Alberta, Edmonton, Alberta T6G 2E8, Canada. e-mail: behsaz@ualberta.ca. Supported in part by Alberta Innovates Graduate Student Scholarship.

†Department of Computing Science, University of Alberta, Edmonton, Alberta T6G 2E8, Canada. e-mail: mrs@ualberta.ca. Supported by NSERC and an Alberta Ingenuity New Faculty Award.

to partition V into k sets $V_1, V_2, \ldots, V_k$. In the MSR problem, we want to minimize $\sum_{i=1}^{k} \mathrm{rad}(V_i)$, where $\mathrm{rad}(V_i)$ is equal to $\min_{u \in V_i} \max_{v \in V_i} d(u, v)$. In the MSD problem, we want to minimize $\sum_{i=1}^{k} \mathrm{diam}(V_i)$, where $\mathrm{diam}(V_i)$ is equal to $\max_{u,v \in V_i} d(u, v)$.

1.1 Related Work

The MSR and MSD problems are well studied in general metrics. When the cost function is symmetric, a simple observation is that for any graph (or cluster) G: $\mathrm{rad}(G) \leq \mathrm{diam}(G) \leq 2\,\mathrm{rad}(G)$. Therefore, an α-approximation algorithm for MSD yields a 2α-approximation algorithm for MSR and vice versa. Doddi *et al.* [5] considered the MSD problem and showed that unless **P**= **NP**, for any $\epsilon > 0$, there is no $(2 - \epsilon)$-approximation algorithm for the MSD problem even when the graph is unweighted (i.e. the metric is the shortest-path metric of an unweighted graph). Note that this result does not imply NP-hardness of MSR. They also presented a bicriteria algorithm that returns a solution with $O(k)$ clusters whose cost is within $O(\log(n/k))$ factor of the optimum. Charikar and Panigrahy [4] significantly improved this result by giving a $(3.504 + \epsilon)$-approximation algorithm for MSR, and consequently a $(7.008 + \epsilon)$-approximation algorithm for MSD, that runs in time $n^{O(\frac{1}{\epsilon})}$. These are the current best ratios for the MSR and MSD problems on general metrics. In an interesting result, Gibson *et al.* [9] designed an exact algorithm for the MSR problem which runs in time $n^{O(\log n \log \Delta)}$ where Δ is the ratio of the largest distance over the smallest distance. They translate this result to a quasi-polynomial time approximation scheme (QPTAS) for general metrics. In contrast, they showed that the MSR problem is NP-hard even in metrics induced by weighted planar graphs and in metrics of constant doubling dimension [9].

There are also several results for the special cases of these problems. When $k = 2$, the MSD problem is solvable in polynomial time by a reduction to the 2-SAT problem [11]. When k is fixed and the metric is Euclidean, Capoyleas *et al.* [2] present an exact algorithm for MSD. When k is fixed, for general metrics, there is a 2-approximation for MSD [5]. This result can be obtained from a simple polynomial time exact algorithm for MSR [5], which uses the observation that the number of distinct maximal clusters is polynomially bounded. For Euclidean MSR, there is an exact polynomial time algorithm [10]. This result also extends to L_1 and L_∞ metrics. This also implies a 2-approximation for MSD on Euclidean plane, which is the current best ratio.

1.2 Our Results

For graphs with polynomially bounded Δ (for instance for unweighted graphs), the exact algorithm of Gibson *et al.* for the MSR problem [8] runs in time $n^{O(\log^2 n)}$. This gives us strong evidence that the MSR problem for these metrics is not **NP**-hard and one should be able to design an exact algorithm for the MSR problem when restricted to these metrics. In contrast, the best known polynomial time algorithm even for this case is the $(3.504 + \epsilon)$-approximation algorithm of Charikar and Panigrahy [3]. We make some progress in this direction and give a polynomial time exact algorithm for metrics of unweighted graphs in the case that no singleton clusters are allowed.

Theorem 1 *There is a polynomial time exact algorithm for the unweighted MSR problem when no singletons are allowed.*

This result reduces the unweighted MSR problem to the problem of finding the singleton clusters. In other words, it shows the difficult core of the problem is to determine which points should be a cluster by themselves.

Moreover, we show that there is a simple $\frac{3}{2}$-approximation algorithm for unweighted graphs in the case that no singleton clusters are allowed. This algorithm has a better running time than

the above exact algorithm. We contrast this simple algorithm with an integrality gap of at least essentially $\frac{3}{2}$ for a natural LP relaxation of the problem [1].

Theorem 2 *Finding the best single cluster of each connected component is a $\frac{3}{2}$-approximation algorithm for the unweighted MSR problem without singletons.*

For Euclidean MSD (*i.e.*, points in $\mathbb{R}^2$ and Euclidean metric), the exact algorithm of Capoyleas *et al.* for fixed k raises a question about the complexity of this problem for variable k. This is first asked by Doddi *et al.* as an open problem (see Section 6 in [5]). Recall that the exact algorithm of [10] gives a 2-approximation for Euclidean MSD. In contrast, there is a ratio 2 hardness for the general case [5]. Thus, it is not obvious if we can beat this factor of 2. We present a polynomial time approximation scheme (PTAS) for the Euclidean MSD.

Theorem 3 *For any given $\epsilon > 0$, there is an algorithm such that given a set of n points in $\mathbb{R}^2$ and integer k, finds a $(1+\epsilon)$-approximate solution for the MSD problem in time $n^{O(1/\epsilon)}$.*

Recall that Hansen and Jaumard [11] gave an exact algorithm for the MSD problem when $k = 2$. The best known approximation algorithm for the MSD problem with constant $k > 2$ is the 2-approximation algorithm that comes from the exact algorithm of the MSR problem in this case. Doddi et al. raised an open question (see Section 1.3 in [5]) about the complexity of this problem in this case. We answer this question by giving an exact algorithm for the MSD problem with constant k.

Theorem 4 *There is a polynomial time exact algorithm for the MSD problem when k is constant.*

Here is a short overview of the proofs' ideas. The results of Theorems 1 and 4 are based on the following simple question: assuming we are working with an optimal solution with minimum number of clusters, what prevents two clusters from being merged into a single cluster? For MSR without singletons (Theorem 1) the answer to this question reveals an interesting structural property of the solution, specifically the clusters either form a path or a cycle. This allows one to obtain a polynomial time algorithm using dynamic programming. For MSD with fixed k (Theorem 4), the answer is that two points (one in each cluster) are sufficiently far apart. If we know this information for every pair of clusters then we can identify the clusters. The algorithm for Theorem 3 is an extension of the results of [10], however we need some new ideas to obtain polynomial running time.

The rest of this paper is organized as follows. We discuss some preliminaries in Section 2. In Section 3, we present the exact algorithm for the unweighted MSR problem when no singleton clusters are allowed. We also present the simple 3/2-approximation algorithm under the same assumptions. In Section 4, we present a PTAS for the Euclidean MSD problem. Next, we give the exact algorithm for the MSD problem with constant k in Section 5. Our concluding remarks come in Section 6.

2 Preliminaries

First notice that when we have k sets covering V , i.e., each vertex of V is in at least one of these sets, we can easily obtain a partition of V from these sets without increasing their total radii or diameters. As a consequence, from now on we only try to find feasible solutions that form a covering of V (instead of a partition).

There is a strong connection between the MSR and MSD problems. This connection comes from the similarity of their objective functions and the relationship between radius and diameter

of a graph with metric weights: $\mathrm{rad}(G) \leq \mathrm{diam}(G) \leq 2\,\mathrm{rad}(G)$. Doddi *et al.* [6] noticed that the above connection has an important implication:

Proposition 1 *[6] An α-approximate solution for the MSR (MSD) problem is a 2α-approximate solution for the MSD (MSR, respectively) problem.*

While we have the above interesting connection between MSR and MSD, these problems have a very important difference. We call a cluster *maximal* for the MSR (MSD) problem if one cannot add any vertex to it without increasing its radius (diameter, respectively). Any solution can be turned into one having only maximal clusters without increasing the cost. Therefore, we can ignore non-maximal clusters and only consider maximal clusters in the solutions for the MSR and MSD problems.

Consider a maximal cluster of radius r for the MSR problem and let v be an arbitrary center of this cluster. By definition, this cluster must contain all the vertices in the graph having distance at most r from v. We call the set of such vertices the *ball of radius r around v* and denote it by $B(v, r)$. As it is stated in the following proposition, the number of distinct maximal clusters for the MSR problem is at most n^2, while the number of distinct maximal clusters for the MSD problem can be exponential.

Proposition 2 *[6] In the MSR problem, the number of distinct maximal clusters is at most n^2.*

This yields a very important advantage when we deal with the MSR problem. We can enumerate all subsets of the maximal clusters having size at most l in time $O(n^{2l})$. This enumeration is an important part of many related results. From now on, when we discuss the clusters in the MSR problem, we mean the clusters from the above set of distinct maximal clusters, which has size at most n^2.

3 MSR Restricted to Unweighted Graphs

In this section, we focus on the MSR problem when the metric is the shortest path metric of an unweighted graph. First, note that if one can optimally solve the MSR problem for some arbitrary metric in polynomial time for connected graphs, then using a standard dynamic programming approach, one can optimally solve the problem for that metric for all graphs:

Proposition 3 *The MSR problem reduces in polynomial time to the MSR problem for connected graphs.*

Proof. Assume we know how to solve the problem for connected graphs. First, for each $\kappa \leq k$, we find an optimal solution of κ clusters for each connected component. Let $H_1, H_2, \ldots, H_l$ be the connected components. We present how we can find the solution by adding one connected component at a time. For $1 \leq i < l$, we show how we can find an optimal solution of κ clusters for the graph $\cup_{j=1}^{i+1} H_j$ given the optimal solutions for the graph $\cup_{j=1}^{i} H_j$ for all $\kappa_1 \leq k$. For all $\kappa_1, \kappa_2 > 0$ such that $\kappa_1 + \kappa_2 = \kappa$, we combine the optimal solution of $\cup_{j=1}^{i} H_j$ with κ_1 clusters with the optimal solution of H_{i+1} with κ_2 clusters and store the best of these $\kappa - 1$ combined solutions as the optimal solution. ■

As a consequence, we are going to assume that the input graph is connected in the rest of this section. We start with some definitions.

Definition 1 We call a ball of positive radius a *non-zero ball*. We say two balls *intersect* if they share at least one common vertex. We say two balls are *adjacent* if they do not intersect but there is an edge that connects two vertices of these balls. Among the optimal solutions for covering a graph with at most k clusters, a *canonical optimal* solution is a solution with the minimum possible number of balls.

We have the following lemmas about the structure of a canonical optimal solution in an unweighted graph. Note that the next two lemmas hold even in presence of zero balls and we do not assume the input graph does not have singleton clusters here.

Lemma 1 *A canonical optimal solution does not have any intersecting balls.*

Proof. We show that two intersecting balls can be merged into one ball without increasing the total cost. Assume the intersecting balls are $B(v_1, r_1)$ and $B(v_2, r_2)$, where $r_1 \geq r_2$, and vertex u belongs to both of them (see Figure 1). Consider the shortest path between v_1 and u. Let v be the vertex on this path that has distance $\min(d(v_1, u), r_2)$ from v_1.

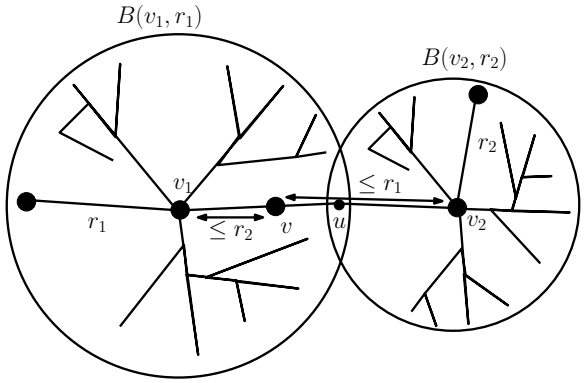

Figure 1: The intersecting balls $B(v_1, r_1)$ and $B(v_2, r_2)$. The ball $B(v, r_1 + r_2)$ covers all the vertices in these two balls.

The distance of v to all the vertices in $B(v_1, r_1)$ is at most $r_1 + r_2$, because the distance of v to v_1 is at most r_2 and the distance v_1 to the vertices in $B(v_1, r_1)$ is at most r_1. Also, the distance of v to all the vertices in $B(v_2, r_2)$ is at most $r_1 + r_2$, because the distance of v to u is at most $r_1 - r_2$, the distance of u to v_2 is at most r_2, and the distance of v_2 to the vertices in $B(v_2, r_2)$ is at most r_2. Therefore, $B(v, r_1 + r_2)$ contains all the vertices in $B(v_1, r_1)$ and $B(v_2, r_2)$ and in a solution, we can replace these two balls with $B(v, r_1 + r_2)$ without increasing the cost. ■

Remark 1 With a similar proof, one can show that we can cover two adjacent balls with radii r_1 and r_2 with a ball of radius $r_1 + r_2 + 1$. This shows that for the case of unweighted graphs, if we increase the number of clusters by one, the optimum value decreases by at most one.

Lemma 2 *In a canonical optimal solution, each ball is adjacent to at most two non-zero balls.*

Proof. Let $B(v, r)$ be a ball in a canonical optimal solution and let $B(v_1, r_1), B(v_2, r_2), \ldots B(v_l, r_l)$ be its adjacent balls such that $r_1 \geq r_2 \geq \cdots \geq r_l$ and assume that $l \geq 3$. We show that we can find a ball with radius $r + r_1 + r_2 + 1$ that covers $B(v, r)$ and all its adjacent balls. As a result, if $r_3 > 0$, we can decrease the number of balls in the canonical optimal solution without increasing the cost, which is a contradiction. Consider the shortest path between v and v_1. Let u be the vertex on

this path that has distance $\min(d(v, v_1), r_1 - r_2)$ from v (see Figure 2). Since the shortest path has length at most $r + 1 + r_1$, the distance of u to v_1 is at most $r + r_1 + 1 - (r_1 - r_2) = r + r_2 + 1$. Clearly, the distance of u to v is at most $r_1 - r_2$.

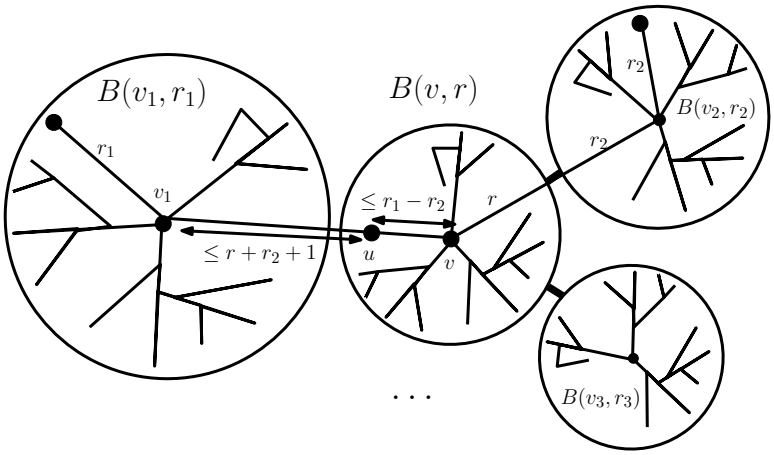

Figure 2: The ball $B(v, r)$ and its adjacent balls. The ball $B(u, r + r_1 + r_2 + 1)$ covers $B(v, r)$ and all its adjacent balls.

We claim that the ball $B(u, r+r_1+r_2+1)$ covers $B(v, r)$ and all its adjacent balls. The distance of u to all the vertices in $B(v_1, r_1)$ is at most $r + r_1 + r_2 + 1$, because the distance of u to v_1 is at most $r + r_2 + 1$ and the distance of v_1 to the vertices in the ball $B(v_1, r_1)$ is at most r_1. The distance of u to all the vertices in $B(v_i, r_i)$ for $i \geq 2$ is at most $r + r_1 + r_2 + 1$, because the distance of u to v is at most $r_1 - r_2$, the distance v to v_i is at most $r + 1 + r_i \leq r + r_2 + 1$, and the distance of v_i to the vertices in the ball $B(v_i, r_i)$ is at most $r_i \leq r_2$. Similarly, the distance of u to all the vertices in $B(v, r)$ is at most $r_1 - r_2 + r \leq r + r_1 + r_2 + 1$. ■

Suppose (G, k) is an instance of the (unweighted) MSR problem where *no zero balls* are allowed. Consider a canonical optimal solution $\mathcal{S}^*$ and suppose that we have $k^* \leq k$ balls in $\mathcal{S}^*$. Since we can run the algorithm for all values $q \leq k$ and take the best solution of all, for simplicity of exposition we assume $k^* = k$. By Lemmas 1 and 2 and because no zero balls are allowed, the balls in $\mathcal{S}^*$ are disjoint and each has at most two adjacent balls. Therefore, the balls in $\mathcal{S}^*$ form a path or cycle. Assume that these balls form a path, say $B_1^*, B_2^*, \ldots, B_k^*$, where each B_i^* is adjacent to B_{i+1}^*, $1 \leq i < k$ (the case of a cycle reduces to the case of a path as we show shortly). We give an exact algorithm for this case. Let $\mathcal{B}$ be the set of all distinct maximal clusters. By Proposition 2: $|\mathcal{B}| \leq n^2$.

The general idea of our algorithm is as follows. For every $V' \subseteq V$, let $G[V']$ denote the induced subgraph of G on V'. Let $G_i = G[\cup_{l=1}^{i} B_l^*]$ (i.e. the subgraph of G induced by the vertices in the first i balls of the path). It is easy to see that for all $1 \leq i \leq k$, the solution $B_1^*, B_2^*, ..., B_i^*$ is an optimal solution for covering G_i with i balls. We present a recursive algorithm, called BestCover, that given a graph H and the number of clusters j, returns a feasible solution, and this feasible solution is optimal when $H = G_i$ and $j = i$, for any $1 \leq i \leq k$. For the moment suppose that the algorithm works as described for values of $j < l$ (for some $l \leq k$). When $j = l$, we guess the ball B_l^* (by enumerating over all possible balls in $\mathcal{B}$) and remove this ball to get a new graph H'. Then, run BestCover with parameters H' and $l-1$ and return the union of the guessed ball and the solution of the recursive call. Note that regardless of whether $H = G_l$ or not, assuming that the recursive call returns a feasible solution for H', we have a feasible solution for H. Furthermore, if $H = G_l$ then for the guessed ball (namely B_l^*), $H' = G_{l-1}$ and the solution returned by the recursive call

is optimal and has the same cost as the solution $\{B_1^*, B_2^*, ..., B_{l-1}^*\}$. Adding the guessed ball B_l^* to the returned solution, we get an optimal solution for G_l.

The difficulty with the above general idea is that even if we use a dynamic programming approach and save the solution of each distinct input of BestCover that we encounter, there may still be exponentially many different inputs corresponding to exponentially many subsets of $V(G)$. We fix this problem with the crucial observation that we are interested to solve only the sub-problems corresponding to graphs G_i. Of course, we do not know $\mathcal{S}^*$ and the subsets $\cup_{l=1}^{i} B_l^*$ in advance, but we show that we can find a polynomial size family of subsets of $V(G)$, called candidate family $\mathcal{F}$, that contains subsets $\cup_{l=1}^{i} B_l^*$ for $1 \leq i \leq k$. Then we only solve the problem for graphs induced by subsets in $\mathcal{F}$, which gives the solution for graph G as well, because $V(G) = \cup_{l=1}^{k} B_l^*$ is in $\mathcal{F}$.

Definition 2 A *candidate family* $\mathcal{F}$, is a set of subsets of $V(G)$ which consists of the following sets: a subset $S \subseteq V(G)$ is in $\mathcal{F}$ if $S = V(G)$ or if there exists a ball B such that $G \setminus B$ has at most two connected components and the set of vertices in one of those components is S.

Lemma 3 *A candidate family can be computed in polynomial time, has at most* $2n^2 + 1$ *members and contains subsets* $\cup_{l=1}^{i} B_l^*$ *for all* $1 \leq i \leq k$.

Proof. Recall that $|\mathcal{B}| \leq n^2$. We remove each of these $|\mathcal{B}|$ balls from G. If the number of connected components is at most two, we add the set of vertices in each component to $\mathcal{F}$. We add $V(G)$ to $\mathcal{F}$ as well. This can be done in polynomial time and we must have considered all members of $\mathcal{F}$ after checking all the balls. The number of subsets obtained this way can be at most $2|\mathcal{B}| + 1 \leq 2n^2 + 1$. When in this process, we remove B_i^* for some $1 < i \leq k$, we get at most two connected components. Therefore, we add the set of vertices $\cup_{l=1}^{i-1} B_l^*$ to $\mathcal{F}$ for all $1 < i \leq k$. Also, we added $V(G) = \cup_{l=1}^{k} B_l^*$ to $\mathcal{F}$ as well. Thus, $\mathcal{F}$ contains subsets $\cup_{l=1}^{i} B_l^*$ for all $1 \leq i \leq k$. ■

The procedure BestCover is presented below.

Algorithm 1 BestCover(H, l)

Input: A graph H and an integer $l > 0$

Output: A subset of at most l balls covering H, which is optimal when $H = G_i$ and $l = i$ for some $1 \leq i \leq k$

1: If Table$[V(H), l] \neq \emptyset$, return Table$[V(H), l]$.
2: If $V(H) = \emptyset$, return $\emptyset$.
3: If $l = 0$ then return "infeasible".
4: Find v the center of H and $r = \text{rad}(H)$.
5: If $l = 1$, return $B(v, r)$.
6: **for all** choices of a ball $B \in \mathcal{B}$ **do**
7: **if** $V(H) \setminus B \in \mathcal{F}$ **then**
8: Store the union of B and the result of BestCover$(H \setminus B, l-1)$ in the set of solutions $\mathcal{C}$.
9: If $\mathcal{C}$ is empty, store $B(v, r)$ in Table$[V(H), l]$ and return it.
10: Choose a solution in $\mathcal{C}$ having the minimum cost, store it in Table$[V(H), l]$ and return it.

We prove the following which is a re-statement of Theorem 1. In this theorem, a canonical optimal solution can form a path or cycle of balls.

Theorem 5 *Algorithm 2 is a polynomial time exact algorithm for the unweighted MSR problem when no singletons are allowed.*

Algorithm 2

Input: A graph G and an integer $k > 0$

Output: A subset $\mathcal{S}^*$ of at most k balls covering G having optimal cost

1: **for all** choices of a ball $B \in \mathcal{B}$ **do**
2: **if** $H = G \setminus B$ is connected **then**
3: Compute a candidate family $\mathcal{F}$ for H as in Lemma 3.
4: Define an array TABLE. For all $S \in \mathcal{F}$ and for all $1 \leq l < k$, set $\text{TABLE}[S, l] = \emptyset$.
5: **for all** $1 \leq q \leq k-1$ **do**
6: Store the union of B and the result of $\text{BESTCOVER}(H, q)$ in the set of solutions $\mathcal{C}$.
7: Let u be a center of G. Add the ball $B(u, \text{rad}(G))$ as a solution to $\mathcal{C}$. Return the minimum cost solution in $\mathcal{C}$.

Proof. First, we show that Algorithm 2 runs in polynomial time. By Lemma 3, we can compute $\mathcal{F}$ in polynomial time and the number of table entries is polynomial. Thus, we just need to prove that the call $\text{BESTCOVER}(H, q)$ runs in polynomial time. We only call $\text{BESTCOVER}(.,.)$ (including in recursive calls) for graphs H with $V(H) \in \mathcal{F}$. Since by Lemma 3, $|\mathcal{F}| \leq 2n^2 + 1$ and $l \leq k \leq n$, the number of distinct instances that are given as parameter to $\text{BESTCOVER}(.,.)$ is polynomial. Also, for a fixed H and l, $\text{BESTCOVER}(H, l)$ makes a total of at most $|\mathcal{B}| \leq n^2$ calls to other instances of $\text{BESTCOVER}(.,.)$. As a result, the overall number of calls of $\text{BESTCOVER}(.,.)$ is polynomial.

Now, we prove the correctness of Algorithm 2. By Lemmas 1 and 2 and because no zero balls are allowed, the balls in a canonical optimal solution form a path or a cycle. Suppose that the set of balls in a canonical optimal solution are $B_1^*, \ldots, B_{k^*}^*$ (for some $k^* \leq k$) where each B_i^* is adjacent to B_{i+1}^*, $1 \leq i < k^*$ and when the balls form a cycle, $B_{k^*}^*$ and B_1^* are also adjacent. If $k^* = 1$, since we consider the best single cluster in Step 7 of the algorithm, we find the optimum solution. Assume $k^* > 1$. Consider the iteration in the main loop that $B = B_{k^*}^*$. Then, there is a canonical optimal solution in $H = G \setminus B$, which forms the path $B_1^*, \ldots, B_{k^*-1}^*$. Consider the iteration in Step 5 where $q = k^* - 1$. We need to prove that $\text{BESTCOVER}(H, q)$ returns an optimal solution in this case.

We prove that $\text{BESTCOVER}(H, l)$ returns a subset of at most l balls covering H, and the cost of this solution is optimal when $H = G_i$ and $l = i$ for all $1 \leq i \leq q = k^* - 1$ (recall that $G_i = G[\cup_{l=1}^{i} B_l^*]$). It is easy to see that $\text{BESTCOVER}(.,.)$ always returns a feasible solution. We prove the rest by induction on l. When $l = 1$, the function returns the ball with minimum radius that covers all the vertices in $V(H)$, which is trivially optimal. In particular, when $H = G_1$ this is a ball with the same radius as B_1^*, and is the optimum solution for H. Assume that when $H = G_{j-1}$ and $l = j - 1$, the function returns an optimal solution and consider the case $H = G_j$ and $l = j$. In Step 6 of Algorithm 1, when B is equal to B_j^*, $V(H) \setminus B$ will be equal to $\cup_{l=1}^{j-1} B_l^* = V(G_{j-1})$. By Lemma 3, the set $\cup_{l=1}^{j-1} B_l^*$ is in $\mathcal{F}$. Thus, in Step 8 of Algorithm 1, $\text{BESTCOVER}(G_{j-1}, j-1)$ will be called and by the induction hypothesis, an optimal solution will be returned having cost the same as cost of $\cup_{l=1}^{j-1} B_l^*$. Therefore, the union of this solution and B_j^* has the same cost as cost of $\cup_{l=1}^{j} B_l^*$ and is optimal. Hence, an optimal cover will be returned in Step 10 of Algorithm 1. ■

Corollary 1 *Given the location of singleton balls, there exists an exact algorithm for the unweighted MSR problem.*

The above corollary shows that we essentially reduced the unweighted MSR problem to the problem of finding the singleton clusters in an optimal solution.

Here, we show that the simple algorithm that returns the best single cluster is a $\frac{3}{2}$-approximation for the unweighted MSR without singletons; this is Theorem 2.

Proof of Theorem 2. By Lemmas 1 and 2 and because no zero balls are allowed, the balls in a canonical optimal solution form a path or a cycle. Suppose that the set of balls in a canonical optimal solution are $B_1^*, \ldots, B_{k^*}^*$ (for some $k^* \leq k$) where each B_i^* is adjacent to B_{i+1}^*, for all $1 \leq i < k^*$ and when the balls form a cycle, $B_{k^*}^*$ and B_1^* are also adjacent. Color all the balls with odd indices with red and all the balls with even indices with blue. Let $\mathrm{OPT} = \mathrm{OPT}_r + \mathrm{OPT}_b$ where OPT_r and OPT_b are the total radii of red and blue balls, respectively. Let n_r and n_b be the number of red and blue balls, respectively. Consider the following two modifications: first increase the radius of red balls by one to get a solution with value at most $\mathrm{OPT}_r + n_r + \mathrm{OPT}_b \leq 2\mathrm{OPT}_r + \mathrm{OPT}_b$. Now, every two adjacent balls intersect and we can merge all the balls into a single ball without increasing the cost (similar to what we did in Lemma 1). By doing the same thing for the blue balls, we can get a single ball with cost at most $\mathrm{OPT}_r + 2\mathrm{OPT}_b$. Therefore, the solution of our algorithm is less than or equal to $\min(2\mathrm{OPT}_r + \mathrm{OPT}_b, \mathrm{OPT}_r + 2\mathrm{OPT}_b) \leq \frac{3}{2}\mathrm{OPT}$. ■

Remark 2 *It is interesting that we cannot beat this simple algorithm by using the natural LP relaxation for this problem as one can show it has an integrality gap of at least essentially $\frac{3}{2}$ [1]*

We conclude this section with a simple observation. If we are given the centers of balls in a canonical optimal solution, but we do not know the radius corresponding to each center then there is a simple 2-approximation algorithm for unweighted MSR. Note that this may be useful for a local search heuristic that given a set of centers swaps each center with another vertex in hope of getting a better solution.

Proposition 4 *There exists a 2-approximation algorithm for the unweighted MSR problem when the centers of balls in a canonical optimal solution is given.*

Proof. Let S be the set of centers which are adjacent to another center. Find the best single ball in each component of $G \setminus S$ and return these balls and zero balls centered around the vertices in S as our solution. This algorithm is a 2-approximation. Notice that if two centers are adjacent in a canonical solution, both of them must be centers of zero balls. Consider an optimal solution and increase the size of all its non-zero balls by one. This at most doubles the cost of solution and makes all non-zero balls intersect their adjacent balls. This also covers all zeros ball not in S, because they must be adjacent to some non-zero ball. After merging all the intersecting balls, we obtain a solution with cost at most twice of the optimum and a single ball in each component of $G \setminus S$. Clearly, the cost of our solution is not more than the cost of this solution, which completes the proof. ■

4 PTAS for Euclidean MSD

In this section, we present a PTAS for the MSD problem in $\mathbb{R}^2$. Throughout this section, we assume that $\epsilon > 0$ is a given fixed constant. We build upon the framework of Gibson *et al.* [10] and introduce some new ideas to make it work for the MSD problem. Recall that Gibson *et al.* present an exact algorithm (that we call GKKPV) for the MSR problem restricted to Euclidean plane. Since our algorithm follows similar steps as the GKKPV algorithm for MSR, we first give a brief overview of that algorithm along with the necessary lemmas for its proof of correctness.

4.1 The GKKPV Algorithm for Euclidean MSR

Consider an instance of the Euclidean MSR problem which consists of a set of points V on the plane along with an integer k. Here, the distance between any pair of points is the Euclidean distance of

these points in the plane. Let $\mathcal{D}$ be the set of distinct maximal clusters, more sepecifically the set of discs with a center $p \in V$ and radius $\|p - q\|$ for some $q \in V$. Note that by Proposition 2, we have $|\mathcal{D}| \leq n^2$.

The high level description of the algorithm is as follows. An axis parallel rectangle is called *balanced* if the ratio of its width to length is at least $1/3$. A balanced rectangle containing a set R of points is minimal if at least three of its sides contain a point of R. The GKKPV algorithm is a dynamic programming algorithm which uses balanced rectangles to define the sub-problems (i.e. the set of points enclosed in a balanced rectangle to be covered with a given number of discs). A *separator* for a (balanced) rectangle is any line which is perpendicular to its longer side and cuts it in the middle third of its longer side. The algorithm starts with a rectangle containing all the points, denoted by R_0, and cuts it into two smaller rectangles by selecting a separator line and solves the sub-problems corresponding to smaller rectangles recursively.

A vertical or horizontal line is called *critical* if it either goes through a point $p \in V$ or if it is tangent to some disk in $\mathcal{D}$. It is not hard to see that all vertical lines between two consecutive critical vertical lines intersect the same set of discs. Thus, it is enough to fix an arbitrary vertical line between any two consecutive critical vertical lines, one to the left of the leftmost critical vertical line and one to the right of the rightmost critical vertical line and only consider these lines as potential vertical separators. We can also do the same for the horizontal lines to reduce the number of horizontal separators. Thus, there are only $\Theta(n^2)$ vertical or horizontal lines to consider as separators. Let $L(R)$ denote the separators of a rectangle R from these fixed set of lines and the leftmost and rightmost separators of it.

The algorithm has a recursive procedure $\mathrm{DC}(R, \kappa, T)$ shown below, which takes as input a rectangle R, an integer $\kappa \geq 0$ and a subset $T \subseteq \mathcal{D}$ and computes an optimum MSR solution using at most κ discs for the set of points in $Q = \{q : q \in V \cap R,\ q \text{ is not covered by } T\}$. The idea is that T is the set of discs in the optimal solution that intersect R and are chosen in the higher levels of recursion. The algorithm calls $\mathrm{DC}(R_0, k, \emptyset)$ to find the best cover for V. The value of the sub-problem for a recursive call is stored in a dynamic programming table $\textsc{Table}[V \cap R, \kappa, T]$. They prove that we only need to consider $|T| \leq \beta = 424$ to get an optimal solution (we explain this below); this combined with the fact that the number of distinct $V \cap R$ is $O(n^4)$ and $\kappa \in O(n)$, implies that the size of the dynamic programming table (and hence sub-problems to compute) is $O(n^{2\beta+5})$, which is polynomially bounded. In the following algorithm, we assume that I is a dummy disc of radius ∞.

For the proof of correctness of GKKPV, the authors of [10] prove the following lemmas. The first one is used to show that in Step 8, it is sufficient to only consider subsets of size at most 12.

Lemma 4 (Lemma 2.1 in [10]) *If R is a rectangle containing a set of points $P \subseteq V$ and $\mathcal{O}$ is an optimum solution for P, then there is a separator for R that intersects at most 12 discs in $\mathcal{O}$.*

The next lemma essentially bounds the number of large discs of optimum intersecting a rectangle and is used to show that it is sufficient to only consider the choices of T_1 and T_2 as in Step 11. The proof of this lemma is based on the fact that the centers of discs of an optimum solution cannot contain the centers of other discs; so you cannot pack too many of them close to each other.

Lemma 5 (Lemma 2.2 in [10]) *If $\mathcal{O}$ is an optimum solution for a set of points $P \subseteq \mathbb{R}^2$ and R is an arbitrary rectangle of length $a > 0$, then the number of discs in $\mathcal{O}$ of radius at least a that intersect R is at most 40.*

The following lemma puts a lower bound on the distance of separator lines of nested rectangles. Its proof follows from the definition of separator lines.

Algorithm 3 [GKKPV] Recursive Clustering : $\mathrm{DC}(R, \kappa, T)$

Input: Balanced rectangle R, integer κ, and constant size subset $T \subset \mathcal{D}$ of optimum intersecting R selected so far.

Output: Find best cover for points in R not covered by those in T using at most κ clusters.

If $\textsc{Table}[V \cap R, \kappa, T]$ is created then return its value; otherwise create it.
Let $Q = \{q : q \in V \cap R,\ q \text{ is not covered by } T\}$. If $Q = \emptyset$ then $\textsc{Table}[V \cap R, \kappa, T] \leftarrow \emptyset$ and return $\emptyset$.
If $\kappa = 0$, let $\textsc{Table}[V \cap R, \kappa, T] \leftarrow \{I\}$ (infeasible) and return $\{I\}$.
If $|Q| = 1$, let $\textsc{Table}[V \cap R, \kappa, T]$ be the solution with a singleton cluster and return its value.
Let R' be a minimal balanced rectangle containing $V \cap R$ (this rectangle always can be found in polynomial time).
Initialize $\mathcal{D}' \leftarrow \{I\}$.
for all choices $\ell \in L(R')$ **do**
  for all choices of a set $\mathcal{D}_0 \subseteq \mathcal{D}$ of size at most 12 that intersect ℓ **do**
    for all choices of $\kappa_1, \kappa_2 \geq 0$ with $\kappa_1 + \kappa_2 + |\mathcal{D}_0| \leq \kappa$ **do**
      Let R_1 and R_2 be the two rectangles obtained from cutting R' by ℓ. Let $T_1 = \{D \in T \cup \mathcal{D}_0 :\ D \text{ intersects } R_1\}$ and $T_2 = \{D \in T \cup \mathcal{D}_0 :\ D \text{ intersects } R_2\}$.
      if $|T_1| \leq \beta$ and $|T_2| \leq \beta$ **then**
        Recursively call $\mathrm{DC}(R_1, \kappa_1, T_1)$ and $\mathrm{DC}(R_2, \kappa_2, T_2)$.
        if $\mathrm{cost}(\mathcal{D}_0 \cup \textsc{Table}[V \cap R_1, \kappa_1, T_1] \cup \textsc{Table}[V \cap R_2, \kappa_2, T_2]) < \mathrm{cost}(\mathcal{D}')$ **then**
          Update $\mathcal{D}' \leftarrow \mathcal{D}_0 \cup \textsc{Table}[V \cap R_1, \kappa_1, T_1] \cup \textsc{Table}[V \cap R_2, \kappa_2, T_2]$.
Assign $\textsc{Table}[V \cap R, \kappa, T] \leftarrow \mathcal{D}'$ and return its value.

Lemma 6 (Lemma 3.1 in [10]) *Let* $\mathrm{DC}(R_i, \cdot, \cdot)$ *be an ancestor of* $\mathrm{DC}(R_j, \cdot, \cdot)$ *in the recursion tree. Let* a *be the length of* R_j*. Let* ℓ_i *be the separator that resulted in the recursion subtree that contains* $\mathrm{DC}(R_j, \cdot, \cdot)$ *and let* ℓ_j *be an arbitrary separator for* R_j*. If both of these separators are vertical or horizontal, then the distance between them is at least* $a/3$*.*

The main part of the proof of correctness is Lemma 3.2 in [10] which inductively proves the correctness of procedure $\mathrm{DC}(R, \kappa, T)$. Let OPT be an optimal solution. The heart of this proof is the induction step. For that, suppose that $T \subseteq \mathrm{OPT}$ contains every cluster of OPT that contains a point in $V \cap R$ as well as a point in $V \setminus R$, OPT' is the set of clusters of optimum that have a point in $Q = \{q : q \in V \cap R,\ q \text{ is not covered by } T\}$, and $\kappa \geq |\mathrm{OPT}'|$. Note that the sole purpose of OPT' is to cover Q and hence, OPT' is an optimum cover for Q. From Lemma 4, R' has a separator ℓ' that intersects a set $\tilde{\mathcal{D}}_0 \subseteq \mathcal{D}$ of at most 12 clusters of OPT'. Without loss of generality, assume ℓ' is vertical. We can move ℓ' (to left or right) to become a line $\ell \in L(R')$ that intersects the same set $\tilde{\mathcal{D}}_0$. Consider the choice of $\mathcal{D}_0 = \tilde{\mathcal{D}}_0$. Let OPT_1 and OPT_2 be the clusters of OPT' to the left and right of ℓ and consider choices of $\kappa_1 = |\mathrm{OPT}_1|$ and $\kappa_2 = |\mathrm{OPT}_2|$. Let R_1, R_2, T_1, T_2 be exactly as in the algorithm.

Then, the proof of correctness follows inductively. One can show that, by the induction hypothesis, $\mathrm{DC}(R_1, \kappa_1, T_1)$ and $\mathrm{DC}(R_2, \kappa_2, T_2)$ return an optimal solution. To complete the proof of correctness, we only need to prove that $|T_1| \leq \beta$ and $|T_2| \leq \beta$. The proof for $|T_1| \leq \beta$ is as follows. The proof for T_2 is analogous.

Every disc in T_1 comes from a guessed $\mathcal{D}_0$ for a separator in a higher level of the nested recursive calls. We partition the separators of higher level recursive calls, into two groups and bound the number of discs from each set. Assume R_1 is a rectangle of length a. Let $\bar{R}$ be a square of side $5a$ centered at the center of R_1. Consider the separators of higher level calls that do not intersect

$\bar{R}$. A disc that intersects these separators and R_1 has radius at least a and by Lemma 4, there can be only 40 of them in T_1, because $T_1 \subseteq \text{OPT}$. Now, consider the rest of separators. These separators intersect $\bar{R}$. By Lemma 6, it can be seen that there are at most $5a/(a/3) + 1 = 16$ vertical (horizontal) such separators. Therefore, there are at most $32 \times 12 = 384$ discs coming from these separators in T_1 that can intersect R_1. Thus, we must have $|T_1| \leq 384 + 40 = 424 = \beta$.

4.2 A PTAS for Euclidean MSD

Let $\text{CH}(C)$ be the convex hull of the points in a cluster C in Euclidean MSD. Consider two clusters C_1 and C_2 of an optimum solution. If $\text{CH}(C_1)$ and $\text{CH}(C_2)$ intersect, we can replace the two clusters with one containing all the points in $C_1 \cup C_2$ without increasing the total diameter. Thus, without loss of generality, we can focus on the solutions in which the convex-hulls of their clusters are disjoint. Therefore, each cluster can be unambiguously defined by its convex-hull. This convex-hull belongs to the set of convex polygons having all corners from V. Let $\mathcal{P}$ be the set of these convex polygons. For this reason, from now on, when we say a *cluster*, we consider a convex polygon in $\mathcal{P}$.

There are three main difficulties in extending the arguments of GKKPV for MSR to MSD. First, as we have seen before in the MSR problem, one can bound the number of distinct maximal clusters that can appear in a solution by n^2. This allows one to enumerate over all possible clusters (and more generally over all constant size subsets of clusters) that appear in an optimal solution. This fact is critically used in the GKKPV algorithm. For the MSD problem, we do not have such a nice (i.e. polynomially bounded) characterization of clusters that can appear in an optimum solution. To overcome this obstacle, we show that for every cluster C, there is a cluster C' enclosing it whose diameter is at most a factor $(1 + \epsilon)$ of the diameter of C, and C' belongs to a polynomial size set of polygons. The critical property of C' is that it is simpler to describe: it is determined by $O(1/\epsilon)$ points of input.

The second difficulty in adapting the GKKPV algorithm is that they show one cannot have too many large clusters close to a rectangle of comparable length in any optimum solution (Lemma 5). To prove this, they use the simple fact that no disc (cluster) in an optimum solution can contain the center of another disc (cluster) as otherwise one can merge the two clusters into one with smaller total radius. In the MSD problem, clusters are not necessarily defined by a disc and there is no notion of center here. We can potentially have many thin (e.g. non-fat triangles) and non-overlapping polygons of an optimum solution inside a region. We can still argue that in an optimum solution, we cannot have too many large clusters in a bounded region but the packing argument is different from that of [10] (see Lemma 10). Third, when we fix the first two issues, some parts of the correctness proof fail and we need to prove some extra properties to handle this new difficulty.

We start by stating a special case of Jung's theorem about the minimum enclosing ball of a set of points in $\mathbb{R}^n$ (*i.e.*, the special case that $n = 2$) and a definition that will be used later.

Lemma 7 *[13] A convex polygon in $\mathbb{R}^2$ with diameter D has an enclosing circle with radius at most $\sqrt{3}D/3$.*

Definition 3 A *σ-restricted polygon* is a convex polygon with at most σ sides such that each side contains (not necessarily at the corners) at least two points of V.

We emphasize that each corner of a σ-restricted polygon is not necessarily a point of V; it is obtained from the intersection of two lines each of which contains at least two points of V. We need the following lemma. It was brought to our attention that this lemma (and in fact its extensions

to higher dimensions) could be implied from Theorem 4.1 of [7]. We present our own proof here for completeness.

Lemma 8 *For any convex polygon P and fixed $\epsilon > 0$, there is a $\sigma(\epsilon)$-restricted polygon Q with $\sigma(\epsilon) = 13 + \frac{4\sqrt{3}\pi}{3}\frac{1}{\epsilon} \approx 13 + \frac{7.26}{\epsilon}$, satisfying the following properties:*

1. *Q contains P,*
2. *$\text{diam}(Q) \leq (1+\epsilon)\,\text{diam}(P)$,*
3. *For any vertical or horizontal line ℓ, ℓ intersects P if and only if ℓ intersects Q.*

Proof. The sides of Q are extensions of a subset of sides of P that we choose. In other words, we choose a subset of sides of P, consider the straight lines extending these sides and the convex polygon defined by these lines is the polygon Q. The number of sides of Q is exactly equal to the size of the chosen subset of sides of P. Clearly, if we choose Q this way, it encloses P and therefore contains P. We show there is a subset of size $\sigma(\epsilon)$ of sides of P, call it S, such that the resulting polygon Q satisfies Properties 2 and 3 as well.

Let us order the vertices of P in clock-wise order $u_1, u_2, \ldots$ (starting from the left-most corner of P). We traverse the sides of P in clockwise order and choose some of them along the way, starting with selecting u_1u_2, and select only a small number of sides in total. In order to satisfy Property 3, we also make sure that the left-most, top-most, right-most, and bottom-most corners of P are selected to be in Q (along with the two sides defining those corners of P). So first, we choose the (at most) 8 sides defining each of these (at most) 4 corners of P and let S consists of these sides. The remaining sides added to Q are obtained by traversing on the sides of P (in clock-wise order) starting from u_1u_2 and we add them (if not already among those 8 sides) according to the following rules.

Let D be $\text{diam}(P)$. Suppose the last side chosen was u_iu_{i+1}. We call u_ju_{j+1} ($j \geq i+1$) *dismissible* with respect to u_iu_{i+1} if it satisfies both of the following conditions (see Figure 3):

- its angle, say θ, with u_iu_{i+1} is less than $\pi/2$,
- $\text{dist}(u_{i+1}, u_j) \leq \epsilon D/2$.

Let j be the largest index such that u_ju_{j+1} is dismissible with respect to the last chosen side. Then, we add this side to S, and continue until we arrive at u_1u_2. We let Q be the convex polygon defined by the set of straight lines obtained from the sides in S. Note that Q satisfies Property 3.

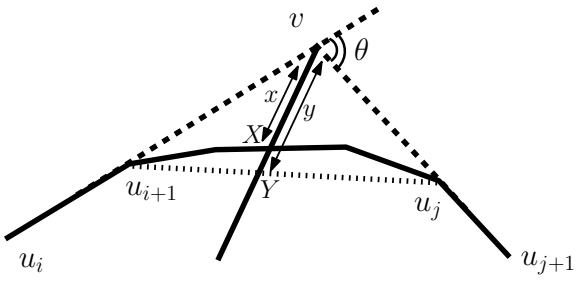

Figure 3: The solid sides u_iu_{i+1} and u_ju_{j+1} belong to P. The extension of sides u_iu_{i+1} and u_ju_{j+1} in polygon Q are shown with dashed lines. Line ℓ is shown with a solid line. The part of ℓ outside P has length x which is less than the length of side $u_{i+1}u_j$.

Let R be the perimeter of P. By Lemma 7, P can be enclosed by a circle of perimeter $2\sqrt{3}\pi D/3$ and hence, we have $R \leq 2\sqrt{3}\pi D/3$. It is not hard to see that we choose at most $2\pi/(\pi/2)$ sides, because of the first condition in the definition of a dismissible side and we choose at most $R/(\epsilon D/2)$ sides, because of the second condition. Therefore, we choose a total of at most $1 + 4 + \frac{4\sqrt{3}\pi}{3}\frac{1}{\epsilon}$ sides in our traversal. Given that we add at most 8 sides for the left-most, top-most, right-most, and bottom-most vertices of P, we have $|S| \leq 13 + \frac{4\sqrt{3}\pi}{3}\frac{1}{\epsilon}$.

It only remains to show that $\text{diam}(Q) \leq (1+\epsilon)D$. It is enough to show that for any two corners of Q their distance is at most $(1+\epsilon)D$. Consider two arbitrary corners of Q, say v and w, and the line segment ℓ connecting these points. At most two segments of ℓ may be outside of P and this happens when both v and w are not corners of P. The section of ℓ that lies inside P has clearly length at most D. We prove that each of the (at most) two segments of ℓ which lies in between P and Q has length at most $\epsilon D/2$. Let us assume that v is the corner obtained from the two lines that contain sides u_iu_{i+1} and u_ju_{j+1} of P and consider the segment of ℓ that has v as an end-point (see Figure 3). Let X be the other end-point of it, which is a cross point of P and ℓ. We also name the cross-point of vw and $u_{i+1}u_j$ point Y. Let x be the length of vX and let y be the length of vY. Because of the convexity of P, we must have $y \geq x$. Consider the triangle $u_{i+1}vu_j$. Since the angle of lines u_iu_{i+1} and u_ju_{j+1}, θ, is less than $\pi/2$, the angle $\widehat{u_{i+1}vu_j} \geq \pi/2$, which implies $u_{i+1}u_j$ is the longest side of triangle $u_{i+1}vu_j$. We know that the length of $u_{i+1}u_j$ is at most $\epsilon D/2$. We also know that a segment enclosed in a triangle has length at most equal to the longest side of the triangle. Thus, we must have $y \leq \epsilon D/2$ which implies $x \leq \epsilon D/2$, as wanted. ■

Let OPT denote an optimal solution. The above lemma implies that for every cluster P of OPT, there is a $\sigma(\epsilon)$-restricted polygon Q containing all the points of P such that $\text{diam}(Q) \leq (1+\epsilon)\,\text{diam}(P)$. Thus, if we replace each cluster P with the set of points in the corresponding $\sigma(\epsilon)$-restricted polygon (breaking the ties for the points that belong to different $\sigma(\epsilon)$-restricted polygons arbitrarily), we find a $(1+\epsilon)$-approximate solution. This enables us to use essentially the same Algorithm 3 (*i.e.*, GKKPV algorithm), where we enumerate over $\sigma(\epsilon)$-restricted polygons (instead of all clusters), which is a polynomially bounded set of polygons.

Let R_0 be a rectangle containing all the points of V. First, we define a $\Theta(n)$ size set of separators. The notion of a balanced rectangle and a separator line is the same. A vertical (horizontal) line is called *critical* if it passes through a point in the input. It is not hard to see that all vertical (horizontal) lines between two consecutive critical vertical (horizontal) lines are equivalent in the sense that they intersect the same set of clusters (recall that the clusters are convex polygons). Choose an arbitrary vertical (horizontal) line between each consecutive vertical (horizontal) critical lines. As before, let $L(R)$ show the separators of a rectangle R from these chosen lines and the leftmost and rightmost separators of it.

We apply the following modifications to Algorithm 3 to obtain our PTAS (Algorithm 4). We substitute $\mathcal{D}$ with $\mathcal{P}$, which is the set of all $\sigma(\epsilon)$-restricted polygons, *i.e.*, for each $\sigma(\epsilon)$-restricted polygon Q, the set of points in Q forms a cluster in $\mathcal{P}$. By definition of a $\sigma(\epsilon)$-restricted polygon, $|\mathcal{P}| \in O(n^{2\sigma(\epsilon)})$ which is a polynomial in the size of input for a fixed $\epsilon > 0$.

Note that for each cluster $P \in \mathcal{P}$, there is a cluster $Q \in \mathcal{P}$ with $\text{diam}(Q) \leq (1+\epsilon) \cdot \text{diam}(P)$, by Lemma 8. We fix an arbitrary such cluster Q and call it the representative of P and denote it by $\text{Rep}(P)$. Note that (again by Lemma 8):

Corollary 2 *A vertical (or horizontal) line intersects a cluster P if and only if it intersects* $\text{Rep}(P)$. *In particular for every axis-parallel rectangle R, P intersects R if and only if* $\text{Rep}(P)$ *intersects R.*

Although we have an exponential number of clusters in $\mathcal{P}$, we have a polynomially bounded size set of representatives. For a set of clusters $\tilde{\mathcal{P}}$, we use $\text{Rep}(\tilde{\mathcal{P}})$ to denote the set of clusters in $\mathcal{P}$

Algorithm 4 Recursive Clustering for Euclidean MSD: DC(R, κ, T)

Input: Balanced rectangle R, integer κ, and constant size subset $T \subset \mathcal{D}$ of optimum intersecting R selected so far.

Output: Find best cover for points in R not covered by those in T using at most κ clusters.

1: If TABLE$[V \cap R, \kappa, T]$ is created then return its value; otherwise create it.
2: Let $X = \{x : x \in V \cap R,\ x \text{ is not covered by } T\}$. If $X = \emptyset$ then TABLE$[V \cap R, \kappa, T] \leftarrow \emptyset$ and return $\emptyset$.
3: If $\kappa = 0$, let TABLE$[V \cap R, \kappa, T] \leftarrow \{I\}$ (infeasible) and return $\{I\}$.
4: If $|X| = 1$, let TABLE$[V \cap R, \kappa, T]$ be the solution with a singleton cluster and return its value.
5: Let R' be a minimal balanced rectangle containing $V \cap R$.
6: Initialize $\mathcal{P}' \leftarrow \{I\}$.
7: **for all** choices $\ell \in L(R')$ **do**
8: **for all** choices of a set $\mathcal{P}_0 \subseteq \mathcal{P}$ of size at most 4 that intersect ℓ **do**
9: **for all** choices of $\kappa_1, \kappa_2 \geq 0$ with $\kappa_1 + \kappa_2 + |\mathcal{P}_0| \leq \kappa$ **do**
10: Let R_1 and R_2 be the two rectangles obtained from cutting R' by ℓ. Let $T_1 = \{Q \in T \cup \mathcal{P}_0 :\ Q \text{ intersects } R_1\}$ and $T_2 = \{Q \in T \cup \mathcal{P}_0 :\ Q \text{ intersects } R_2\}$.
11: **if** $|T_1| \leq \beta$ and $|T_2| \leq \beta$ **then**
12: Recursively call DC(R_1, κ_1, T_1) and DC(R_2, κ_2, T_2).
13: **if** cost$(\mathcal{P}_0 \cup$ TABLE$[V \cap R_1, \kappa_1, T_1] \cup$ TABLE$[V \cap R_2, \kappa_2, T_2]) <$ cost$(\mathcal{P}')$ **then**
14: Update $\mathcal{P}' \leftarrow \mathcal{P}_0 \cup$ TABLE$[V \cap R_1, \kappa_1, T_1] \cup$ TABLE$[V \cap R_2, \kappa_2, T_2]$.
15: Assign TABLE$[V \cap R, \kappa, T] \leftarrow \mathcal{P}'$ and return its value.

that are representative of clusters in $\tilde{\mathcal{P}}$. In particular, Rep(OPT) are the representative clusters of OPT.

We keep the same dynamic programming table, TABLE$[V \cap R, \kappa, T]$, which stores a cover with at most κ clusters of $\mathcal{P}$ having cost within $(1+\epsilon)$ factor of the optimum cover for the set of points $\{q : q \in V \cap R, \text{q is not covered by } T\}$. We only consider sets $T \subseteq \mathcal{P}$ with size at most $\beta \leq 83$. Similar to GKKPV algorithm, we later show that considering sets of at most this size is enough to get a solution within factor $1+\epsilon$ of the optimum value. As a result, the size of the table is polynomially bounded. Also, we substitute $\mathcal{D}_0$ with a set $\mathcal{P}_0 \subseteq \mathcal{P}$ in line 8 and change the bound of 12 for $|\mathcal{D}_0|$ to a bound of 4 for $|\mathcal{P}_0|$. This comes from Lemma 9 which is the equivalent version of Lemma 4 for the Euclidean MSR problem. In addition, we use $\beta = 83$ for the bounds of $|T_1|$ and $|T_2|$ in line 11. Also, Lemma 10 is the equivalent version of Lemma 5.

Lemma 9 *Suppose R is a rectangle containing a set of points $V' \subseteq V$ and $\mathcal{O}$ is an optimum solution of MSD on V'. Let $\mathcal{P}'$ be the set of representatives of clusters of $\mathcal{O}$. Then there is a separator ℓ for R such that it intersects at most 4 clusters of $\mathcal{O}$ (and their representatives in $\mathcal{P}'$).*

Proof. Let a be the length of the longer side of R. If one chooses a separator ℓ uniformly at random, then the probability that ℓ intersects a cluster $P \in \mathcal{O}$ is at most $\text{diam}(P)/(a/3)$ (recall that the separators are in the middle one third of a rectangle). So the expected number of clusters of $\mathcal{O}$ that intersect ℓ is $\sum_{P \in \mathcal{O}} 3\,\text{diam}(P)/a \leq 3 \cdot \text{cost}(\mathcal{O})/a \leq 3\sqrt{2} < 4.25$, because $\text{cost}(\mathcal{O}) \leq \sqrt{2}a$ as they all are inside R. So there is a separator ℓ that intersects at most 4 clusters of $\mathcal{O}$, and by Corollary 2, it intersects at most 4 representative clusters of them in $\mathcal{P}'$. ∎

Lemma 10 *A rectangle R of length a intersects at most 3 clusters of* OPT *with diameter at least a.*

Proof. Let OPT$'$ be the set of clusters of OPT with diameter at least a that are intersecting R. By way of contradiction, assume that C_1, C_2, C_3, and C_4 are four clusters of OPT$'$ with diameters

d_1, d_2, d_3, and d_4 each intersecting R. Without loss of generality, assume that $d_1 \geq d_2 \geq d_3 \geq d_4 \geq a$. This implies that the distance of any two points $u, v \in C_1 \cup C_2 \cup C_3 \cup C_4$ is at most $d_1 + d_2 + \sqrt{2}a$ since the total distance from u and v to R is at most $d_1 + d_2$ and the diameter of R is at most $\sqrt{2}a$. But $d_1 + d_2 + \sqrt{2}a < d_1 + d_2 + d_3 + d_4$, so if one replaces these 4 clusters of OPT with one single cluster (containing all those points), the total diameter decreases, which is a contradiction. ■

Corollary 3 *A rectangle R of length a intersects at most 3 clusters, which are representatives of clusters of diameter at least a in* OPT.

Note that Lemma 6 still holds here as it is based on the properties of balanced rectangles and separator lines, which we did not change. We prove the following Lemma which is the equivalent version of Lemma 3.2 in [10].

Lemma 11 *Suppose that* $\mathrm{DC}(R, \kappa, T)$ *is called at some level of recursion by top-level invocation to* $DC(R_0, k, \emptyset)$ *in Algorithm 4. Let* $T' \subseteq \mathrm{OPT}$ *be those clusters of* OPT *that have a point in both* $V \cap R$ *as well as a point in* $V \setminus R$ *and suppose* $T = \mathrm{Rep}(T')$. *Also, let* $X' = \{x : x \in V \cap R,\ x \text{ is not covered by } T'\}$ *and* $X = \{x : x \in V \cap R,\ x \text{ is not covered by } T\}$, *and* OPT' *be the set of clusters of* OPT *that contain a point in* X'. *Suppose that* $\kappa \geq |\mathrm{OPT}'|$. *Then after calling* $\mathrm{DC}(R, \kappa, T)$, $\textsc{Table}(V \cap R, \kappa, T)$ *contains a* κ*-cover for* X *whose cost is at most* $(1+\epsilon)\cdot\mathrm{cost}(\mathrm{OPT}')$.

Proof. The proof is similar to that of Lemma 3.2 in [10] and is by induction on $|V \cap R|$. The proof of base cases is straightforward. When $|X| < 2$, the reader can verify that the algorithm returns an optimal solution in the lines 2 to 4.

Consider the case that $|X| \geq 2$, and recall that R' is a balanced rectangle over points of $V \cap R$. Without loss of generality, assume the separators of R' are vertical. First note that OPT' is an optimum solution for X' with $|\mathrm{OPT}'|$ clusters since each point in $(V \cap R) \setminus X'$ is covered by T'. Therefore, using Lemma 9, R' has a separator ℓ' that intersects at most 4 clusters of OPT'; let S be that set of clusters of OPT' and let $\tilde{\mathcal{P}}_0 = \mathrm{Rep}(S)$. We can move ℓ' to be one of the separators in $L(R')$; so we consider the choice of $\ell \in L(R')$ in the algorithm that intersects only set S of those in OPT' and consequently, only $\tilde{\mathcal{P}}_0$.

Consider the iterations of the loops in lines 7 and 8 where ℓ is chosen as above and $\mathcal{P}_0$ is chosen to be $\tilde{\mathcal{P}}_0$, and κ_1 and κ_2 are the numbers of clusters of OPT' to the left and right of ℓ, respectively. Similarly, we denote the subsets of OPT' to the left and right of ℓ by OPT'_1 and OPT'_2, respectively. We define $T'_1 = \{C \in T' \cup S :\ C \text{ intersects } R_1\}$ and $T'_2 = \{C \in T' \cup S :\ C \text{ intersects } R_2\}$. Note that from the definitions of T, T', S, $\mathcal{P}_0$, and Corollary 2, it follows that $T_1 = \mathrm{Rep}(T'_1)$ and $T_2 = \mathrm{Rep}(T'_2)$. Observe that $T'_1 \subseteq \mathrm{OPT}$ and T'_1 contains every cluster of OPT that contains a point in both $V \cap R_1$ as well as a point in $V \setminus R_1$. Also, OPT'_1 is the set of clusters of OPT' that contain points in $X'_1 = \{x : x \in V \cap R_1,\ x \text{ is not covered by } T'_1\}$ and $|\mathrm{OPT}'_1| = \kappa_1$. Let us define $X_1 = \{x : x \in V \cap R_1,\ x \text{ is not covered by } T_1\}$. Thus, we can use induction hypothesis on the call to $\mathrm{DC}(R_1, \kappa_1, T_1)$ so that the entry $\textsc{Table}[V \cap R_1, \kappa_1, T_1]$ contains a κ_1-cover of X_1 of cost at most $(1+\epsilon)\mathrm{cost}(\mathrm{OPT}'_1)$. Similarly, after call $\mathrm{DC}(R_2, \kappa_2, T_2)$, the entry $\textsc{Table}[V \cap R_2, \kappa_2, T_2]$ contains a κ_2-cover of X_2 of cost at most $(1+\epsilon)\mathrm{cost}(\mathrm{OPT}'_2)$, where $X_2 = \{x : x \in V \cap R_2,\ x \text{ is not covered by } T_2\}$. Thus, the clusters in $\mathcal{P}_0 \cup \textsc{Table}[V \cap R_1, \kappa_1, T_1] \cup \textsc{Table}[V \cap R_2, \kappa_2, T_2]$ form a κ-cover of X with cost at most $(1+\epsilon)\mathrm{cost}(S) + (1+\epsilon)(\mathrm{cost}(\mathrm{OPT}'_1) + \mathrm{cost}(\mathrm{OPT}'_2)) \leq (1+\epsilon) \cdot \mathrm{cost}(\mathrm{OPT}')$.

It only remains to show that $|T_1|, |T_2| \leq 83$ and as a result, we actually call $\mathrm{DC}(R_1, \kappa_1, T_1)$ and $\mathrm{DC}(R_2, \kappa_2, T_2)$. The argument for this is similar to that in Lemma 3.2 of [10] whose outline was given at the end of Section 4.1. We prove this for $|T_1|$ and the case of $|T_2|$ is analogous. Suppose we have called $\mathrm{DC}(\tilde{R}_0, \cdot, \cdot), \ldots, \mathrm{DC}(\tilde{R}_t, \cdot, \cdot)$ before arriving at instance (R, κ, T), where $\tilde{R}_t = R$. Then it follows that $T_1 \subseteq \bigcup_{j=0}^{t} \mathcal{P}_0(\tilde{R}_j)$. Assuming that R_1 in the algorithm has length a, we take $\bar{R}$ to be

the square of side $3a$ with the same center as R_1. By Lemma 6, the number of vertical (horizontal) separators $\ell(\tilde{R}_0), \ldots, \ell(\tilde{R}_t)$ that intersect $\bar{R}$ is at most $(3a)/(a/3)+1 = 10$ (as they are at least $a/3$ apart). Therefore, using Lemma 9, the number of clusters of T_1 that belong to $\mathcal{P}_0(\tilde{R}_j)$ (for those j's) is at most $4 \times 20 = 80$, *i.e.*, 4 for each of 20 vertical and horizontal separators that intersect $\bar{R}$. Now we bound the number of clusters of T_1 that belong to a $\mathcal{P}_0(\tilde{R}_j)$ where $\ell(\tilde{R}_j)$ does not intersect $\bar{R}$. These clusters must have diameter at least a as they intersect both R_1 as well as $\ell(\tilde{R}_j)$. From Corollary 3, the number of such clusters is bounded by 3. Thus, we have $|T_1| \leq 83$. ■

It follows from Lemma 11 that after termination of $\mathrm{DC}(R_0, k, \emptyset)$, the entry $\textsc{Table}[V, k, \emptyset]$ contains a $(1+\epsilon)$-approximate solution for V. This completes the proof of Theorem 3.

Remark 3 *We believe that this algorithm can be extended to three-dimensional Euclidean space or any fixed dimension Euclidean space. The extension of all lemmas (except Lemma 8) would be similar to the extension of [10] for the MSR problem. The extension of Lemma 8 to higher dimensions would be based on Theorem 4.1 of [7].*

5 MSD with Constant k

Recall that Hansen and Jaumard [11] presented an exact algorithm for the MSD problem when $k = 2$, but the case of constant $k > 2$ has been open. The reader may ask why their algorithm fails to generalize to this case. Let us consider the case of $k = 3$. The first step of their algorithm is to guess the diameter of the clusters, which we can do for the case of $k = 3$ in polynomial time, too. Then, they reduce the problem of finding a feasible solution with these guessed diameters to a 2-SAT problem. They use a Boolean variable to represent the cluster of each vertex and for any pair of vertices u and v, they add between zero to two clauses to the SAT formula based on the distance between u and v. For $k = 3$, the variable that shows the cluster of each vertex should have 3 values. Therefore, if we try to model the problem this way, we need to solve a constraint satisfaction problem which is hard in general case.

Our approach is as follows. Define a *canonical optimal solution* as before, *i.e.,* among the optimal solutions, a *canonical optimal* solution is a solution with the minimum possible number of clusters. We denote the set of all the distinct distances between vertices in the input graph by $\mathcal{D}$. We have the following trivial observations:

Observation 1 *Let C_1^* and C_2^* be two clusters in a canonical optimal solution. There is a vertex v in C_1^* and a vertex u in C_2^* such that* $\mathrm{dist}(u, v) > \mathrm{diam}(C_1^*) + \mathrm{diam}(C_2^*)$.

Observation 2 *The size of $\mathcal{D}$ is at most n^2. In addition, the diameter of any cluster in an optimal solution is in $\mathcal{D}$.*

The general idea of the algorithm (Algorithm 5) is as follows. Since the size of $\mathcal{D}$ is polynomially bounded, we can guess the diameters of clusters in an optimal solution (done in Line 3 of Algorithm 5). Consider a canonical optimal solution and two clusters C_1^* and C_2^* in it with diameters D_1 and D_2 that we have already guessed. By Observation 1, there is a vertex v in C_1^* and a vertex u in C_2^* such that $\mathrm{dist}(u, v) > \mathrm{diam}(C_1^*) + \mathrm{diam}(C_2^*)$. Denote by V_1 and V_2 the set of vertices with distance at most D_1 from v and distance at most D_2 from u, respectively. The crucial observations are that C_1^* and C_2^* are subsets of V_1 and V_2, respectively and V_1 and V_2 are disjoint. As a result, if we guess the vertices u and v (done in Line 4) and compute V_1 and V_2, we have separated the set of vertices that can be in C_1^* and C_2^*. If we do the same process for all pairs of clusters, we can separate the vertices that can be in a specific optimal cluster from the vertices in other optimal clusters, which is equivalent to finding the clusters in an optimal solution.

Algorithm 5 BESTCOVER(G, k)

Input: A graph G and an integer $k > 0$
Output: An optimal clustering $\mathcal{C}$

1: Initialize $\mathcal{C}$ with the cluster containing all the vertices.
2: **for all** $2 \leq q \leq k$ **do**
3: **for all** choices of q values $D_1, \ldots, D_q$ from $\mathcal{D}$ **do**
4: **for all** choices of q groups $S_1, \ldots S_q$ of at most $q-1$ vertices each **do**
5: **for all** $1 \leq i \leq q$ **do**
6: Compute V_i the set of vertices with distance at most D_i from all the vertices in S_i.
7: **if** there are two vertices in V_i with distance more than D_i **then**
8: Discard this choice of $S_1, \ldots, S_q$ and go to Step 4.
9: **if** $\cup_{i=1}^{q} V_i \neq V$ **then**
10: Discard this choice of $S_1, \ldots, S_q$ and go to Step 4.
11: **if** $\sum_{i=1}^{q} D_i$ is less than cost of $\mathcal{C}$ **then**
12: Update $\mathcal{C}$ with the solution $V_1, \ldots, V_q$.
13: **return** $\mathcal{C}$.

The correctness of this algorithm is stated in the following theorem, which implies Theorem 4.

Theorem 6 *Algorithm 5 returns an optimal solution for the MSD problem in time $n^{O(k^2)}$, which is polynomial for any fixed k.*

Proof. We first show that the algorithm runs in time $n^{O(k^2)}$. Assume we know the distances between all the pairs of vertices and the numbers in set $\mathcal{D}$ by running an all-pairs-shortest-path algorithm in a preprocessing step. There are $O(k)$ choices in Step 2. In Step 3, we choose at most k values from the set $\mathcal{D}$. By Observation 2, there are $O(n^{2k})$ choices for this step. In Step 4, we choose at most $k(k-1)$ vertices from V. There are $O(n^{k(k-1)})$ choices for this step. It is not hard to see that the body of the for-loop in Step 4 takes $O(kn^2)$. As a result, the final running time is in $O(k^2 n^{k^2+k+2})$, which is a polynomial for constant k.

One can verify that when the algorithm is in Step 11, $V_1, \ldots, V_q$ is a feasible solution with cost $\sum_{i=1}^{q} D_i$. Therefore, we just need to prove that for some choices in the for-loops, we encounter an optimal solution in Step 11. Consider a canonical optimal solution $\mathcal{C}^* = \{C_1^*, \ldots, C_{k^*}^*\}$, with $k^* \leq k$. Consider the iteration in Step 2 that $q = k^*$. By Observation 2, we have $\text{diam}(C_i^*) \in \mathcal{D}$ for all $1 \leq i \leq q$. Consider the iteration in Step 3 that $D_i = \text{diam}(C_i^*)$ for all $1 \leq i \leq q$. By observation 1, for all $1 \leq i \neq j \leq q$, there are vertices in C_i^* and C_j^*, say $v_j^{(i)}$ and $v_i^{(j)}$, respectively, such that $\text{dist}(v_j^{(i)}, v_i^{(j)}) > D_i + D_j$. Consider the iteration in Step 4 that $S_i = \{v_j^{(i)} : 1 \leq j \leq q, j \neq i\}$. We shortly prove that for each $1 \leq i \leq q$, set V_i computed from this choice of S_i is equal to C_i^* and as a result, for the above mentioned choices in the for-loops, we encounter an optimal solution in Step 11.

Clearly, we have $S_i \subseteq \mathcal{C}_i^*$ and hence, each vertex in $\mathcal{C}_i^*$ has distance at most D_i from all the vertices in S_i. Thus, we must have $\mathcal{C}_i^* \subseteq V_i$. In addition, for any $i \neq j$, we must have $V_i \cap V_j = \emptyset$. The reason is that if there is at least a vertex, say u, in the intersection of these sets, by definition of V_i and V_j, we have $\text{dist}(v_j^{(i)}, u) \leq D_i$ and $\text{dist}(v_i^{(j)}, u) \leq D_j$. Therefore, we must have $\text{dist}(v_j^{(i)}, v_i^{(j)}) \leq D_i + D_j$, which is a contradiction. We claim that $V_i \setminus C_i^* = \emptyset$ and in other words, $V_i = C_i^*$. Assume there is at least one vertex in $V_i \setminus C_i^*$, say u. Let C_j^* be a cluster covering u where trivially $j \neq i$. Clearly, we have $u \in V_j$ that means $u \in V_i \cap V_j$, which is a contradiction. ■

6 Concluding Remarks

We presented an exact algorithm for the unweighted MSR problem when no singleton clusters are permitted. To get the exact algorithm, we proved that in the absence of zero balls, each connected component of the input graph is covered by a path or cycle of balls in a canonical optimal solution. By using this structural property, we designed an exact dynamic programming algorithm that runs in polynomial time.

The difficult core of the MSR problem is to find the location of zero balls. If we can find these locations even approximately, our exact algorithm solves the rest of the problem and at least, we can hope for a good approximation algorithm for this case. The current best ratio even for unweighted graphs is 3.504, which is the same for the general metrics. To solve the difficulty with zero balls, one might try the widely used LP-based and local search techniques. Unfortunately, the natural ways to utilize these techniques do not seem promising as both the LP and any standard local search approach have large gaps [1]. Even for the case that there are no zero balls, the gap of standard LP is at least $\frac{3}{2}$ [1]. It seems that to design better approximation for MSR, one needs to introduce substantially new techniques.

We also presented a PTAS for the Euclidean MSD problem that partially answers the open question asked by Doddi *et al.* [5], but the complexity of the Euclidean MSD problem remains open. Before our algorithm, the best approximation ratio for this problem was 2. Finally, we settled another open question of Doddi *et al.* [5] regarding the complexity of the MSR problem when k is constant. We presented a polynomial time exact algorithm for this case, while the best previous result was a 2-approximation algorithm.

There are still several interesting questions left open related to the considered problems. Perhaps the most interesting question is to obtain a PTAS for general MSR. The existence of a QPTAS for this problem [10] is a strong evidence that there could be a PTAS. Also, given a fixed set of centers, we gave a 2-approximation algorithm for the problem of assigning a radius to each center to minimize the total radii. It is an interesting question to find an algorithm with an improved ratio for this case. For MSD on Euclidean metrics, the complexity of the problem is open. We suspect that unlike MSR (which has an exact algorithm on Euclidean metrics), the Euclidean MSD problem is NP-hard. Finally, we presented an exact algorithm for the MSD problem with constant k, but the running time is too high even for small k. It will be useful if one can find a more efficient algorithm for this case.

Acknowledgment: We would like to thank two anonymous referees for their great comments and suggestions, especially for bringing to our attention the connection of Lemma 8 and [7].

References

[1] Babak Behsaz. *Approximation Algorithms for Clustering Problems.* PhD thesis, University of Alberta, Department of Computing Science, 2012.

[2] Vasilis Capoyleas, Gunter Rote, and Gerhard Woeginger. Geometric Clusterings. *Journal of Algorithms*, 12(2):341–356, 1991.

[3] M Charikar and R Panigrahy. Clustering to Minimize the Sum of Cluster Diameters. In *Proceedings of the 33rd Annual ACM Symposium on Theory of Computing*, pages 1–10, 2001.

[4] Moses Charikar and Rina Panigrahy. Clustering to minimize the sum of cluster diameters. *Journal of Computer and System Sciences*, 68(2):417–441, 2004.

[5] Srinivas Doddi, Madhav V Marathe, S S Ravi, David Scot Taylor, and Peter Widmayer. Approximation algorithms for clustering to minimize the sum of diameters. *Nordic J. of Computing*, 7(3):185–203, 2000.

[6] Srinivas Doddi, Madhav V Marathe, S S Ravi, David Scot Taylor, and Peter Widmayer. Approximation algorithms for clustering to minimize the sum of diameters. In *SWAT '00: Proceedings of the 3rd Scandinavian workshop on Algorithm Theory*, pages 237–250, 2000.

[7] R.M. Dudley. Metric entropy of some classes of sets with differentiable boundaries. *Journal of Approximation Theory*, 10:227–236, 1974.

[8] Matt Gibson, Gaurav Kanade, Erik Krohn, Imran A Pirwani, and Kasturi Varadarajan. On Metric Clustering to Minimize the Sum of Radii. In *SWAT '08: Proceedings of the 11th Scandinavian workshop on Algorithm Theory*, pages 282–293, Berlin, Heidelberg, 2008. Springer-Verlag.

[9] Matt Gibson, Gaurav Kanade, Erik Krohn, Imran A Pirwani, and Kasturi Varadarajan. On Metric Clustering to Minimize the Sum of Radii. *Algorithmica*, 57(3):484–498, 2010.

[10] Matt Gibson, Gaurav Kanade, Erik Krohn, Imran A Pirwani, and Kasturi Varadarajan. On Clustering to Minimize the Sum of Radii. *SIAM Journal on Computing*, 41(1):47–60, 2012.

[11] P Hansen and B Jaumard. Minimum sum of diameters clustering. *Journal of Classification*, 4(2):215–226, 1987.

[12] Dorit S Hochbaum and David B Shmoys. A Best Possible Heuristic for the k-Center Problem. *Mathematics of Operations Research*, 10(2):180–184, 1985.

[13] Heinrich W E Jung. Über die kleinste Kugel, die eine räumliche Figur einschliesst. *J. reine angew. Math.*, 123:241–257, 1901.