Improved FPT Approximation for Sum of Radii Clustering with Mergeable Constraints

Sayan Bandyapadhyay
Portland State University, OR, USA

Tianzhi Chen
Portland State University, OR, USA

Abstract

In this work, we study k-min-sum-of-radii (k-MSR) clustering under mergeable constraints. k-MSR seeks to group data points using a set of up to k balls, such that the sum of the radii of the balls is minimized. A clustering constraint is called mergeable if merging two clusters satisfying the constraint, results in a cluster that also satisfies the constraint. Many popularly studied constraints are mergeable, including fairness constraints and lower bound constraints.

In our work, we design a $(4+\epsilon)$-approximation for k-MSR under any given mergeable constraint with runtime $2^{O(\frac{k}{\epsilon} \cdot \log^2 \frac{k}{\epsilon})} n^4$, i.e., fixed-parameter tractable in k for constant ϵ. Our result directly improves upon the FPT $(6+\epsilon)$-approximation by Carta et al. [10]. We also provide a hardness result that excludes the exact solvability of k-MSR under any given mergeable constraint in time $f(k)n^{o(k)}$, assuming ETH is true.

2012 ACM Subject Classification Theory of computation → Approximation algorithms analysis

Keywords and phrases sum-of-radii clustering, mergeable constraints, approximation algorithm

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2025.23

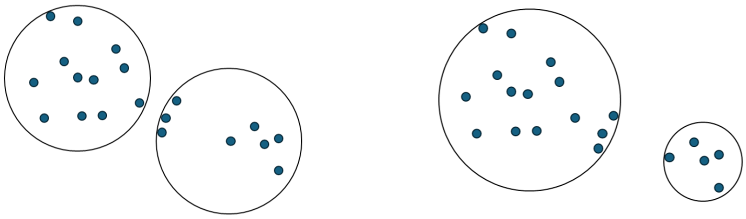
Category APPROX

Funding This work was supported by the National Science Foundation under Grant No. AF 2311397.

1 Introduction

Given a set of points in a metric space and an integer $k > 0$, the *k-min-sum-of-radii* clustering problem (k-MSR for short) seeks to group the points using a set of at most k balls, called clusters, each centered at a point, minimizing the sum of the radii of those balls. Like most clustering objectives, finding an optimal k-MSR clustering is NP-hard, as shown by Gibson et al. [21]. They also designed a quasi-polynomial-time approximation scheme (QPTAS) for k-MSR in the same work, implying it is unlikely that k-MSR is APX-hard under standard complexity-theoretic assumptions. Consequently, substantial attention has been directed toward designing approximation algorithms. Currently, the best-known polynomial-time approximation factor is $3+\epsilon$ [9].

To highlight the motivation for studying k-MSR, we compare it to k-center and k-median, two popular clustering objectives. k-center is similar to k-MSR, but instead of minimizing the sum of the radii of balls, it minimizes the radius of the largest ball. Simple 2-approximation algorithms exist for k-center [22, 23], and variants of k-center are relatively easy to solve. However, as a trade-off, k-center is prone to outliers, as well as artifacts such as the *dissection effect* shown in Figure 1, where clusters might be split unnecessarily. k-median, on the other hand, seeks to minimize the sum of each point's distance to the closest center, rather than the radii of a cluster. k-median and its variants are relatively more difficult to solve, but its robustness has given rise to its popularity [3, 16]. Informally, the k-MSR problem lies between k-center and k-median – it reduces the dissection effect seen in k-center, while often allowing for simpler solutions than those typically admitted by k-median, which motivates our study.



(a) k-center.

(b) k-MSR.

Figure 1 An example of the dissection effect in k-center. In Figure 1a, k-center splits a group of data that should have been in the left cluster, and assign them to the cluster on the right in effort to decrease the maximum ball size. In Figure 1b, k-MSR mitigates the issue and provides better separation between clusters.

k-MSR clustering has also been studied under additional constraints, popularly known as constrained variants. In our work, we focus on specific types of constraints, namely *mergeable constraints*. A clustering constraint is said to be *mergeable* if merging two clusters that satisfy the constraint, results in a cluster that also satisfies the constraint. The property has proven useful for analyzing constrained k-MSR clustering, with a number of works dedicated to it [10, 18, 6]. A popular class of mergeable constraints includes many *fairness constraints*. First introduced by Chierichetti et al. [15], given a partition of the input dataset into groups based on some attribute (commonly referred to as colors), fair clustering aims to ensure that each group in the dataset is proportionally represented within every cluster. For instance, given a set of red and blue points, fair clustering seeks to have a balanced number of red and blue points in each cluster. Carta et al. showed that many common fairness constraints are mergeable [10]; we discuss these constraints further in Section 1.2. There are also non-fair constraints that are mergeable, such as the *lower bound* constraint, where each cluster is required to have at least L points, for some given $L > 0$ [1].

Constrained versions of k-MSR are often difficult to approximate and lack tractability in polynomial time. A potential approach to address this issue is the design of *fixed-parameter tractable* (FPT) algorithms. FPT algorithms still have a runtime polynomial in the input size but do not have to be polynomial in their parameters. For example, an algorithm running in $O(n^2 \cdot 2^k)$ time is considered to be FPT in parameter k. Fixed-parameter tractability essentially allows efficient computation, assuming the parameters are small in practice. For challenging constrained clustering problems such as constrained k-MSR, FPT algorithms are often the preferred approach, as they offer computational tractability and can yield improved approximation guarantees. In our work, we also adopt this approach to address the complexity of the problem.

1.1 Our Contributions

Our work establishes two main results concerning k-MSR under mergeable constraints: one demonstrating that the problem admits a $(4+\epsilon)$-approximation in FPT time in k for constant ϵ, and another proving its computational hardness. We begin by presenting the following theorem.

▶ **Theorem 1.** *For any $\epsilon > 0$, there is a $(4 + \epsilon)$-approximation algorithm for the k-MSR problem under mergeable constraints that runs in $2^{O(\frac{k}{\epsilon} \cdot \log^2 \frac{k}{\epsilon})} n^4$ time.*

Our result directly improves upon the FPT $(6+\epsilon)$-approximation by Carta et al. [10]. Moreover, to solve k-MSR under a given mergeable constraint, their algorithm requires the existence of a constant-factor approximation algorithm for k-center under the same constraint, while ours does not have this restriction. There are constraints under which no constant-factor approximation is known for k-center. For instance, no approximation algorithm is known for k-center under what is called fair representational constraint, which is one of the more general definitions of fair constraints. Our algorithm also has the advantage of generality. While the FPT algorithms by Drexler et al. [18] and Banerjee et al. [6] achieve an approximation factor of $(1+\epsilon)$, they are limited to specific metric spaces such as Euclidean spaces or those with bounded doubling dimension. In contrast, our algorithm applies to general metric spaces.

Next, we give the following theorem regarding the hardness of k-MSR clustering under the fair representational constraint.

▶ **Theorem 2.** *Fair Representational Clustering cannot be solved in time $f(k)n^{o(k)}$ unless ETH is false.*

This result also trivially extends to k-MSR under mergeable constraints, as the fair representational constraint is known to be mergeable [18].

1.2 Related Work

Gibson et al. showed that the k-MSR objective is NP-hard, even in planar graphs and metrics with doubling dimensions [21]; consequently, a line of work gave approximation algorithms for the problem. Charikar and Panigrahy first gave a polynomial time 3.504-approximation for the problem in their seminal work [12]. This result remained the best approximation until Friggstad and Jamshidian followed up with a 3.489-approximation nearly twenty years later [20]. Soon after, Buchem et al. improved the approximation factor to $(3+\epsilon)$, which is the current best known in polynomial time [9].

k-MSR has been studied under various constraints, most notably the capacitated constraint and the matroid constraint. A line of work on devising $O(1)$-approximation for capacitated k-MSR led to an FPT 3-approximation [24, 19, 25, 5]. k-MSR has also been studied under matroid constraints, leading to an FPT $(9+\epsilon)$ approximation by Inamdar and Varadarajan [24] and an improved $(3+\epsilon)$ approximation by Chen et al. [13]. Polynomial time $O(1)$-approximations for these variants of k-MSR are not known. Interestingly, polynomial time $O(1)$-approximations exist for k-MSR with lower bound [2, 9].

Another common k-MSR constraint to study is the mergeable constraint. Most of the studies are geared towards fairness constraints, since many fairness constraints are mergeable. Drexler et al. first developed an FPT $(1+\epsilon)$-approximation algorithm for k-MSR under mergeable constraints in Euclidean metrics of arbitrary dimension with constant k [18]. The result was subsequently extended to metrics with bounded doubling dimension by Banerjee et al. [6]. Carta et al. gave a $(6+\epsilon)$-approximation algorithm for k-MSR under mergeable constraints in general metrics [10].

Fair k-MSR has also received attention from researchers. Carta et al. showed that several fairness constraints are mergeable [10]. We name a few of these constraints that are commonly used. The simplest definition is called *exact fairness* [8], where each cluster must have exactly the same number of points from each color. The next step up is Pairwise Fair Clustering (or sometimes called (t,k)-fair clustering) [15]. Given a set of red points and blue points and integer $t>0$, Pairwise Fair Clustering ensures that the ratio between blue and red points in each cluster is between t and $\frac{1}{t}$. This definition is sometimes extended to have more than two colors. Another common fairness definition is called Fair Representational Clustering [7, 8], where the fraction of points in some color i allowed in each cluster is at most α_i and at least β_i, given parameters $\alpha_i, \beta_i \in [0, 1]$. Bandyapadhyay et al. gave polynomial-time $O(1)$-approximation algorithms for k-MSR under the Pairwise Fair Clustering constraint with two groups and exact fair constraint for $l \geq 2$ groups [4]. Chen et al. developed an FPT $(3+\epsilon)$-approximation algorithm for k-MSR under an alternate notion of fairness that is a special case of matroid k-MSR [13]. In their definition of fairness, each color is allowed to have a pre-defined number k_i of points that can be chosen as cluster centers. Banerjee et al. [6] also gave an FPT $(1+\epsilon)$-approximation algorithm for the same problem in Euclidean metric. For more works on fair clustering, one can refer to the survey by Chhabra et al. [14].

Roadmap. We first define some notations in Section 2, then describe our algorithm in Section 3, and finally prove our hardness result in Section 4.

2 Preliminaries

In k-MSR clustering, we are given a set P of n points in a metric space with metric d. The goal is to find (i) a set of up to k points $C = \{c_1, \ldots, c_k\} \subset P$ called *centers*, and (ii) a function ϕ assigning each point $p \in P$ to some center $c_i \in C$. We call (C, ϕ) a *clustering* of P. For each $c_i \in C$, we call $\mathcal{C}_i = \phi^{-1}(c_i)$ the *cluster* centered at c_i, and we define the *radius* corresponding to c_i as $r_i = \max_{p:\phi(p)=c_i} d(p, c_i)$. We refer to the sum of radii of a clustering (C, ϕ) as the *cost* of the clustering, and we denote it by $\mathrm{cost}(C, \phi)$; we use the same notation $\mathrm{cost}(R)$ to denote a sum of a set of radii R, or $\mathrm{cost}(\mathcal{C})$ for a set of clusters $\mathcal{C}$.

We *merge* two clusters $C_i = \phi^{-1}(c_i), C_j = \phi^{-1}(c_j)$ by setting $\phi(p) = c$ for all points $p \in C_i \cup C_j$ and some $c \in P$. We say that a constraint is *mergeable* if for any two clusters satisfying the constraint, merging the clusters results in a new cluster that also satisfies the constraint. We refer to a clustering that satisfies some (mergeable) constraint as a *constrained* clustering, and a clustering without any constraint as a *vanilla* clustering. We say that a clustering (C, ϕ) is *feasible* if every point in P is assigned to some center, $|C| \leq k$, and each cluster satisfies the desired constraint (if any). Moreover, we say that a clustering (C, ϕ) is *optimal* if it is feasible, and $\mathrm{cost}(C, \phi)$ is minimized over all such clustering.

We denote a ball with center c and radius r by $B(c, r)$. Given a point p and a ball $B(c, r)$, we say that p is *contained* in $B(c, r)$ if $d(p, c) \leq r$, denoted by $p \in B(c, r)$; note that a point p may be contained in a ball $B(c, r)$, but not assigned to its center c, as p may be contained in multiple balls but only assigned to one cluster.

3 An FPT $(4+\epsilon)$-approximation Algorithm

In this section, we describe an FPT $(4+\epsilon)$-approximation algorithm for k-MSR under mergeable constraints. We first describe the algorithm at a high level, then state the procedures that it calls, present the algorithm, and lastly analyze it.

Our algorithm stems from the idea that we can find a vanilla clustering and expand the balls such that each cluster of any constrained clustering is contained within one of the expanded balls. This allows us to transform a vanilla clustering into a constrained clustering in FPT time in k. Our algorithm has four major steps: (1) Guess the radii of an optimal vanilla clustering up to a $(1+\epsilon)$ factor. (2) Compute a $(2+\epsilon)$-approximate vanilla clustering using the guessed radii, (3) Expand the vanilla clusters, resulting in a $(3+\epsilon)$-approximate set of balls, and (4) Merge the expanded balls to create clusters that satisfy the given mergeable constraint. These steps are accomplished through calls to a series of procedures, which we describe in the following subsections.

Similar to our algorithm, the algorithm of Carta et al. [10] also finds a $(3+\epsilon)$-approximate set of balls with the aforementioned containment property. However, our algorithm finds the balls in a simpler and more general manner in that it does not rely on a k-center algorithm under the same constraint to obtain the balls. Carta et al. also made the same observation as ours that one can satisfy the mergeable constraint by merging the $(3+\epsilon)$-approximate balls, but our approach differs in how we choose the new center for each set of balls being merged. The algorithm of Carta et al. picks the center of the largest ball, which results in a 2 factor increase in cost, but we show that one can always pick a center that results in at most a $\frac{4}{3}$ factor increase in cost.

3.1 Finding a Vanilla Clustering

We first describe a procedure that takes a radii vector $(r_1, \ldots, r_l)$, and a multiplicity vector $(k_1, \ldots, k_l)$ as input. Moreover, if there is a feasible vanilla clustering that uses those radii with the respective multiplicities, the procedure outputs a vanilla clustering that uses those radii values up to a factor of 2 with the same multiplicities. We make use of the fact shown in Observation 3 that for any point p in a cluster $\mathcal{C}_i$ with radius r_i, we can cover all points in the cluster with a ball centered at p with a radius of $2 \cdot r_i$. This allows us to recursively find a point p that is not yet covered, and guess the cluster containing p and the corresponding radius to cover all points of that cluster. We note that our procedure is essentially the same as the procedure used by Chen et al. in their work on k-MSR [13], but restated to take in a pair of radii and multiplicity vectors as an additional input; this allows us to use the same set of radii and multiplicity vectors in both the vanilla clustering step and the cluster expansion step of our algorithm. The procedure is given as Algorithm 1.

Algorithm 1 Vanilla-Clustering(P, R_i, K_i).

Input : A set of points P, a radii vector $R_i = (r_1, \ldots, r_l)$, and a multiplicity vector $K_i = (k_1, \ldots, k_l)$.
Output : A vanilla clustering (C_v, ϕ_v).

for *each* $r_j \in R_i$ *where* $k_j > 0$ **do**
  $C_j \leftarrow \emptyset$
  $K'_i \leftarrow K_i$ with k_j decremented by 1
  $P' \leftarrow P \setminus \{q \in P \mid d(p,q) \leq 2 \cdot r_j\}$
  if $P' \neq \emptyset$ **then**
    $(C_j, \phi_j) \leftarrow$ Vanilla-Clustering(P', R_i, K'_i)
  else
    flag $\leftarrow$ true
  $C_j \leftarrow C_j \cup \{p\}$ $\phi_j(p) \leftarrow \{q \in P \mid d(p,q) \leq 2 \cdot r_j\}$
// All recursive calls failed to cover P'
if *flag* $==$ *false* **then**
  return
$(C_v, \phi_v) \leftarrow$ a (C_j, ϕ_j) with the smallest cost from Loop 1.
return (C_v, ϕ_v)

▶ **Observation 3.** *For any cluster $\mathcal{C}_i$ with center c and radius r, we can cover all points in $\mathcal{C}_i$ using a ball centered at some point $p \in C$ with radius $2 \cdot r$.*

Proof. Consider the point p and some other point $p' \in \mathcal{C}_i$. By triangle inequality, we have that $d(p,p') \leq d(p,c) + d(p',c)$. Furthermore, since $p,p' \in \mathcal{C}_i$, $d(p,c), d(p',c) \leq r$. It follows that $d(p,p') \leq 2 \cdot r$ for any $p,p' \in \mathcal{C}_i$, therefore we can cover all points in C with any point $p \in \mathcal{C}_i$ and a radius $2 \cdot r$. ◀

▶ **Lemma 4.** *Suppose there is a feasible vanilla clustering of P that uses exactly k_j balls of radius r_j for $1 \leq j \leq l$. Then Algorithm 1 outputs a feasible vanilla clustering of P that uses at most k_j balls of radius $2 \cdot r_j$ for $1 \leq j \leq l$.*

Proof. Let $\mathcal{C}^* = \{\mathcal{C}_1^*, \ldots, \mathcal{C}_k^*\}$ be a set of feasible vanilla clusters corresponding to the radii vector R_i. We consider the following construction of (C_v, ϕ_v). Let p be some point in P. Note that p is in some $\mathcal{C}_i^* \in \mathcal{C}^*$ with radius r_j for some j. We add p to C_v as a new center and set $\phi_v(q) = p$ for all points q such that $d(p,q) \leq 2 \cdot r_j$, and remove p and all such q from P. We also remove the cluster $\mathcal{C}_i^*$ from consideration. We can do this because by Observation 3 we have that p with a radius of $2 \cdot r_j$ covers all points in $\mathcal{C}_i^*$. We repeat this process until there are no points left in P. We can always find a clustering within k iterations this way because each time we repeat the process, we cover all points in some cluster $\mathcal{C}_i^*$. Specifically, there exists a recursion branch in Algorithm 1 that follows this construction. It follows that Algorithm 1 outputs the desired vanilla clustering of P. ◀

▶ **Lemma 5.** *Given a vector R_i with $O(\log_{1+\epsilon} \frac{k}{\epsilon})$ radii, Algorithm 1 runs in $(O(\log_{1+\epsilon} \frac{k}{\epsilon}))^k n$ time.*

Proof. First, we consider the depth of the recursive calls in Algorithm 1. Since each clustering is restricted to having a maximum of k clusters, and each recursive call decreases the number of clusters by one, each branch of recursive calls will have at most k levels of recursion. Now, we consider how many additional calls each recursive call makes. By our assumption, we have $O(\log_{1+\epsilon}(\frac{k}{\epsilon}))$ radii in R_i. Since each recursive call makes an additional call for each $r_j \in R_i$ with $k_j > 0$, each recursive call can make $O(\log_{1+\epsilon}(\frac{k}{\epsilon}))$ additional calls. Since each call can be computed in $O(n+k)$ time, it follows that Algorithm 1 runs in $(O(\log_{1+\epsilon} \frac{k}{\epsilon}))^k n$ time. ◀

3.2 Expanding the Vanilla Clustering

Given a feasible vanilla clustering, we describe a process to expand the vanilla balls so that each cluster of a fixed, constrained clustering is contained in one of the expanded balls. The idea is to guess how much to expand the vanilla clusters using a guessed set of radii for the constrained clustering. In particular, for each vanilla cluster $\mathcal{C}_v^i$, we guess the biggest constrained cluster whose center is contained in $\mathcal{C}_v^i$, and expand $\mathcal{C}_v^i$ by its radius, thereby fully containing all the constrained clusters whose centers are in $\mathcal{C}_v^i$. We assume that the radii vector $R_i = (r_1, \ldots, r_l)$ and a multiplicity vector $K_i = (k_1, \ldots, k_l)$ of the constrained clustering are given. The procedure is given as Algorithm 2.

Now we show some properties of the output of Algorithm 2.

▶ **Lemma 6.** *Given a vanilla clustering with balls $B_1(c_1, r_1^v), \ldots, B_k(c_k, r_k^v)$, and the radii vector $R_i = (r_1, \ldots, r_l)$ and a multiplicity vector $K_i = (k_1, \ldots, k_l)$ of a fixed constrained clustering, the output R_e of Algorithm 2 contains a vector $(r_1^c, \ldots, r_k^c)$ such that each cluster of the constrained clustering is contained in $B(c_j, r_j^c)$ for some $1 \leq j \leq k$. Moreover, $\sum_{j=1}^k r_j^c \leq \sum_{j=1}^k r_j^c + \sum_{t=1}^l k_t \cdot r_t$.*

Algorithm 2 Expand-Balls(C_v, ϕ_v, R_i, K_i).

Input : A vanilla clustering (C_v, ϕ_v), a radii vector $R_i = (r_1, \ldots, r_l)$, and a multiplicity vector $K_i = (k_1, \ldots, k_l)$.
Output : A set of guesses R_e for expanded radii.

1 $R_v \leftarrow \{r_1^v, \ldots, r_k^v\}$, the radii of (C_v, ϕ_v)
2 $R_e \leftarrow \emptyset$
3 $E \leftarrow \{(r_1^e, \ldots, r_k^e) \mid r_j^e \in \{r_1, \ldots, r_l\} \cup \{0\}$ and the number of $r_j^e = r_x$ does not exceed k_x, for each $r_x \in R_i$ and $k_x \in K_i\}$
4 **for** *each* $(r_1^e, \ldots, r_k^e) \in E$ **do**
5   Add $(r_1^e + r_1^v, \ldots, r_k^e + r_k^v)$ to R_e
6 **return** R_e

Proof. Let $\mathcal{C}^* = \{\mathcal{C}_1^*, \ldots, \mathcal{C}_k^*\}$ be a fixed constrained clustering that for each $1 \le t \le l$ uses k_t clusters of radii r_t. Consider the following construction of a vector $(r_1^c, \ldots, r_k^c)$. For each center $c_j \in C_v$ with radius r_j^v, let $\mathcal{C}_{\psi(j)}^*$ be the largest cluster in $\mathcal{C}^*$ whose center is in the vanilla cluster $\mathcal{C}_v^j$, and let $r_{\psi(j)}^*$ be its radius. We set each r_j^c to $r_j^v + r_{\psi(j)}^*$. Note that the expanded ball $B(c_j, r_j^c)$ contains any cluster in $\mathcal{C}^*$ whose center is in $\mathcal{C}_v^j$, satisfying the desired property. As the center of each constrained cluster in $\mathcal{C}^*$ is contained in a unique vanilla cluster, the radius of the constrained cluster is used at most once to expand the radius of that vanilla cluster. In particular, each $r_t \in R_i$ is used at most k_t times for the expansion. It follows that $(r_1^c, \ldots, r_k^c)$ is in R_e. For the same reason, $\sum_{j=1}^k r_j^c = \sum_{j=1}^k (r_j^v + r_{\psi(j)}^*) \le \sum_{j=1}^k r_j^v + \sum_{t=1}^l k_t \cdot r_t$, which proves the moreover part. ◀

Now we show the following two lemmas about the output size and runtime of Algorithm 2.

▶ **Lemma 7.** *Given the vectors R_i, K_i of size $O(\log_{1+\epsilon} \frac{k}{\epsilon})$, the number of guesses $|R_e|$ that Algorithm 2 outputs is bounded by $(O(\log_{1+\epsilon} \frac{k}{\epsilon}))^k$.*

Proof. Algorithm 2 computes R_e by enumerating a set of guesses E for how much to expand each vanilla cluster. There are k vanilla clusters, and there are $O(\log_{1+\epsilon} \frac{k}{\epsilon})$ ways to expand each cluster, since there are $O(\log_{1+\epsilon} \frac{k}{\epsilon})$ distinct radii in R_i. It follows that the number of entries in E and R_e are bounded by $(O(\log_{1+\epsilon} \frac{k}{\epsilon}))^k$. ◀

Since each entry in E takes $O(k)$ time to compute, we have the following lemma.

▶ **Lemma 8.** *Given the vectors R_i, K_i of size $O(\log_{1+\epsilon} \frac{k}{\epsilon})$, Algorithm 2 runs in $(O(\log_{1+\epsilon} \frac{k}{\epsilon}))^k$ time.*

3.3 Merging Balls

Suppose we are given a set of balls that cover a set of points P. Moreover, we assume that there is a clustering of P with a mergeable constraint such that each cluster is contained in one of the given balls. We design a procedure that helps find a clustering of P with the mergeable constraint such that its cost is at most $4/3$ times the sum of the radii of the given balls.

Before describing the algorithm, we define a few notations. Let $B = \{B_1, \ldots, B_k\}$ denote a set of balls. Let $G = (B, E)$ be a graph with the balls in B as vertices and the set of edges $E = \{(B_i, B_j) \mid B_i, B_j \in B$ and there is a point such that $p \in B_i$ and $p \in B_j\}$. A *connected component* of G therefore contains a maximal subset of balls where every pair is connected by a path.

The merging algorithm, Algorithm 3, is quite simple. Given a set of balls B that cover a set of points P, Algorithm 3 first construct the graph $G = (B, E)$ as defined above. For each connected component of balls B' in G, Algorithm 3 finds a point c contained in the balls of B' such that assigning all other points contained in B' to c results in a cluster whose radius is minimized.

Algorithm 3 Merge-Balls(B).

Input : A set of balls B.
Output : A clustering (C, ϕ) of P.

$E \leftarrow \{(B_i, B_j) \mid B_i, B_j \in B$ and there is a point such that $p \in B_i$ and $p \in B_j\}$
$G \leftarrow (B, E)$
$C \leftarrow \emptyset$
for *each connected component with set of balls* $B' = \{B'_1, \ldots, B'_l\}$ *in* G **do**
  $c \leftarrow \arg\min_p max_q d(p, q)$, where p, q are points in some balls in B'
  $C \leftarrow C \cup \{c\}$
  for *each* $q \in B'_i$*, for some* $B'_i \in B'$ **do**
    $\phi(q) \leftarrow c$
return (C, ϕ)

In the following, we show that the output clustering (C, ϕ) of Algorithm 3 satisfies a mergeable constraint based on an additional assumption.

▶ **Lemma 9.** *Suppose there is a clustering $\mathcal{C}$ of P with a mergeable constraint such that each cluster is contained in a ball of B. Then the output clustering (C, ϕ) of Algorithm 3 satisfies the mergeable constraint.*

Proof. Since each cluster in $\mathcal{C}$ is in a ball of B, the union of the balls in each connected component of G is the disjoint union of a subset of clusters of $\mathcal{C}$. This means that when Algorithm 3 merges the balls in a connected component into a single cluster, the resulting cluster satisfies the constraint, as the constraint is mergeable. It follows that the output clustering (C, ϕ) also satisfies the mergeable constraint. ◀

▶ **Lemma 10.** *Algorithm 3 runs in $O(kn^2)$ time.*

Proof. The main bottlenecks of Algorithm 3 are the construction of the edges E for graph G, and finding a center c in each connected component B' that minimizes the radius when merging the balls in B'. Since there are $O(n)$ points in P and $O(k)$ balls, checking if each point $p \in P$ is in both of each pair of B_i, B_j to construct E takes $O(nk^2)$ time. To find c, we can simply probe all points in the connected component. Since there are $O(k)$ components, and each component can contain up to $O(n)$ points, we can find c in $O(kn^2)$ time, resulting in an overall runtime of $O(kn^2)$. ◀

Next, we give an upper bound on the cost of the clustering (C, ϕ) computed by Algorithm 3. Consider any connected component of G with a set of balls B'. Let P' be the set of points in the union of the balls of B'. Denote the sum of the radii of the balls in B' by $\mathcal{R}$. Then we have the following lemma.

▶ **Lemma 11.** *There is a point $c \in P'$ such that for any $q \in P'$, $d(p, q) \leq \frac{4}{3}\mathcal{R}$.*

Before proving the lemma, we describe a major consequence of it. Recall that for each connected component of G, Algorithm 3 assigns all points in the associated balls to a single cluster center c. Moreover, this center c is chosen to be a point $p \in P'$ that minimizes the maximum distance from all other points. Thus, by the above lemma, the resulting cluster $\mathcal{C} = \phi^{-1}(c)$ has $\mathrm{cost}(\mathcal{C}) \leq \frac{4}{3}\mathcal{R}$, where $\mathcal{R}$ is the sum of the radii of the balls in the component. As B is the disjoint union of balls in the components, we have the following corollary.

▶ **Corollary 12.** *The cost of the output clustering (C, ϕ) of Algorithm 3 is at most $\frac{4}{3}$ times the sum of the radii of the input balls in B.*

Now, we move on towards proving Lemma 11. We need a few additional definitions and observations. Let T be a spanning tree of the concerned connected component. We call a path $\pi = \{\pi_1, \ldots, \pi_m\}$ in T a *maximum leaf-to-leaf path* if π is a simple path from a leaf node in T to another leaf node, and the sum of the radii of the balls along π is the maximum among all such paths. Now, consider any path $\pi = \{\pi_1, \ldots, \pi_m\}$ between two leaves. Also, consider the graph $T' = T - \pi$, constructed by removing the vertices and edges of π from T. Note that T' is a collection of connected components each having exactly one ball B'_i connected to exactly one $\pi_i \in \pi$ with an edge of T. We say that such a component is *connected to T* via π_i. A set of balls $Z \subseteq B$ is said to form a *connected set* if the subgraph of G induced by the balls in Z is connected.

▶ **Observation 13.** *Consider a connected set Z of balls with the sum of radii $\mathcal{R}'$. For any two points a, b contained in the balls of Z, $d(a, b) \leq 2 \cdot \mathcal{R}'$.*

Proof. Let $B(c_i, r_i), B(c_j, r_j) \in Z$ be the balls containing a, b, respectively. Since Z is a connected set, there is a simple path between $B(c_i, r_i)$ and $B(c_j, r_j)$ in G that uses only balls of Z. Thus, by triangle inequality, $d(c_i, c_j) \leq r_i + 2(\mathcal{R}' - r_i - r_j) + r_j = 2 \cdot \mathcal{R}' - r_i - r_j$. Hence, $d(a, b) \leq d(a, c_i) + d(c_i, c_j) + d(c_j, b) \leq 2 \cdot \mathcal{R}'$. ◀

Similarly, we also have the following observation.

▶ **Observation 14.** *Consider a connected set Z of balls with the sum of radii $\mathcal{R}'$. For any point p in a ball of Z and a ball $B(c_i, r_i) \in Z$, $d(c_i, p) \leq 2 \cdot \mathcal{R}' - r_i$.*

Proof. Let $B(c_j, r_j) \in Z$ be a ball containing p. Since Z is a connected set, there is a simple path between $B(c_i, r_i)$ and $B(c_j, r_j)$ in G that uses only balls of Z. Thus, by triangle inequality, $d(c_i, p) \leq d(c_i, c_j) + d(c_j, p) \leq r_i + 2(\mathcal{R}' - r_i - r_j) + r_j + d(c_j, p) \leq 2 \cdot \mathcal{R}' - r_i$. ◀

The proof of Lemma 11 is as follows.

Proof. We prove the existence of the point c with the desired property by considering Procedure 4 that explicitly computes a center. We clarify that this procedure is only for the purpose of analysis, and we do not run it as a part of our algorithm.

Procedure 4 picks the new center according to two main cases: (i) there exists a ball in B' with radius at least $\frac{2}{3}\mathcal{R}$, (ii) all balls in B' have radii less than $\frac{2}{3}\mathcal{R}$.

In case (i), there exists a ball $B(c'_i, r'_i)$ such that $r'_i \geq \frac{2}{3}\mathcal{R}$. In this case, Procedure 4 picks c'_i as the new center. By Observation 14, we have that for any point $p \in P'$, $d(c'_i, p) \leq 2 \cdot \mathcal{R} - r'_i \leq 2 \cdot \mathcal{R} - \frac{2}{3}\mathcal{R} = \frac{4}{3}\mathcal{R}$.

In case (ii), each ball in B' has a radius of less than $\frac{2}{3}\mathcal{R}$. The center selection is split up into two subcases: (iia) $\mathrm{Sum}_L \geq \frac{1}{3}\mathcal{R}$, and (iib) $\mathrm{Sum}_L < \frac{1}{3}\mathcal{R}$.

We first consider case (iia), where a point p contained in both π_{i-1} and π_i is chosen as the new center. As each ball has radius less than $\frac{2}{3}\mathcal{R}$ and π_1 is a leaf, while loop 7 cannot terminate with $i = 1$, i.e., at its termination $i \geq 2$. Thus, π_{i-1} is well-defined. Consider the partition of B' into two parts B_L and B_R. B_L is the union of the balls $\pi_1, \ldots, \pi_{i-1}$ and any ball that lies in a component of $T - \pi$ that is connected to T via π_j for $1 \leq j \leq i - 1$.

Procedure 4 Find-Center(B').

Input : The set of balls $B' = \{B(c'_1, r'_1), \dots, B(c'_l, r'_l)\}$ having a sum of radii of $\mathcal{R}$.
Output : A point $c \in P'$.

`// Case (i)`
1 **if** *there exists* $B(c'_i, r'_i) \in B'$ *with* $r'_i \geq \frac{2}{3}\mathcal{R}$ **then**
2   **return** c'_i
`// Case (ii)`
3 **else**
4     $T \leftarrow$ a spanning tree of the component with the set B' of balls
5     $\pi \leftarrow (\pi_1, \dots, \pi_m)$, a maximum leaf-to-leaf path in T
6     $i \leftarrow 0$
7     $\mathrm{Sum}, \mathrm{Sum}_L \leftarrow 0$
8   **while** $Sum < \frac{2}{3}\mathcal{R}$ **do**
9         $i \leftarrow i + 1$
10         $\mathrm{Sum}_L \leftarrow \mathrm{Sum}$
11         $\mathrm{Sum} \leftarrow \mathrm{Sum} +$ the radius of π_i
12   **for** *each component* Γ *in* $T' = T - \pi$*, connected to* T *via* π_i **do**
13   **for** *each ball* $B(c'_j, r'_j)$ *in* Γ **do**
14                 $\mathrm{Sum} \leftarrow \mathrm{Sum} + r'_j$
    `// Case (iia)`
15   **if** $Sum_L \geq \frac{1}{3}\mathcal{R}$ **then**
16         $p \leftarrow$ a point contained in the intersection of the balls π_{i-1} and π_i **return** p
    `// Case (iib)`
17   **else**
18   **return** the center of the ball π_i

Similarly, B_R is the union of the balls $\pi_i, \dots, \pi_m$ and any ball that lies in a component of $T - \pi$ that is connected to T via π_j for $i \leq j \leq m$. Let $P_L = \{p_L \mid \text{point } p_L \text{ is in a ball of } B_L\}$ and $P_R = \{p_R \mid \text{point } p_R \text{ is in a ball of } B_R\}$. Note that $P_L \cup P_R = P'$. Next, we bound the distance between the chosen center p and the points in P_L and P_R. Note that by our definition of p, P_L, and P_R, $p \in P_L \cap P_R$. First, we consider $d(p, p_L)$ for any $p_L \in P_L$. Note that $\mathrm{Sum}_L < \frac{2}{3}\mathcal{R}$ at the end of while loop 7. Also, by our definition of Sum_L, it is the sum of the radii of the balls in B_L. Then, as B_L is a connected set of balls and $p \in P_L$, by Observation 13, it follows that $d(p, p_L) \leq 2 \cdot \mathrm{Sum}_L < \frac{4}{3}\mathcal{R}$. Now we consider $d(p, p_R)$ for any $p_R \in P_R$. Since $\mathrm{Sum}_L \geq \frac{1}{3}\mathcal{R}$ in this subcase and (B_L, B_R) is a partition of B', we know that the sum of the radii of the balls in B_R, $\mathrm{Sum}_R \leq \frac{2}{3}\mathcal{R}$. Then, as B_R is a connected set of balls and $p \in P_R$, by Observation 13, it follows that $d(p, p_R) \leq 2 \cdot \mathrm{Sum}_R \leq \frac{4}{3}\mathcal{R}$. Thus, the lemma follows in this subcase.

Next, we consider case (iib), here, we pick center c'_i of ball $\pi_i = B(c'_i, r'_i)$ as the new center. Similar to case (iia), we partition B' into three parts B_L, B_M, and B_R. B_L is the union of the balls $\pi_1, \dots, \pi_{i-1}$ and any ball that lies in a component of $T - \pi$ that is connected to T via π_j for $1 \leq j \leq i-1$. Similarly, B_M is the union of the ball π_i and any ball that lies in a component of $T - \pi$ that is connected to T via π_i. Lastly, B_R is the union of the balls $\pi_{i+1}, \dots, \pi_m$ and any ball that lies in a component of $T - \pi$ that is connected to T via π_j for $i + 1 \leq j \leq m$. Let $P_L = \{p_L \mid \text{point } p_L \text{ is in a ball of } B_L\}$, $P_M = \{p_M \mid \text{point } p_M \text{ is in a ball of } B_M\}$, and $P_R = \{p_R \mid \text{point } p_R \text{ is in a ball of } B_R\}$. Note that $P_L \cup P_M \cup P_R = P'$, so we can again bound the overall distance by considering P_L, P_M, and P_R separately. Towards this end, note that by the end of Loop 5, $\mathrm{Sum}_L = \mathrm{cost}(B_L)$ and $\mathrm{Sum} = \mathrm{cost}(B_L) + \mathrm{cost}(B_M)$.

We first consider $d(p, p_L)$ for any $p_L \in P_L$. Note that $B_L \cup \{\pi_i\}$ is a connected set of balls with a sum of radii of $\mathrm{Sum}_L + r'_i$. Furthermore, recall that $\mathrm{Sum}_L < \frac{1}{3}\mathcal{R}$ by the assumption of case (iib), and $r'_i < \frac{2}{3}\mathcal{R}$ by the assumption of case (ii). By Observation 14, we have that $d(c'_i, p_L) \leq 2 \cdot (\mathrm{Sum}_L + r'_i) - r'_i = 2 \cdot \mathrm{Sum}_L + r'_i < 2 \cdot \frac{1}{3}\mathcal{R} + \frac{2}{3}\mathcal{R} = \frac{4}{3}\mathcal{R}$.

Next, we consider $d(p, p_R)$ for some $p_R \in P_R$. Note that $\mathrm{cost}(B_R) = \mathcal{R} - \mathrm{Sum}$. Since $\mathrm{Sum} \geq \frac{2}{3}\mathcal{R}$ by the end of Loop 7, we have that $\mathrm{cost}(B_R) = \mathcal{R} - \mathrm{Sum} \leq \frac{1}{3}\mathcal{R}$. Note also that $B_R \cup \{\pi_i\}$ is a connected set of balls with a sum of radii of $\mathrm{cost}(B_R) + r'_i$. Since $r'_i < \frac{2}{3}\mathcal{R}$ by the assumption of case (ii), by Observation 14 we have that for any $p_R \in P_R$, $d(c'_i, p_R) \leq 2 \cdot (\mathrm{cost}(B_R) + r'_i) - r'_i = 2 \cdot \mathrm{cost}(B_R) + r'_i < 2 \cdot \frac{1}{3}\mathcal{R} + \frac{2}{3}\mathcal{R} = \frac{4}{3}\mathcal{R}$.

Lastly, we consider $d(p, p_M)$ for some $p_M \in P_M$. Let $B_l \in B_M$ be a ball that contains p_M. Consider any simple path π' in T between any leaf ball in B_M and π_i such that π' also contains B_l. Such a path exists, as B_l lies in a component of $T - \pi$ that is connected to T via π_i. We denote by $\pi' - \pi_i$ the path obtained by removing π_i from π'. Note that

$$\mathrm{Sum}_L + \mathrm{cost}(\pi' - \pi_i) + r'_i + \mathrm{Sum}_R \leq \mathcal{R}. \tag{1}$$

Since balls along the path π have the maximum sum of radii, we have that

$$\mathrm{cost}(\pi' - \pi_i) + r'_i = \mathrm{cost}(\pi') \leq \mathrm{cost}(\pi) \leq \mathrm{Sum}_L + r'_i + \mathrm{Sum}_R$$

and therefore

$$\mathrm{cost}(\pi' - \pi_i) \leq \mathrm{Sum}_L + \mathrm{Sum}_R. \tag{2}$$

Combining Inequalities 1 and 2 we have that

$$2 \cdot \mathrm{cost}(\pi' - \pi_i) + r'_i \leq \mathcal{R}. \tag{3}$$

Now, note that π' is a connected set of balls including $B_l \ni p_M$ with the sum of radii $\mathrm{cost}(\pi' - \pi_i) + r'_i$. Thus, by Observation 14 we have that $d(c'_i, p_M) \leq 2 \cdot (\mathrm{cost}(\pi' - \pi_i) + r'_i) - r'_i = 2 \cdot \mathrm{cost}(\pi' - \pi_i) + r'_i$. Applying Inequality 3 we have that $d(c'_i, p_M) \leq \mathcal{R}$.

Since $d(p_L, c'_i), d(p_M, c'_i), d(p_R, c'_i) \leq \frac{4}{3}\mathcal{R}$ for all $p_L \in P_L, p_M \in P_M$, and $p_R \in P_R$, and $P_L \cup P_M \cup P_R$ covers all points in P', c'_i is at a distance of at most $\frac{4}{3}\mathcal{R}$ from any point in P' in case (iib). This finishes the proof of the lemma. ◀

3.4 Guessing Radii

Here we describe a procedure for guessing the radii of any k-MSR clustering up to $(1+\epsilon)$ factor. Formally, one can enumerate a set R of k-sized vectors such that for any k-MSR clustering with radii $r_1 \leq \ldots \leq r_k$, there is a vector $(r'_1, \ldots, r'_k)$ in R where for all $1 \leq j \leq k$, $r_j \leq r'_j$ and if $r_j > \frac{\epsilon \cdot r_k}{k}$, $r'_j \leq (1+\epsilon) r_j$. Moreover, the size of R is only $k^{O(\log_{1+\epsilon} \frac{k}{\epsilon})} n^2$. From now on, we call this type of enumeration process a *guessing* procedure, and we refer to each enumerated entry as a *guess*. Similar procedures have been used previously for guessing the radii of an optimal capacitated sum of radii clustering [24, 5, 19, 25, 13]. We restate those procedures to obtain a general algorithm that can enumerate the radii of all clusterings. In our algorithm, this procedure will be used to guess both a set of optimal vanilla radii and a set of optimal radii under any given mergeable constraint. Towards this end, we clarify that in the guessed radii vector $(r'_1, \ldots, r'_k)$, all radii might not be distinct, and specifically we have $O(\log_{1+\epsilon} \frac{k}{\epsilon})$ many distinct radii. So, we will enumerate $O(\log_{1+\epsilon} \frac{k}{\epsilon})$-sized vectors instead while keeping track of the multiplicities.

The procedure works in the following manner. Given a set of n points P in a metric space, and parameters k and ϵ, we first compute a radius r_{max} that can potentially be the maximum radius of some clustering; we can do this in $O(n^2)$ time by trying distances between each pair of points in P. For each r_{max}, we set $[\frac{\epsilon \cdot r_{max}}{k}, (1+\epsilon)r_{max}]$ as the range in which we guess the radii. The procedure is described as Algorithm 5.

Algorithm 5 Guess-Radii(P, k, ϵ).

Input : A set of points P in a metric space, and parameters k and ϵ.
Output : Sets $R = \{R_i \mid 1 \leq i \leq x\}$ and $K = \{K_{ij} \mid 1 \leq i \leq x, 1 \leq j \leq y\}$ for some x and y, where each R_i, K_{ij} are l-sized vectors for $l = O(\log_{1+\epsilon} \frac{k}{\epsilon})$.

$R \leftarrow \emptyset, K \leftarrow \emptyset, i \leftarrow 1$
for *each* $p_1, p_2 \in P$ **do**
  $r_{max} \leftarrow d(p_1, p_2)$
  $M \leftarrow \arg\min_\tau (1+\epsilon)^\tau \geq r_{max}$
  $m \leftarrow \arg\min_\tau (1+\epsilon)^\tau \geq \frac{\epsilon \cdot r_{max}}{k}$
  $R_i \leftarrow ((1+\epsilon)^m, \ldots, (1+\epsilon)^M)$
  $R \leftarrow R \cup \{R_i\}$
  $j \leftarrow 1$
  for *each* $(k_m, k_{m+1}, \ldots, k_M)$, $0 \leq k_t \leq k$ *for* $m \leq t \leq M$ **do**
    $K_{ij} \leftarrow (k_m, \ldots, k_M)$
    $K \leftarrow K \cup \{K_{ij}\}$
    $j \leftarrow j + 1$
  $i \leftarrow i + 1$
return R, K

Now we show some properties of the output size, runtime, and correctness of the procedure. First, we note the following observation about the number of distinct radii in each $R_i \in R$.

▶ **Observation 15.** *Each $R_i \in R$ from the output of Algorithm 5 has $O(\log_{1+\epsilon} \frac{k}{\epsilon})$ distinct radii.*

Proof. Consider the radii vector $R_i = ((1+\epsilon)^m, \ldots, (1+\epsilon)^M)$. Since R_i is formed by picking powers of $1+\epsilon$ from the range $[\frac{\epsilon \cdot r_{max}}{k}, (1+\epsilon)r_{max}]$, the number of radii in R_i is bounded by $M - m + 1 \leq 1 + \log_{1+\epsilon} \frac{(1+\epsilon) \cdot r_{max}}{\frac{\epsilon \cdot r_{max}}{k}} = O(\log_{1+\epsilon} \frac{k}{\epsilon})$. ◀

Furthermore, we have the following observation about the size of K, which will help us bound the runtime of both Algorithm 5 and some other algorithms.

▶ **Observation 16.** *The size of K is bounded by $k^{O(\log_{1+\epsilon} \frac{k}{\epsilon})} n^2$.*

Proof. First, note that there are a total of n^2 choices for r_{max}. Next, we consider the number of distinct radii vectors $K_{ij} = (k_m, \ldots, k_M)$ that are computed for each r_{max}. By Observation 15 we have that $M - m + 1 = |R_i|$ is bounded by $O(\log_{1+\epsilon} \frac{k}{\epsilon})$. Since each $k_t \leq k$, there are $k^{O(\log_{1+\epsilon} \frac{k}{\epsilon})}$ possible vectors K_{ij} for each choice of r_{max}. It follows that there are a total of $k^{O(\log_{1+\epsilon} \frac{k}{\epsilon})} n^2$ many vectors in K. ◀

Since Algorithm 5 enumerates each entry of K, the next lemma follows.

▶ **Lemma 17.** *Algorithm 5 runs in $k^{O(\log_{1+\epsilon} \frac{k}{\epsilon})} n^2$ time.*

Next, we show that for any clustering, Algorithm 5 outputs a solution with a correct guess.

▶ **Lemma 18.** *For any k-MSR clustering with radii $r_1, \ldots, r_k$, there are vectors $R_i = (r'_1, \ldots, r'_l)$ in R and $K_{ij} = (k'_1, \ldots, k'_l)$ in K for some i, j such that there exist exactly k'_t indices $\kappa \in \{1, \ldots, k\}$ with $r_\kappa \le r'_t < (1+\epsilon) r_\kappa$ for $2 \le t \le l$ and k'_1 indices $\kappa \in \{1, \ldots, k\}$ with $r_\kappa \le r'_1$. Moreover, $\sum_{t=1}^{l} k'_t \cdot r'_t \le (1+\epsilon) \sum_{i=1}^{k} r_i$.*

Proof. Consider the choice of r_{max} made in Algorithm 5 so that $r_{max} = \max_{i=1}^{k} r_i$. Also consider the vector $R_i = (r'_1, \ldots, r'_l)$ in R corresponding to this choice. By definition, $r'_l \le (1+\epsilon) r_{max}$ and $r'_1 \ge \frac{\epsilon \cdot r_{max}}{k}$. Define k'_1 to be the number of radii in $\{r_1, \ldots, r_k\}$ that are at most r'_1. For $2 \le t \le l$, define k'_t to be the number of radii in $\{r_1, \ldots, r_k\}$ that are also in $(r'_{t-1}, r'_t]$. By definition, $\sum_{t=1}^{l} k'_t = k$, and so $(k'_1, \ldots, k'_l)$ is a vector in K which is considered w.r.t r_{max}. Also by definition of k'_t for $2 \le t \le l$, there are k'_t indices $\kappa \in \{1, \ldots, k\}$ with $r_\kappa \in (r'_{t-1}, r'_t]$, i.e., $r_\kappa \le r'_t < (1+\epsilon) r_\kappa$ as desired.

Next, we prove the moreover part. The sum of the k'_1 radii r'_1 is at most $k'_1 (1+\epsilon) \frac{\epsilon \cdot r_{max}}{k} \le 2\epsilon \cdot r_{max} \le 2\epsilon \sum_{i=1}^{k} r_i$. Also, for $2 \le t \le l$, $k'_t \cdot r'_t \le \sum_{\kappa \in \{1, \ldots, k\} | r_\kappa \in (r'_{t-1}, r'_t]} (1+\epsilon) r_\kappa$. Summing over all $2 \le t \le l$, $\sum_{t=2}^{l} k'_t \cdot r'_t \le (1+\epsilon) \sum_{i=1}^{k} r_i$. Hence, $\sum_{t=1}^{l} k'_t \cdot r'_t \le (1+3\epsilon) \sum_{i=1}^{k} r_i$. Scaling ϵ down by a factor of 3, we obtain the lemma. ◀

▶ **Corollary 19.** *For any optimal sum of radii clustering (vanilla or constrained) $\mathcal{C}$, there are vectors $R_i = (r'_1, \ldots, r'_l)$ in R and $K_{ij} = (k'_1, \ldots, k'_l)$ in K for some i, j such that there is a feasible $(1+\epsilon)$-approximate clustering that uses k'_t balls of radius r'_t for all $1 \le t \le l$.*

Proof. Let $r_1, \ldots, r_k$ be the radii of an optimal clustering. Fix the vectors $R_i = (r'_1, \ldots, r'_l)$ and $K_{ij} = (k'_1, \ldots, k'_l)$ given by Lemma 18. We show that each cluster of $\mathcal{C}$ can be covered by a ball of radius r'_τ for some τ such that k'_t balls of radius r'_t are used for all $1 \le t \le l$. As the clustering induced by these balls is essentially the same clustering $\mathcal{C}$, it remains feasible. To cover each cluster of $\mathcal{C}$ that uses a ball of radius r_κ, if $r_\kappa \le r'_1$, use the same center, but with radius r'_1; otherwise, as $\sum_{t=1}^{l} k'_t = k$, there is $2 \le \tau \le l$ with $r_\kappa \le r'_\tau < (1+\epsilon) r_\kappa$, and in this case use the same center, but with radius r'_τ. By the correspondence of the radii $\{r_\kappa\}$ and $\{r'_t\}$ in Lemma 18, k'_t balls of radius r'_t are used for all $1 \le t \le l$. Lastly, as $\sum_{t=1}^{l} k'_t \cdot r'_t \le (1+\epsilon) \sum_{i=1}^{k} r_i$, the new set of balls induces a $(1+\epsilon)$-approximate clustering. ◀

3.5 The Algorithm and its Analysis

With all of the procedures defined, we state the main algorithm. Let $k > 0$ and $\epsilon > 0$ be some constants, our algorithm is given as Algorithm 6.

We now analyze Algorithm 6, starting with the correctness and approximation factor. We have the following Lemma.

▶ **Lemma 20.** *Algorithm 6 produces a $(4+\epsilon)$-approximate k-MSR clustering satisfying a given mergeable constraint.*

Proof. Let OPT_v and OPT_c be the respective optimal cost of vanilla k-MSR and k-MSR with the given mergeable constraint. By Corollary 19, there are vectors $R_i = (r'_1, \ldots, r'_l)$ in R and $K_{ii'} = (k'_1, \ldots, k'_l)$ in K for some i, i' such that there is a feasible $(1+\epsilon)$-approximate vanilla clustering that uses k'_t balls of radius r'_t for all $1 \le t \le l$. Fix this choice of R_i and $K_{ii'}$ in Line 2. Then by Lemma 4, Vanilla-Clustering$(P, R_i, K_{ii'})$ (Algorithm 1) outputs a feasible $(2+2\epsilon)$ vanilla clustering (C_v, ϕ_v). Let $c_1, \ldots, c_k$ be the centers in C_v. Similarly, by Corollary 19, there are vectors $R_j = (r_1, \ldots, r_l)$ in R and $K_{jj'} = (k_1, \ldots, k_l)$ in K for some j, j' such

Algorithm 6 Mergeable-Constraint-k-MSR(P, k, ϵ).

Input : A set of points P in a metric space and parameters k and ϵ.
Output : A clustering (C, ϕ) of P satisfying the mergeable constraint.

// Find sets of radii and multiplicity vectors
1 $R, K \leftarrow$ Guess-Radii$(P, k, \epsilon/3)$
2 **for** *each radii vector* $R_i \in R$ *with multiplicity vector* $K_{ii'} \in K$ **do**
    // Find a vanilla clustering $(C_v^{ii'}, \phi_v^{ii'})$
3     $(C_v^{ii'}, \phi_v^{ii'}) \leftarrow$ Vanilla-Clustering$(P, R_i, K_{ii'})$
4 $(C_v, \phi_v) \leftarrow$ a feasible clustering $(C_v^{ii'}, \phi_v^{ii'})$ with the smallest sum of radii over all i, i'
5 Let $c_1^v, \dots, c_k^v$ be a fixed ordering of the centers in C_v
6 $R_e \leftarrow \emptyset$
7 **for** *each radii vector* $R_j \in R$ *with multiplicity vector* $K_{jj'} \in K$ **do**
    // A set $R_e^{jj'}$ of expanded radii vectors
8     $R_e^{jj'} \leftarrow$ Expand-Balls$(C_v, \phi_v, R_j, K_{jj'})$ $R_e \leftarrow R_e \cup R_e^{jj'}$
9 **for** *each vector* $R_j' = (r_1^e, \dots, r_k^e) \in R_e$ **do**
    // Find a constrained clustering (C_j, ϕ_j)
10     $B \leftarrow \{B(c_1^v, r_1^e), \dots, B(c_k^v, r_k^e)\}$
11     $(C_j, \phi_j) \leftarrow$ Merge-Balls(B)
12 $(C, \phi) \leftarrow (C_j, \phi_j)$ satisfying the mergeable constraint with the smallest sum of radii over all j
13 **return** (C, ϕ)

that there is a feasible $(1+\epsilon)$-approximate clustering $\mathcal{C}^*$ with the given mergeable constraint that uses k_t balls of radius r_t for all $1 \le t \le l$. Fix this choice of R_j and $K_{jj'}$ in Line 7. Then by Lemma 6, the output $R_e^{jj'}$ of Expand-Balls$(C_v, \phi_v, R_j, K_{jj'})$ (Algorithm 2) and hence R_e contains a vector $(r_1^c, \dots, r_k^c)$ such that each cluster of $\mathcal{C}^*$ is contained in $B(c_j, r_j^c)$ for some $1 \le j \le k$. Moreover, by the same lemma, $\sum_{\mu=1}^k r_\mu^c \le \sum_{\tau=1}^k r'_\tau + \sum_{t=1}^l k_t \cdot r_t \le (2+2\epsilon)OPT_v + (1+\epsilon)OPT_c \le (3+3\epsilon)OPT_c$. This is true, as the optimal vanilla clustering cost is at most the optimal cost of any constrained clustering.

Fix the above choice of $(r_1^c, \dots, r_k^c) \in R_e$ in Line 10. Then $B = \{B(c_1, r_1^c), \dots, B(c_l, r_k^c)\}$, and hence by Lemma 9, the clustering (C_j, ϕ_j) returned by Merge-Balls(B) (Algorithm 3) satisfies the mergeable constraint. Moreover, by Corollary 12, the cost of (C_j, ϕ_j) is at most $\frac{4}{3}\sum_{\mu=1}^k r_\mu^c \le \frac{4}{3}(3+3\epsilon)OPT_c = (4+4\epsilon)OPT_c$. Scaling ϵ by a constant, we obtain the lemma. ◀

Now we analyze the runtime of Algorithm 6. We have the following lemma.

▶ **Lemma 21.** *Algorithm 6 runs in* $2^{O(\frac{k}{\epsilon}\cdot\log^2\frac{k}{\epsilon})}n^4$ *time.*

Proof. We first show the runtime of each part of the algorithm. By Lemma 17 we have that the Guess-Radii procedure has a runtime of $k^{O(\log_{1+\epsilon}\frac{k}{\epsilon})}n^2$. Next, by Observation 16 we also have that Guess-Radii produces $k^{O(\log_{1+\epsilon}\frac{k}{\epsilon})}n^2$ many vectors in K. By Lemma 5 we have that the Vanilla-Clustering procedure runs in $O((\log_{1+\epsilon}\frac{k}{\epsilon})^k \cdot n)$ time; since we run Vanilla-Clustering on each pair $R_i \in R$ and $K_{ii'} \in K$, this step of the algorithm has a runtime of

$$\begin{aligned} & k^{O(\log_{1+\epsilon}\frac{k}{\epsilon})}n^2 \cdot (\log_{1+\epsilon}\frac{k}{\epsilon})^k \cdot n \\ = & k^{O(\log_{1+\epsilon}\frac{k}{\epsilon})} \cdot (\log_{1+\epsilon}\frac{k}{\epsilon})^k n^3. \end{aligned}$$

Moving on to the expand radii step, by Lemma 8 we have that the Expand-Balls procedure runs in $(O(\log_{1+\epsilon}\frac{k}{\epsilon}))^k$ time. Also, Expand-Balls runs for each of the $k^{O(\log_{1+\epsilon}\frac{k}{\epsilon})}n^2$ pairs $R_j \in R$ and $K_{jj'} \in K$, resulting in a runtime of $k^{O(\log_{1+\epsilon}\frac{k}{\epsilon})} \cdot (\log_{1+\epsilon}\frac{k}{\epsilon})^k n^2$. The next step of the algorithm, Merge-Balls, runs in $O(kn^2)$ time by Lemma 10. We also showed in Lemma 7 that Expand-Balls outputs $(O(\log_{1+\epsilon}\frac{k}{\epsilon}))^k$ pairs $R_j \in R$ and $K_{jj'} \in K$, so the total number of guesses for expanded radii is bounded by $k^{O(\log_{1+\epsilon}\frac{k}{\epsilon})} \cdot (\log_{1+\epsilon}\frac{k}{\epsilon})^k n^2$. Since Algorithm 6 runs Merge-Balls for each set of guessed radii $R'_j \in R_e$, this step has a runtime of

$$\begin{aligned}
&k \cdot n^2 \cdot k^{O(\log_{1+\epsilon}\frac{k}{\epsilon})} \cdot (\log_{1+\epsilon}\frac{k}{\epsilon})^k n^2 \\
&= k^{O(\log_{1+\epsilon}\frac{k}{\epsilon})} \cdot (\log_{1+\epsilon}\frac{k}{\epsilon})^k n^4.
\end{aligned}$$

Adding everything together we have that Algorithm 6 runs in time

$$\begin{aligned}
&k^{O(\log_{1+\epsilon}\frac{k}{\epsilon})}n^2 + k^{O(\log_{1+\epsilon}\frac{k}{\epsilon})} \cdot (\log_{1+\epsilon}\frac{k}{\epsilon})^k n^3 + \\
&k^{O(\log_{1+\epsilon}\frac{k}{\epsilon})} \cdot (\log_{1+\epsilon}\frac{k}{\epsilon})^k n^2 + k^{O(\log_{1+\epsilon}\frac{k}{\epsilon})} \cdot (\log_{1+\epsilon}\frac{k}{\epsilon})^k n^4 \\
&= k^{O(\log_{1+\epsilon}\frac{k}{\epsilon})} \cdot (\log_{1+\epsilon}\frac{k}{\epsilon})^k n^4 \\
&= 2^{O(\frac{k}{\epsilon}\cdot\log^2\frac{k}{\epsilon})} n^4.
\end{aligned}$$ ◀

Lemma 20 and Lemma 21 together produces Theorem 1.

▶ **Theorem 1.** *For any $\epsilon > 0$, there is a $(4+\epsilon)$-approximation algorithm for the k-MSR problem under mergeable constraints that runs in $2^{O(\frac{k}{\epsilon}\cdot\log^2\frac{k}{\epsilon})}n^4$ time.*

4 Hardness of Clustering with Mergeable Constraints

▶ **Definition 22** (Fair Representational Clustering)**.** *We are given ℓ disjoint groups of points, denoted as $P_1, \ldots, P_\ell$, with a total of n points with a distance metric d. Each group P_i is associated with fairness parameters $\alpha_i, \beta_i \in [0,1]$, for $1 \le i \le \ell$. A clustering is said to be* fair representational *if, within every cluster, the proportion of points from group i lies between α_i and β_i, for all i. The objective is to find a fair representational clustering with k clusters, that minimizes the sum of the radii of the clusters.*

As shown by Carta et al. [11], the fair representational constraint is mergeable. We prove that Fair Representational Clustering cannot be solved in time $f(k)n^{o(k)}$ unless ETH is false. Specifically, we show a reduction from the Dominating set problem.

Given an instance of Dominating set containing an n-vertex graph $G = (V, E)$ and a parameter k, we construct a new graph G'. Initially, $G' = G$. Add two sets V_1, V_2 of k vertices each to G' and the edges $\{v, v_i\}, \{v, v_j\}$ for all $v \in V, v_i \in V_1, v_j \in V_2$. The instance $\mathcal{I}$ of Fair Representational Clustering is defined as follows: $\ell = 3$, $P_1 = V$, $P_2 = V_1$, $P_3 = V_2$, d is the shortest path metric in G', $\alpha_1 = 1/3, \beta_1 = 1$, and for $i \in \{2,3\}$, $\alpha_i = 1/(n+2)$ and $\beta_i = 1$.

▶ **Lemma 23.** *Dominating set admits a solution of size k if and only if there is a fair representational clustering of $\cup_{i=1}^3 P_i$ with k clusters of cost k.*

Proof. The forward direction is easy to prove. Suppose Dominating set admits a solution $V' = \{v_1, v_2, \ldots, v_k\}$. We construct a clustering (C, ϕ) of $\cup_{i=1}^3 P_i$ as follows. Use the points $\{v_1, v_2, \ldots, v_k\}$ in P_1 as centers. For each $v \in P_1$, assign v to a center v_j that dominates v. Assign the j-th points of P_2 and P_3 to v_j. It is easy to verify that each cluster has radius

1, so the total cost is k. Also in each cluster, the proportion of points of P_1 is at least $1/3$, the proportion of points of P_2 is at least $1/(n+2)$, and the same for P_3. So, (C, ϕ) is a fair representational clustering.

Now, suppose there is a fair representational clustering of $\cup_{i=1}^{3} P_i$ with k clusters of cost k. First, we argue that each cluster has a non-zero radius, as otherwise, it is a singleton cluster, which is not fair. This implies that the radius of each cluster is 1. Now, each cluster center must be in $P_1 = V$, as the distance between any two points $u \in P_2, u' \in P_3$ is 2, excluding them to be a center of a fair cluster of radius 1. It follows that any point $v \in P_1$ is at a distance at most 1 from the k cluster centers. Hence, these vertices in V form a dominating set of size k. ◀

Since it is not possible to solve Dominating set in $f(k)n^{o(k)}$ time unless ETH is false [17], we obtain Theorem 2.

▶ **Theorem 2.** *Fair Representational Clustering cannot be solved in time $f(k)n^{o(k)}$ unless ETH is false.*

References

1 Sara Ahmadian and Chaitanya Swamy. Approximation algorithms for clustering problems with lower bounds and outliers. In Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi, editors, *43rd International Colloquium on Automata, Languages, and Programming, ICALP*, volume 55 of *LIPIcs*, pages 69:1–69:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPICS.ICALP.2016.69.

2 Sara Ahmadian and Chaitanya Swamy. Approximation Algorithms for Clustering Problems with Lower Bounds and Outliers. In Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi, editors, *43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016)*, volume 55 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 69:1–69:15, Dagstuhl, Germany, 2016. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ICALP.2016.69.

3 Vijay Arya, Naveen Garg, Rohit Khandekar, Adam Meyerson, Kamesh Munagala, and Vinayaka Pandit. Local search heuristics for k-median and facility location problems. *SIAM J. Comput.*, 33(3):544–562, 2004. doi:10.1137/S0097539702416402.

4 Sayan Bandyapadhyay, Eden Chlamtáč, Yury Makarychev, and Ali Vakilian. A polynomial-time approximation for pairwise fair k-median clustering. *arXiv preprint arXiv:2405.10378*, 2024. doi:10.48550/arXiv.2405.10378.

5 Sayan Bandyapadhyay, William Lochet, and Saket Saurabh. FPT constant-approximations for capacitated clustering to minimize the sum of cluster radii. In Erin W. Chambers and Joachim Gudmundsson, editors, *39th International Symposium on Computational Geometry, SoCG 2023, June 12-15, 2023, Dallas, Texas, USA*, volume 258 of *LIPIcs*, pages 12:1–12:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPICS.SOCG.2023.12.

6 Sandip Banerjee, Yair Bartal, Lee-Ad Gottlieb, and Alon Hovav. Improved fixed-parameter bounds for min-sum-radii and diameters k-clustering and their fair variants. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(15):15481–15488, April 2025. doi:10.1609/aaai.v39i15.33699.

7 Suman Bera, Deeparnab Chakrabarty, Nicolas Flores, and Maryam Negahbani. Fair algorithms for clustering. In *Advances in Neural Information Processing Systems*, pages 4954–4965, 2019.

8 Ioana O Bercea, Martin Groß, Samir Khuller, Aounon Kumar, Clemens Rösner, Daniel R Schmidt, and Melanie Schmidt. On the cost of essentially fair clusterings. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2019)*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.APPROX-RANDOM.2019.18.

9 Moritz Buchem, Katja Ettmayr, Hugo KK Rosado, and Andreas Wiese. A (3)-approximation algorithm for the minimum sum of radii problem with outliers and extensions for generalized lower bounds. In *Proceedings of the 2024 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1738–1765. SIAM, 2024.

10 Lena Carta, Lukas Drexler, Annika Hennes, Clemens Rösner, and Melanie Schmidt. FPT Approximations for Fair k-Min-Sum-Radii. In Julián Mestre and Anthony Wirth, editors, *35th International Symposium on Algorithms and Computation (ISAAC 2024)*, volume 322 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 16:1–16:18, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ISAAC.2024.16.

11 Lena Carta, Lukas Drexler, Annika Hennes, Clemens Rösner, and Melanie Schmidt. Fpt approximations for fair k-min-sum-radii, 2024. doi:10.48550/arXiv.2410.00598.

12 Moses Charikar and Rina Panigrahy. Clustering to minimize the sum of cluster diameters. *J. Comput. Syst. Sci.*, 68(2):417–441, 2004. doi:10.1016/j.jcss.2003.07.014.

13 Xianrun Chen, Dachuan Xu, Yicheng Xu, and Yong Zhang. Parameterized approximation algorithms for sum of radii clustering and variants. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 20666–20673, 2024. doi:10.1609/AAAI.V38I18.30053.

14 Anshuman Chhabra, Karina Masalkovaitė, and Prasant Mohapatra. An overview of fairness in clustering. *IEEE Access*, 2021. doi:10.1109/ACCESS.2021.3114099.

15 Flavio Chierichetti, Ravi Kumar, Silvio Lattanzi, and Sergei Vassilvitskii. Fair clustering through fairlets. In *Advances in Neural Information Processing Systems*, pages 5029–5037, 2017. URL: https://proceedings.neurips.cc/paper/2017/hash/978fce5bcc4eccc88ad48ce3914124a2-Abstract.html.

16 Vincent Cohen-Addad, Anupam Gupta, Lunjia Hu, Hoon Oh, and David Saulpic. An improved local search algorithm for k-median, 2021. arXiv:2111.04589.

17 Marek Cygan, Fedor V Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized algorithms*, volume 5. Springer, 2015. doi:10.1007/978-3-319-21275-3.

18 Lukas Drexler, Annika Hennes, Abhiruk Lahiri, Melanie Schmidt, and Julian Wargalla. Approximating fair k-min-sum-radii in euclidean space. In *International Workshop on Approximation and Online Algorithms*, pages 119–133. Springer, 2023. doi:10.1007/978-3-031-49815-2_9.

19 Arnold Filtser and Ameet Gadekar. Fpt approximations for capacitated sum of radii and diameters. *arXiv preprint arXiv:2409.04984*, 2024. doi:10.48550/arXiv.2409.04984.

20 Zachary Friggstad and Mahya Jamshidian. Improved polynomial-time approximations for clustering with minimum sum of radii or diameters. In Shiri Chechik, Gonzalo Navarro, Eva Rotenberg, and Grzegorz Herman, editors, *30th Annual European Symposium on Algorithms, ESA 2022, September 5-9, 2022, Berlin/Potsdam, Germany*, volume 244 of *LIPIcs*, pages 56:1–56:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.ESA.2022.56.

21 Matt Gibson, Gaurav Kanade, Erik Krohn, Imran A. Pirwani, and Kasturi R. Varadarajan. On clustering to minimize the sum of radii. *SIAM J. Comput.*, 41(1):47–60, 2012. doi:10.1137/100798144.

22 Teofilo F Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, 38:293–306, 1985. doi:10.1016/0304-3975(85)90224-5.

23 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. doi:10.1287/MOOR.10.2.180.

24 Tanmay Inamdar and Kasturi Varadarajan. Capacitated Sum-Of-Radii Clustering: An FPT Approximation. In Fabrizio Grandoni, Grzegorz Herman, and Peter Sanders, editors, *28th Annual European Symposium on Algorithms (ESA 2020)*, volume 173 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 62:1–62:17, Dagstuhl, Germany, 2020. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ESA.2020.62.

25 Ragesh Jaiswal, Amit Kumar, and Jatin Yadav. FPT approximation for capacitated sum of radii. In Venkatesan Guruswami, editor, *15th Innovations in Theoretical Computer Science Conference, ITCS 2024, January 30 to February 2, 2024, Berkeley, CA, USA*, volume 287 of *LIPIcs*, pages 65:1–65:21. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPICS.ITCS.2024.65.