Parameterized Approximation Algorithms for k-Center Clustering and Variants

Sayan Bandyapadhyay[1], **Zachary Friggstad**[*2], **Ramin Mousavi**[2]

[1] Department of Informatics, University of Bergen, Norway
[2] Department of Computing Science, University of Alberta, Canada
Sayan.Bandyapadhyay@uib.no, zacharyf@ualberta.ca, mousavih@ualberta.ca

Abstract

k-center is one of the most popular clustering models. While it admits a simple 2-approximation in polynomial time in general metrics, the Euclidean version is NP-hard to approximate within a factor of 1.93, even in the plane, if one insists the dependence on k in the running time be polynomial. Without this restriction, a classic algorithm yields a $2^{O((k \log k)/\epsilon)}dn$-time $(1+\epsilon)$-approximation for Euclidean k-center, where d is the dimension.

We give a faster algorithm for small dimensions: roughly speaking an $O^*(2^{O((1/\epsilon)^{O(d)} \cdot k^{1-1/d} \cdot \log k)})$-time $(1+\epsilon)$-approximation. In particular, the running time is roughly $O^*(2^{O((1/\epsilon)^{O(1)} \sqrt{k} \log k)})$ in the plane. We complement our algorithmic result with a matching hardness lower bound.

We also consider a well-studied generalization of k-center, called Non-uniform k-center (NUkC), where we allow different radii clusters. NUkC is NP-hard to approximate within any factor, even in the Euclidean case. We design a $2^{O(k \log k)}n^2$ time 3-approximation for NUkC in general metrics, and a $2^{O((k \log k)/\epsilon)}dn$ time $(1+\epsilon)$-approximation for Euclidean NUkC. The latter time bound matches the bound for k-center.

1 Introduction

Given n data points in a metric space, a *clustering* of the data is a partition into a number of groups (or clusters), such that points in each group are more similar compared to points across multiple groups. The notion of similarity is captured by the distances between the points, which form a metric. The task of partitioning the data as above is referred to also as clustering, which has numerous applications in the areas of artificial intelligence (AI), machine learning (ML) and data mining. For example, during the training process of an AI/ML model, the training samples are clustered into similar groups to enhance the expressive power of learning methods. With the goal of retrieving the natural clustering of the points, the task of partitioning is modeled as an optimization problem where the goal is to find a clustering that optimizes some objective function. For center-based objectives, each cluster is represented by a point, which is called the center of the cluster. The similarity (or dissimilarity) of a cluster is measured in terms of the deviation of the points in the cluster from the center. Consequently, this deviation is captured by the corresponding objective function, which one needs to minimize. Arguably the most popular center-based objectives are k-center, k-means, and k-median. All of these problems ask to find k center points, where $k > 0$ is a given integer. The desired k clusters are formed by assigning each point to its closest center. For k-center, the deviation is defined as the maximum distance between a point and its closest center, and for k-median, it is the sum of the distances between points and their closest centers. k-means is similar to k-median except here the deviation is the sum of the square of the distances.

In this article, we limit our discussions to the Euclidean version of the above clustering problems where the data points and centers belong to a real space of dimension $d \geq 2$ and the distance measure is the Euclidean distance. All of these problems are NP-hard even in the plane, i.e., when $d = 2$ (Mahajan, Nimbhorkar, and Varadarajan 2012; Megiddo and Supowit 1984; Feder and Greene 1988). For both k-median and k-means, $(1+\epsilon)$-approximation algorithms (or *approximation schemes*) are known with running time $2^{(f(\epsilon))^{d-1}} n \log^{d+6} n$ (Kolliopoulos and Rao 2007) and $nk(\log n)^{(d/\epsilon)^{O(d)}}$ (Cohen-Addad 2018), respectively, for some function f. Note that for constant dimension and in particular, for $d = 2$, these algorithms run in polynomial time. In contrast, it is widely known that such an approximation scheme does not exist for k-center (Mentzer 1988; Feder and Greene 1988; Chen 2021). Indeed, it is NP-hard to approximate k-center in the plane up to a factor of 1.82 (Mentzer 1988; Chen 2021). On the positive side, several polynomial-time 2-approximations are known for k-center (Gonzalez 1985; Feder and Greene 1988). On the hardness side of k-median and k-means, it is known that polynomial-time approximation scheme is not possible for these problems (or are APX-hard) when the dimension is not necessarily a constant (Awasthi et al. 2015; Bhattacharya, Goyal, and Jaiswal 2020).

From the above discussion, it is evident that all these problems are intractable when the dimension is arbitrary. To cope with this hardness, researchers have designed *fixed-parameter tractable* (FPT) $(1+\epsilon)$-approximations. Fixed-

[*]Supported by an NSERC Discovery Grant and Accelerator Supplement.

Problem	Algorithms/Upper bounds	Hardness/Lower bounds
k-center	$2^{O((k\log k)/\epsilon)}dn$ $2^{O((1/\epsilon)^{O(d)}k^{1-1/d}\log k)}$ Thm. 2.5	1.82-factor in $n^{O(1)}$ $O^*(2^{2^{\bar{\epsilon}\cdot\delta\cdot d}\cdot k^{1-\delta}})$ Thm. 1.3
k-median	$2^{(f(\epsilon))^{d-1}}n\log^{d+6}n$ $2^{(k/\epsilon)^{O(1)}}nd$	APX-hard
k-means	$nk(\log n)^{(d/\epsilon)^{O(d)}}$ $2^{(k/\epsilon)^{O(1)}}nd$	APX-hard
NUkC	3-approx. in $2^{O(k\log k)}n^2$ (metric) Thm. 4.2 $2^{O((k\log k)/\epsilon)}dn$ Thm. 4.6	γ-hard $\forall\gamma$ in $n^{O(1)}$

Table 1: A summary of previous and our work. Our results are marked with theorem numbers.

parameter tractability is a notion in *parameterized complexity* (Cygan et al. 2015a) which allows running time to be expressed as a function of a parameter p. In particular, an algorithm has FPT running time parameterized by p if it runs in $f(p)\cdot n^{O(1)}$ time, where f is a function that solely depends on p. For clustering problems, the most natural parameter is k, which is typically small in practice, and hence an FPT $(1+\epsilon)$-approximation algorithm leads to an efficient approximation scheme. Note, in particular, that now the running time does not depend exponentially on d. Consequently, Badoiu et al. (2002) designed an approximation scheme for k-center that runs in time $2^{O((k\log k)/\epsilon^2)}dn$, and later Badoiu and Clarkson (2003) slightly improved the time to $2^{O((k\log k)/\epsilon)}dn$. For k-median and k-means, several FPT approximation schemes were designed in a series of work (Badoiu, Har-Peled, and Indyk 2002; De La Vega et al. 2003) culminating in a $2^{(k/\epsilon)^{O(1)}}nd$ time approximation scheme due to Kumar et al. (2010).

The study of sub-exponential algorithms is a popular research direction in parameterized complexity. Here the goal is to design a $2^{o(p)}n^{O(1)}$ time algorithm for restricted instances of a problem that admits an $f(p)\cdot n^{O(1)}$ time algorithm (Demaine et al. 2005; Dorn et al. 2013). A central theme of this subarea is to design $2^{O(\sqrt{p}\cdot\text{polylog}(p))}n^{O(1)}$ time algorithms for planar instances of the problems (Fomin et al. 2020, 2016; Nederlof 2020), e.g., planar vertex cover (Cygan et al. 2015a). (polylog(p) is a constant power of $\log p$.) This theme is informally known as the "square root phenomenon".

Motivated by the above research directions, we study sub-exponential algorithms for k-center when the dimension d is small. In particular, we ask the following questions.

> Does the planar version of k-center ($d=2$) admit a $2^{O(\sqrt{k}\cdot\text{polylog}(k))}n^{O(1)}$ time approximation scheme? Does k-center admit a $2^{o(k)}n^{O(1)}$ time approximation scheme when d is a constant?

Note that for constant d, k-median and k-means already admit even better polynomial-time approximation schemes. Thus, the above questions are relevant only to k-center. Considering the above questions, we answer both of them in affirmative.

Theorem 1.1 (Informal)**.** *For any $0<\epsilon\le 1$, there is an $O^*(2^{O((1/\epsilon)^{O(1)}\sqrt{k}\log k)})$-time $(1+\epsilon)$-approximation algorithm for 2-dimensional k-center. In general, there is an $O^*(2^{O((1/\epsilon)^{O(d)}\cdot k^{1-1/d}\cdot\log k)})$-time $(1+\epsilon)$-approximation algorithm for k-center in $\mathbb{R}^d$.*

In this informal statement, the O^*-notation suppresses polynomial factors and a leading factor of d^d in. Note that for any constant d, this yields a sub-exponential $O^*(2^{o(k)})$ time approximation scheme. One should compare these results with the earlier polynomial-time 1.82-factor hardness of approximation for planar k-center, which motivated us to design FPT algorithms. Theorem 1.1 is an informal version of Theorem 2.5 which appears in Section 2.

In Section 3, we prove a new hardness bound for k-center based on Exponential Time Hypothesis (ETH) (Impagliazzo, Paturi, and Zane 2001) that almost complements our running time bound in the 2-dimensional case.

Theorem 1.2. *There exists a constant $\alpha>1$ such that there is no α-approximation algorithm for 2-dimensional k-center in time $2^{o(k^{1/4})}\cdot n^{O(1)}$, unless ETH fails.*

Note that in the above hardness bound, the power of k is only $1/4$, whereas in our upper bound it is $1/2$. However, in the discrete case of k-center (popularly known as k-supplier), where centers can only be chosen from a given finite set of points, we obtain a tight lower bound based on Randomized ETH (rETH) (Dell et al. 2014). We note that the result of Theorem 1.1 also holds for k-supplier (Theorem 2.1) which is described in Section 2. Our hardness result addresses the double-exponential dependence on d in the running time by showing one cannot improve it substantially if one hopes to have better-than-linear dependence on k in the exponent.

Theorem 1.3. *Under rETH, there are constants $\bar{\epsilon}>0$ and $\alpha>1$ such that there is no α-approximation algorithm for d-dimensional Euclidean k-supplier in time $O^*(2^{2^{\bar{\epsilon}\cdot\delta\cdot d}\cdot k^{1-\delta}})$ for any $0<\delta<1$.*

Next, we turn our attention to another clustering problem called Non-uniform k-center. It is a generalization of k-center where it is possible to select clusters of different radii.

Definition 1.4 (Non-Uniform k-center (NUkC))**.** *Given two sets of points $\mathcal{C}$ and $\mathcal{F}$ in a metric space (X,dist), an integer $k>0$, $t\le k$ distinct integers (radii) $r_1>r_2>\ldots>r_t>0$ and non-negative integers $k_1,\ldots,k_t$ such that $\sum_{i=1}^{t}k_i=$*

k, the goal is to find a number (dilation) α and to choose k_i balls centered at the points of $\mathcal{F}$ with radius $\alpha \cdot r_i$ for all $1 \leq i \leq t$, such that the union of the chosen balls contains all the points of $\mathcal{C}$ and α is minimized.

Note that in the special case when $t = 1$, the problem is basically k-center. NUkC was formulated by Chakrabarty et al. (2020) who also described its applications in fine-tuned clustering and vehicle routing. They showed that for any $\gamma \geq 1$, the problem in general metrics is NP-hard to approximate within a factor of γ. The same hardness holds even in the discrete (and hence in the continuous) Euclidean case (Bandyapadhyay 2020). Consequently, we ask the following questions.

> Does NUkC in general metrics admit a constant-factor approximation in FPT ($f(k) \cdot n^{O(1)}$) time? Does Euclidean NUkC admit an approximation scheme in FPT ($f(k) \cdot n^{O(1)}$) time?

In this work, we answer both of the questions in affirmative.

Theorem 1.5 (Informal)**.** *A 3-approximation for NUkC can be computed in time $2^{O(k \log k)} n^2$. Moreover, if $\mathcal{C} = \mathcal{F}$, the approximation factor is 2. For Euclidean NUkC, a $(1+\epsilon)$-approximation can be computed in time $2^{O((k \log k)/\epsilon)} dn$.*

Note that the result for Euclidean NUkC is a strict generalization of the result for Euclidean k-center due to Badoiu and Clarkson (2003). Indeed, our algorithm is motivated by their algorithm. Theorem 1.5 is an informal version of Theorem 4.2 and 4.6 which appear in Section 4. See Table 1 for a summary of our results.

2 k-center

We begin by presenting our faster approximation scheme for k-center. In fact, we give a faster approximation scheme for the closely-related k-supplier problem, and then discuss how this can be used to give a faster approximation for k-center.

k-supplier. We are given points $\mathcal{C}$ to be clustered and given candidate centers $\mathcal{F}$. The goal is to find $F \subseteq \mathcal{F}$ with $|F| = k$ minimizing $\text{cost}(F) := \max_{p \in \mathcal{C}} d(p, F)$.

Throughout this section, $O()$ will suppress only absolute constants that are independent of d and ϵ.

We first prove the following.

Theorem 2.1. *For any $d \geq 1$ and $1 \geq \epsilon > 0$, there is a $(1+\epsilon)$-approximation for instances of k-supplier in $\mathbb{R}^d$ with running time $O(|\mathcal{C}| \cdot |\mathcal{F}| + 2^{O((1/\epsilon)^{O(d)} \cdot k^{1-1/d} \cdot \log k + d \log d)})$.*

Note, if one simply regards d and ϵ as constants then the exponential term in the running time simplifies to $2^{O(k^{1-1/d} \cdot \log k)}$. For the special case of $\mathbb{R}^2$, the exponential part of the running time is $2^{O(\sqrt{k} \cdot \log k)}$. The doubly-exponential dependence on d is clearly not desirable, but we provide evidence in the next section suggesting this may not be possible to improve.

The main idea behind our approach uses a somewhat-recent result by Bhattiprolu and Har-Peled (2016) on Voronoi separators. Intuitively, they show that for any n points in $\mathbb{R}^d$, one can insert an additional $O(n^{1-1/d})$ points such that in the Voronoi diagram of all points (original and inserted), the two sets of original points can be partitioned into two roughly-equal halves and these halves are separated by the Voronoi cells of the new points.

We utilize such a Voronoi separator to help guess $O(k^{1-1/d} \cdot (1/\epsilon)^{O(d)})$ centers of the optimum solution such that if we serve all points near these centers, the remaining problem naturally decomposes into two roughly equal-size halves that can be treated independently (thus, bounding the depth of recursion to be logarithmic). Naturally, to ensure we are guessing these centers from a set of size $O(k \cdot (1/\epsilon)^{O(d)})$ rather than from a possibly-larger set $O(|\mathcal{F}|)$ we also have to filter the input so points are not "close" to each other.

Preliminary Step: Reducing to a feasibility check In $O(|\mathcal{C}| \cdot |\mathcal{F}|)$ time, we can find a solution $F \subseteq \mathcal{F}$ with $\text{cost}(F) \leq 3 \cdot OPT$ (Hochbaum and Shmoys 1986). We then know $OPT \leq \text{cost}(F) \leq 3 \cdot OPT$. In the remaining steps, we will describe a binary search algorithm that for some value $R > 0$, will either find a k-supplier solution with value $(1+\epsilon) \cdot R$ or will (correctly) declare there is no solution with value $\leq R$. We then use a binary search to find a value R in the range $[\text{cost}(F)/3, \text{cost}(F)]$ such that using R produces an infeasible solution whereas using $(1+\epsilon) \cdot R$ will return a solution with cost $(1+\epsilon) \cdot (1+\epsilon) \cdot R \leq (1+3\epsilon) \cdot R$. Scaling ϵ by a constant factor gives the desired result. The number of iterations of the binary search will be at most $\log_2 \frac{3}{\epsilon}$.

By scaling the point set, it suffices to give such an algorithm for $R = 1$. That is, from this point forward we present an algorithm that will either find a solution with cost at most $1+\epsilon$ or correctly declare there is no solution with value ≤ 1.

Step 1: Reducing the input size We perform standard filtering. Initially let $\mathcal{C}' = \emptyset$ and $\mathcal{F}' = \emptyset$. Then we process $p \in \mathcal{C}$ one at a time: if $d(p, \mathcal{C}') > \epsilon$ then we add p to $\mathcal{C}'$. Also process $i \in \mathcal{F}$ one at a time: if $d(i, \mathcal{F}') > \epsilon$ yet $d(i, \mathcal{C}') \leq 1 + \epsilon$, then add i to $\mathcal{F}'$.

Lemma 2.2. *If there is a solution with cost at most 1, we have $|\mathcal{C}'|, |\mathcal{F}'| \leq k \cdot (1/\epsilon)^{O(d)}$ and the k-supplier instance $(\mathcal{C}', \mathcal{F}', k)$ has optimum value at most $1 + 2 \cdot \epsilon$. Conversely, given a solution $F' \subseteq \mathcal{F}'$ with cost at most α in the new instance, its cost in the original instance is at most $\alpha + \epsilon$.*

Proof. Suppose there is a solution with cost at most 1 in the original instance. Let $F^* \subseteq F$ be the optimum centers. Notice $d(p, p') > \epsilon$ for distinct $p, p' \in \mathcal{C}'$. So for each $i \in F^*$, the number of points $p \in \mathcal{C}'$ with $d(i, p) \leq 1$ is at most $(1/\epsilon)^{O(d)}$. Since $\text{cost}(F^*) \leq 1$, all $p \in \mathcal{C}'$ lie within distance 1 from some $i \in F^*$. So the total number of points in $\mathcal{C}'$ is at most $|F^*| \cdot (1/\epsilon)^{O(d)} = k \cdot (1/\epsilon)^{O(d)}$.

Similarly, note $d(i, i') > \epsilon$ for distinct $i, i' \in \mathcal{F}'$ but each $i \in \mathcal{F}'$ lies within distance at most $1 + \epsilon$ from some $p \in \mathcal{C}'$. Therefore, $|\mathcal{F}'| \leq |\mathcal{C}'| \cdot (1/\epsilon)^{O(d)} \leq k \cdot (1/\epsilon)^{O(d)}$.

Next, let F'^* denote the solution obtained from F^* as follows. Consider each $i \in F^*$ that is actually covering a point in $\mathcal{C}'$ (i.e. $d(i, \mathcal{C}') \leq 1$)). If $i \in \mathcal{F}'$, add i to F'^*. Otherwise

let $i' \in \mathcal{F}'$ satisfy $d(i,i') \leq \epsilon$. Add i' to $F^{'*}$. Since each $p \in \mathcal{C}'$ had distance at most 1 from F^*, it then has distance at most $1+\epsilon$ from $F^{'*}$.

Conversely, let $F' \subseteq \mathcal{F}'$ be a solution with cost α in the new instance. Consider any $p \in \mathcal{C}$. Since $d(p,p') \leq \epsilon$ for some $p' \in \mathcal{C}'$ (it could be $p = p'$) then $d(p,F') \leq d(p,p') + d(p',F') \leq \epsilon + \alpha$. □

In our algorithm, if $|\mathcal{C}'|$ or $|\mathcal{F}'|$ exceeds $k \cdot (1/\epsilon)^{O(d)}$ then we declare this instance is a **no** instance and terminate. In fact, if we terminate in the middle of this step once $|\mathcal{C}'|$ or $|\mathcal{F}'|$ becomes too large then a simple implementation runs in time $O((|\mathcal{C}| + |\mathcal{F}|) \cdot k \cdot (1/\epsilon)^{O(d)})$.

Also, if $k \leq d \cdot \log 1/\epsilon$ we simply solve the problem using brute force within the desired time by trying all subsets of $\mathcal{F}'$ of size at most k (there are $(1/\epsilon)^{O(d^2 \log 1/\epsilon)}$ such subsets to try).

Step 2: Identifying a sparse separator We use the following result about separators in Euclidean spaces that mimics the planar separator theorem.

Theorem 2.3 ((Bhattiprolu and Har-Peled 2016)). *Let $\mathcal{X} \subseteq \mathbb{R}^d$ be a set of n points and let $c_d := \lceil 2\sqrt{d} \rceil^d + 1$. In expected $O(n)$ time, one can compute a set of $\mathcal{Z} \subseteq \mathbb{R}^d$ with $|\mathcal{Z}| \leq O(n^{1-1/d})$ and a partition $\mathcal{X}_1, \mathcal{X}_2$ of $\mathcal{X}$ such that: a) In the Voronoi diagram of $\mathcal{X} \cup \mathcal{Z}$, the Voronoi cells of points $p \in \mathcal{X}_1$ and $q \in \mathcal{X}_2$ do not share any common point and b) $|\mathcal{X}_1|, |\mathcal{X}_2| \leq n \cdot (1 - 1/c_d)$.*

Note this is a randomized algorithm, but it is guaranteed to produce such a structure. It is only the running time that is a random variable.

Let $\mathcal{Z}$ be the Voronoi separator computed for point set $\mathcal{X} := \mathcal{C}' \cup \mathcal{F}'$ and let $\mathcal{X}_1, \mathcal{X}_2$ be the two parts of the partition of $\mathcal{X}$.

Step 3: Guessing and recursing Here, we guess a small set of centers in the optimum solution that lie near the Voronoi separator and recurse on both sides.

For each point q in the Voronoi separator $\mathcal{Z}$, we guess all points in $F^{'*}$ (the optimum solution for $(\mathcal{C}', \mathcal{F}', k)$) that lie within distance at most $2 \cdot (1+\epsilon)$ of q. Recall $d(i,i') > \epsilon$ for distinct $i, i' \in \mathcal{F}'$, so there are at most $(1/\epsilon)^{O(d)}$ points to guess for this point q.

Let $\mathcal{G}$ be the union of these guesses for all $q \in \mathcal{Z}$, so $|\mathcal{G}| \leq \zeta := O(k^{1-1/d} \cdot (1/\epsilon)^{O(d)})$. Using standard esimates on binomial coefficients, the number of such guesses to enumerate is at most $\binom{|\mathcal{F}'|}{\zeta} \leq 2^{O((1/\epsilon)^d \cdot k^{1-1/d} \cdot \log k)}$.

For each such guess $\mathcal{G}$, we remove all points in $\mathcal{C}'$ that are within distance at most $1+\epsilon$ from $\mathcal{G}$. Say the remaining points are C''. Also let $\mathcal{F}'' := \mathcal{F}' - \mathcal{G}$.

Lemma 2.4. *Suppose $\mathcal{G} \subseteq \mathcal{F}'$ is the proper guess. For $j = 1, 2$, every point in $\mathcal{C}'' \cap \mathcal{X}_j$ is within distance at most $1+\epsilon$ from $F^{'*} \cap (\mathcal{X}_j - \mathcal{G})$.*

Proof. Let $p \in \mathcal{C}'' \cap \mathcal{X}_j$ and suppose $i \in F^{'*}$ satisfies $d(p,i) \leq 1+\epsilon$. Since $p \in \mathcal{C}''$, it must be that $i \notin \mathcal{G}$. If $i \notin \mathcal{X}_j$, then the straight line connecting i to p must touch the Voronoi cell for some $q \in \mathcal{Z}$.

Let t be the first point along the $p-i$ segment that touches the Voronoi cell for q. As this is a Voronoi diagram, we have $d(t,q) \leq d(t,i)$. So,

$$d(q,i) \leq d(q,t) + d(t,i) \leq 2 \cdot d(t,i) \leq 2 \cdot (1+\epsilon)$$

since $d(t,i) \leq d(p,i) \leq 1+\epsilon$. But then i would have been included in $\mathcal{G}$, a contradiction. So $i \in \mathcal{X}_j - \mathcal{G}$. □

Finally, we guess $k_j := |\mathcal{F}'' \cap F^{'*}|$ for both $j = 1, 2$ and independently recurse starting at step 2 on each of the two instances $(\mathcal{C}'' \cap \mathcal{X}_j, \mathcal{F}'' \cap \mathcal{X}_j, k_j), j = 1, 2$. For the appropriate guess $\mathcal{G}$, each of the two subproblems has a feasible solution of cost at most $1+\epsilon$.

The base case is when we recurse with an empty subproblem (i.e. $\mathcal{C}'' \cap \mathcal{X}_j = \emptyset$) in which case there are no centers to select. If a subproblem we ever recurse on has $\mathcal{C}' \neq \emptyset$ yet $\mathcal{F}' = \emptyset$, we declare this subproblem to be infeasible and return no solution.

If some guess for $\mathcal{G}$ and k_1, k_2 has both recursive calls returning a feasible solution (i.e. of cost $1+\epsilon$), then adding the centers to $\mathcal{G}$ produces a feasible solution for this instance $(\mathcal{C}', \mathcal{F}', k)$ and we return it. Otherwise, if all guesses have at least one of the two recursive calls returning no solution, we return no solution.

Analysis We argued throughout the presentation of the algorithm that if there is a feasible solution $F^{'*}$ (of size $\leq k$ and cost $\leq 1+\epsilon$), it would correctly find a solution of cost $1+\epsilon$ for the original (filtered) instance $(\mathcal{C}', \mathcal{F}', k)$ for the branches of recursion that properly guessed $\mathcal{G}$ and k_1, k_2. One should also note $d(i,i') > \epsilon$ for distinct $i, i' \in \mathcal{F}'$ holds throughout all recursive calls because it holds after step 1 and we only recurse with subsets of $\mathcal{F}'$.

Let $\Delta := k \cdot (1/\epsilon)^{O(d)}$, a bound on the initial size of $|\mathcal{F} \cup \mathcal{C}|$. At depth i of the recursion, the size of $|\mathcal{F}' \cup \mathcal{C}'|$ is bounded by $(1 - 1/c_d)^i \cdot \Delta$. The number of recursive calls spawned from a depth i recursive call is at most $2 \cdot |\mathcal{F}'|^{(1/\epsilon)^{O(d)} \cdot |\mathcal{Z}|} \leq \Delta^{(1/\epsilon)^{O(d)} \cdot (1-1/c_d)^{i/2} \cdot \Delta^{1-1/d}}$ (using $1 - 1/d \geq 1/2$). By summing a geometric series in the exponent and recalling $c_d = d^{O(d)}$ we see the total number of recursive calls reaching depth i is at most $2^i \cdot \Delta^{O((1/\epsilon)^{O(d)} \cdot d^{O(d)} \cdot \Delta^{1-1/d})}$. Noting the depth of recursion is at most $\log_{1/(1-1/c_d)} \Delta = O(c_d \cdot \log \Delta)$ and time taken between subsequent recursive calls is polynomial in $|\mathcal{F}|$, we have that the total running time of steps 1 through 3 is $2^{O((1/\epsilon)^{O(d)} \cdot k^{1-1/d} \cdot \log k + d \log d)}$

2.1 From k-Supplier to k-Center

We now turn to classic k-center in Euclidean spaces. Here, we are given points $\mathcal{C}$ and a value k. The goal is to find k points $F \subseteq \mathbb{R}^d$ minimizing $\max_{p \in \mathcal{C}}$.

Theorem 2.5. *For any $d \geq 1$ and $\epsilon > 0$, there is a $(1+\epsilon)$-approximation for instances of k-center in $\mathbb{R}^d$ with running time $O(|\mathcal{C}| \cdot k \cdot (1/\epsilon)^{O(d)} + 2^{O((1/\epsilon)^{O(d)} \cdot k^{1-1/d} \cdot \log k + d \log d)})$.*

Proof. Just like in the k-supplier algorithm, it suffices to give an algorithm with running time

$2^{O((1/\epsilon)^{O(d)} \cdot k^{1-1/d} \cdot \log k + d \log d)}$ that either finds a solution with cost $1+\epsilon$ or determines there is no solution with cost at most 1.

Begin by forming $\mathcal{C}'$ as in step 1 of the k-supplier algorithm: so $d(p,p') > \epsilon$ for distinct $p, p' \in \mathcal{C}$ but $d(p, \mathcal{C}') \leq \epsilon$ for each $p \in \mathcal{C}$. Just like in step 1 of the k-supplier algorithm, if there is a solution with cost ≤ 1, then $|\mathcal{C}'| \leq k \cdot (1/\epsilon)^{O(d)}$. If $|\mathcal{C}'|$ is larger than this, declare there is no solution.

Finally, for each $p \in \mathcal{C}'$ we let $\mathcal{F}'_p$ be any ϵ-net of the ball of radius $1+\epsilon$ centered at $\mathcal{C}'$. Then $|\mathcal{F}'_p| \leq (1/\epsilon)^{O(d)}$ yet $d(q, \mathcal{F}'_p) \leq \epsilon$ for any q with $d(p,q) \leq 1+\epsilon$. Just like in step 1 of the k-supplier algorithm, the optimum solution to the k-center instance $\mathcal{C}'$ has a feasible solution with cost at most $1+\epsilon$ by restricting the possible locations to $\mathcal{F}' = \cup_{p \in \mathcal{C}'} \mathcal{F}'_p$. Run the k-supplier algorithm with these $\mathcal{C}'$ and $\mathcal{F}'$. □

3 Hardness

In this section, we prove our hardness results. First, we describe the result for the Euclidean k-supplier problem.

Recall that O^*-notation suppresses factors polynomial in the input size. We use randomized Exponential Time Hypothesis (rETH) as our complexity theoretic assumption. Given a 3-SAT instance, we denote by n and m the number of variables and clauses in the instance, respectively. Then, rETH states that there is a constant $c > 0$ such that there is no randomized algorithm that decides 3-SAT in time $O^*(2^{c \cdot n})$ with (two-sided) error probability at most $\frac{1}{3}$ (Dell et al. 2014). Using the sparsification lemma (Jansen 2010) one can show the following, see Exercises 14.1 of (Cygan et al. 2015a).

Theorem 3.1. *Under rETH, there exists a constant $c > 0$ such that there is no randomized algorithm that decides 3-SAT in time $O^*(2^{c \cdot (n+m)})$ with (two-sided) error probability at most $\frac{1}{3}$ where n and m are the numbers of variables and clauses in the 3-SAT instance, respectively.*

There is a known reduction from 3-SAT to Vertex Cover such that the number of vertices in the Vertex Cover instance is linear in the number of variables and clauses of 3-SAT instance, see (Cygan et al. 2015a) for this reduction. Using such reduction and Theorem 3.1 we have the following hardness result on Vertex Cover.

Theorem 3.2 (Hardness of Vertex Cover)**.** *Under rETH, there is a constant $c > 0$ such that there is no randomized algorithm that decides Vertex Cover with n vertices in time $O^*(2^{c \cdot n})$ with (two-sided) error probability at most $\frac{1}{3}$.*

The last ingredient we need to show our hardness result for Euclidean k-supplier problem is the famous Johnson-Lindenstrauss dimentionality reduction (Johnson and Lindenstrauss 1984) or JL lemma for short.

Theorem 3.3 (JL lemma, reformulated)**.** *Consider n points $x_1, ..., x_n$ in $\mathbb{R}^\ell$. There exists a linear map $A : \mathbb{R}^\ell \to \mathbb{R}^{c_{JL} \cdot \log n}$ where $c_{JL} > 0$ is a constant such that for all $1 \leq i, j \leq n$ we have*

$$0.9 \cdot \|x_i - x_j\| \leq \|A(x_i) - A(x_j)\| \leq 1.1 \cdot \|x_i - x_j\|, \quad (1)$$

where the norms are l_2-norm. Furthermore, A can be computed in polynomial time by a randomized algorithm such that (1) holds for all pairs of given points with probability at least $\frac{2}{3}$.

Now we are ready to state and prove our hardness result.

Theorem 1.3. *Under rETH, there are constants $\bar{\epsilon} > 0$ and $\alpha > 1$ such that there is no α-approximation algorithm for d-dimensional Euclidean k-supplier in time $O^*(2^{2^{\bar{\epsilon} \cdot \delta \cdot d} \cdot k^{1-\delta}})$ for any $0 < \delta < 1$.*

Proof. We show a gap-introducing reduction from Vertex Cover to Euclidean k-supplier problem. Consider a Vertex Cover instance $(G = (V, E), k)$ where $V = \{1, 2, ..., n\}$. We construct an instance of Euclidean k-supplier as follows: define the client set as $\mathcal{C} := \{e_i + e_j : \forall (i,j) \in E\}$ and the set of facilities as $\mathcal{F} := \{e_i : \forall 1 \leq i \leq n\}$ where e_i is the standard unit vector in $\mathbb{R}^n$, i.e., e_i has 1 in i-th coordinate and 0 elsewhere.

Suppose G has a vertex cover $S = \{i_1, ..., i_k\}$. We claim the union of balls of radius 1 around each center e_{i_j} for all $1 \leq j \leq k$ covers all the clients. The reason is that for each client $e_r + e_s$, either r or s is in S. W.l.o.g., we assume $r \in S$, and therefore we open e_r as a center with radius 1 and the claim follows by noting that $\|e_r - (e_r + e_s)\| = 1$.

Next, we prove the other direction. Suppose G does not have a vertex cover of size k. We show that an optimal solution for k-supplier instance is at least $\sqrt{3}$. Let F^* be an optimal set of centers. We have the following fact.

Claim 1. *There is a client $e_r + e_s$ that is assigned to $e_{i^*} \in F^*$ where $i \neq r, s$.*

Note $\|e_{i^*} - (e_r + e_s)\| = \sqrt{3}$ and therefore the optimal solution for the k-supplier instance is at least $\sqrt{3}$. So it remains to prove the claim.

Proof. (of claim) Suppose not. Then, every client $e_r + e_s$ is assigned to either center e_r or e_s. It is easy to see if we define S as the set of all vertices i such that $e_i \in F^*$, then S is a feasible vertex cover for G, a contradiction. □

We can use the JL lemma (Theorem 3.3) to further reduce the above k-supplier instance in $\mathbb{R}^n$ to a k-supplier instance in dimension $c_{JL} \cdot \log n$ (recall that c_{JL} is a constant in the JL lemma). Then, with probability at least $\frac{2}{3}$, a YES-instance of Vertex Cover is mapped to an instance of k-supplier in dimension $c_{JL} \cdot \log n$ with optimal solution at most 1.1 and a NO-instance of Vertex Cover is mapped to an instance of k-supplier in dimension $c_{JL} \cdot \log n$ with optimal solution at least $\sqrt{3} \cdot 0.9$.

Finally, set $\alpha := \frac{\sqrt{3} \cdot 0.9}{1.1} > 1.4$ and $\bar{\epsilon} := \frac{1}{2 \cdot c_{JL}}$ (in the statement of Theorem). We finish the proof by way of contradiction. Suppose there is a 1.5-approximation algorithm for k-supplier in $\mathbb{R}^d$ that runs in $O^*(2^{2^{\bar{\epsilon} \cdot \delta \cdot d} \cdot k^{1-\delta}})$. Therefore, we can decide Vertex Cover in time $O^*(2^{2^{\bar{\epsilon} \cdot \delta \cdot d} \cdot k^{1-\delta}})$ with error probability at most $\frac{1}{3}$ where $d = c_{JL} \cdot \log n$. Plugging their values of $d, \alpha, \bar{\epsilon}$ and noting that $k \leq n$, we have a randomized algorithm that decides Vertex Cover in time $O^*(2^{n^{1-\frac{\delta}{2}}})$ for some $0 < \delta < 1$ and this contradicts the hardness result for Vertex Cover (Theorem 3.2).

□

3.1 Hardness of k-center

Our hardness reduction is from 3-Planar Vertex Cover which is a restricted version of Vertex Cover on planar graphs of maximum degree 3. We need the following complexity result based on Exponential Time Hypothesis (ETH) (Impagliazzo, Paturi, and Zane 2001).

Proposition 3.4. *There is no $2^{o(\sqrt{n})}$ time algorithm for 3-Planar Vertex Cover, unless ETH fails, where n is the number of vertices.*

Proof. The proposition for Planar Vertex Cover essentially follows by combining two reductions due to (Lichtenstein 1982): (i) 3-SAT to Planar 3-SAT and (ii) Planar 3-SAT to Planar Vertex Cover. For a more formal exposition, see Theorem 14.9 (Cygan et al. 2015b) which states that there is no $2^{o(\sqrt{n})}$ time algorithm for Planar Vertex Cover, unless ETH fails. Our proposition follows from the fact that the second reduction ensures that the constructed graph has maximum degree 3. □

Theorem 1.2. *There exists a constant $\alpha > 1$ such that there is no α-approximation algorithm for 2-dimensional k-center in time $2^{o(k^{1/4})} \cdot n^{O(1)}$, unless ETH fails.*

Proof. For proving the theorem, we use the gap-preserving reduction of Feder and Greene (1988) from 3-Planar Vertex Cover to Planar k-center. They start with any embedding of the planar graph where each edge is replaced by an odd length path. Let L be the sum of the lengths of the edges in the embedded graph. Their reduction ensures that if there is a vertex cover of size k_1, then there is a k-center solution of radius 1, for a suitable $k \leq k_1 + L$, and if there is no vertex cover of size k_1, then there is no k-center solution of radius at most 1.82. By using the planar embedding scheme in (Shiloach 1976), which ensures that $L = O(n^2)$, it follows that in the constructed instances of k-center, $k = O(n^2)$. Thus, using an α-approximation algorithm for Planar k-center with $\alpha \leq 1.82$ that runs in $2^{o(k^{1/4})} \cdot n^{O(1)}$ time, we can solve 3-Planar Vertex Cover exactly in $2^{o(k^{1/4})} \cdot n^{O(1)} = 2^{o(\sqrt{n})} \cdot n^{O(1)}$ time. This contradicts Proposition 3.4, and hence our claim must be true. □

4 Non-Uniform k-Center

By standard scaling argument (Chakrabarty, Goyal, and Krishnaswamy 2020), we can assume that the optimal dilation is 1. Let OPT be any optimal set of balls. We denote a ball with center c and radius r by $B(c, r)$. Consider any set of points S. A ball B is said to cover S if the points of S are contained in B. A set of balls $\mathcal{B}$ is said to cover S if the points of S are contained in the union of the balls in $\mathcal{B}$.

4.1 The General Metric Case

Note that $\mathcal{C}$ is the set of points that we need to cover and $\mathcal{F}$ is the set of centers. Let $n = |\mathcal{C} \cup \mathcal{F}|$. In this case, we give a simple 3-approximation that runs in $2^{O(k)}n^2$ time.

The Algorithm. Let sol be the solution set of balls which is initialized to $\emptyset$. Also let $\mathcal{C}' \subseteq \mathcal{C}$ be the set of points left to cover, i.e., the set of points which are not covered by sol. If $\mathcal{C}'$ is empty or $|\mathsf{sol}| = k$, then terminate. Otherwise, proceed as follows. Consider a point $p \in \mathcal{C}'$. Suppose r_i be the radius of a ball in OPT that covers p and assume that we know this index i where $1 \leq i \leq t$. Consider the point $c \in \mathcal{F}$ closest to p. Add the ball $B(c, 3r_i)$ to sol. Repeat the above steps.

The assumption that for a point we know the optimal index i can be removed in a trivial manner by trying all $t \leq k$ possible choices. The algorithm runs for at most k iterations, and thus we make the assumption for at most k points. The total number of choices to consider is thus $k^{O(k)}$. Now, in each iteration we need to find c and the ball $B(c, 3r_i)$, and remove the points in $B(c, 3r_i)$ from consideration. We can precompute and store the closest point in $\mathcal{F}$ for each point in $\mathcal{C}$. This helps us implement each iteration in $O(n^2)$ time. Hence, the algorithm runs in total $2^{O(k \log k)}n^2$ time. Next, we prove the correctness.

Lemma 4.1. *Consider the case when the algorithm makes all correct choices. For every iteration j of the algorithm except the last one, where $1 \leq j \leq k$, there is a ball B_l^* in OPT, such that in the beginning of the iteration, B_l^* is not covered by sol, but in the end of the iteration, B_l^* is covered by sol once we add the ball $B(c, 3r_i)$ to sol. Moreover, the radius of B_l^* is r_i.*

Proof. Consider any iteration j which is not the last one. Thus, $\mathcal{C}'$ is non-empty and we pick a point $p \in \mathcal{C}'$. As p is not yet covered by sol, there is a ball B_l^* in OPT containing p such that B_l^* is not covered by sol. We correctly find the radius r_i of this ball by our assumption. Now, the center c^* of B_l^* must be at most at a distance r_i from p, as $p \in B_l^*$. Thus, the closest point c computed for p must be at a distance r_i from p. Now, for any point $p' \in B_l^*$, $\mathsf{dist}(c, p') \leq \mathsf{dist}(c, p) + \mathsf{dist}(p, p') \leq r_i + \mathsf{dist}(p, c^*) + \mathsf{dist}(c^*, p') \leq r_i + r_i + r_i = 3r_i$. (The first and the second inequalities are due to triangle inequality.) Thus, any point in B_l^* is contained in $B(c, 3r_i)$. As this ball is added to sol in the end of the iteration, B_l^* is covered by sol. □

The above lemma shows that in every iteration of the algorithm except the last one, a new ball in OPT is being covered by a ball of radius 3 times the radius of the optimal ball. Thus, sol must cover all the points in $\mathcal{C}$ after at most k iterations. We note that if $\mathcal{C} = \mathcal{F}$, then an algorithm similar to the above indeed gives a 2-approximation, as in that case, one can pick the ball $B(p, 2r_i)$ in every iteration. The analysis is very similar. We obtain the following theorem.

Theorem 4.2. *A 3-approximation for NUkC can be computed in time $2^{O(k \log k)}n^2$. Moreover, if $\mathcal{C} = \mathcal{F}$, the approximation factor is 2.*

4.2 The Euclidean Case

In this case, $\mathcal{C}$ is a subset of n points of $\mathbb{R}^d$ and $\mathcal{F} = \mathbb{R}^d$. A ball $B = B(c, r)$ with $c \in \mathbb{R}^d$ is called the Minimum Enclosing Ball (MEB) of S if B covers S and there is no ball with center in $\mathbb{R}^d$ and radius strictly less than r that

covers S. We denote the center and radius of the MEB of S by $c_{B(S)}$ and $r_{B(S)}$, respectively.

Our algorithm is basically an extension of the k-center algorithm due to Badoiu et al. (2002). In k-center, the goal is to select k balls of equal radius that cover all the input points. However, in our case, we need to find k_i balls of radius r_i for each $1 \leq i \leq t$ to cover the input points. Our main contribution is to handle non-uniform radii.

The Algorithm. Consider any arbitrary ordering of the k balls $B_1^* = B(c_1^*, r_1^*), \ldots, B_k^* = B(c_k^*, r_k^*)$ in OPT. For each index $1 \leq i \leq k$, we maintain a set $S_i \subseteq \mathcal{C} \cap B_i^*$ of points which is initialized to $\emptyset$. To start with, we pick any arbitrary point $q \in \mathcal{C}$ and add it to the set S_i such that $q \in B_i^*$. For the time being, assume that we know this index i for q. Next, we apply the following procedure over $k \cdot \lceil 2/\epsilon \rceil$ iterations. In the beginning of every iteration, we check whether all the points in $\mathcal{C}$ are covered by the union of the balls $\cup_{i=1}^k B(c_{B(S_i)}, (1+\epsilon)r_i^*)$. If yes, then we terminate. Otherwise, we proceed as follows. For each point $p \in \mathcal{C}$, let $\delta(p) = \min_{i=1}^k \|c_{B(S_i)} - p\|$. Pick a point $p' \in \mathcal{C}$ among the points not in $\cup_{i=1}^k B(c_{B(S_i)}, (1+\epsilon)r_i^*)$ such that $\delta(p')$ is maximized, i.e., p' is a point which is farthest away from the centers of the MEBs among the "uncovered" points. Again suppose we know the correct optimal ball B_i^* that contains p'. Add p' to S_i.

The assumption that for a point we know its optimal ball can be removed in a trivial manner by trying all k possible choices. As we make the assumption for $O(k/\epsilon)$ points, the total number of choices to consider is $k^{O(k/\epsilon)}$. In every iteration, computation of the MEBs and p' can be done in $O(ndk)$ time. Thus, the algorithm runs in total $2^{O((k \log k)/\epsilon)} dn$ time.

It is possible to save some computation in the above algorithm by removing all points from consideration which are already covered and do not appear in a set S_i. However, the asymptotic complexity remains the same.

Next, we prove the correctness of the algorithm. In particular, we show that when the algorithm terminates, all points of $\mathcal{C}$ are covered by $\cup_{i=1}^k B(c_{B(S_i)}, (1+\epsilon)r_i^*)$. We need the following lemma proved in (Goel, Indyk, and Varadarajan 2001).

Lemma 4.3. *For any set of points $T \subset \mathbb{R}^d$ and any point z at distance K from $c_{B(T)}$, there is a point $z' \in T$ at distance at least $\sqrt{r_{B(T)}^2 + K^2}$ from z.*

First, we prove the following lemma.

Lemma 4.4. *Consider any set S_i where $1 \leq i \leq k$ and suppose the algorithm added $\lceil 2/\epsilon \rceil$ points of $\mathcal{C} \cap B_i^*$ to S_i. Then for any point $p \in \mathcal{C} \cap B_i^*$, $\|c_{B(S_i)} - p\| \leq (1+\epsilon)r_i^*$.*

Proof. Consider the first $\tau = \lceil 2/\epsilon \rceil$ iterations when the algorithm adds points of $\mathcal{C} \cap B_i^*$ to the set S_i. Let $S_{i,j}$ be the set S_i at the j-th such iteration. Thus, $S_{i,0}$ is a singleton set, and $S_{i,j+1} = S_{i,j} \cup \{p'\}$ for some $p' \in \mathcal{C} \cap B_i^*$ added to $S_{i,j}$. Also let $\hat{r}_j = r_{B(S_{i,j})}$, $\hat{R} = (1+\epsilon)r_i^*$, $\lambda_j = \hat{r}_j/\hat{R}$, and $K_j = \|c_{B(S_{i,j+1})} - c_{B(S_{i,j})}\|$. We have the following claim whose proof is similar to the proof of a theorem (Theorem 2.1) due to Badoiu and Clarkson (2003).

Claim 2. $\lambda_j \geq 1 - \frac{1}{1+j/2}$.

Proof. Note that the point p' that the algorithm adds to $S_{i,j}$ has the property that it is currently not in $\cup_{l=1}^k \{B(c_{B(S_l)}, (1+\epsilon)r_l^*)\}$ and, in particular not in $B(c_{B(S_{i,j})}, \hat{R})$. Moreover, p' is farthest away from $c_{B(S_{i,j})}$ among the points in $\mathcal{C} \cap B_i^*$ not covered by $\cup_{l=1}^k \{B(c_{B(S_l)}, (1+\epsilon)r_l^*)\}$. Thus, $\|c_{B(S_{i,j})} - p'\| > \hat{R}$. Also, $\|c_{B(S_{i,j})} - p'\| \leq \|c_{B(S_{i,j})} - c_{B(S_{i,j+1})}\| + \|c_{B(S_{i,j+1})} - p'\| \leq K_j + \hat{r}_{j+1}$. Thus, $\hat{r}_{j+1} > \hat{R} - K_j$.

By Lemma 4.3, with $S_{i,j} = T$ and $c_{B(S_{i,j+1})} = z$, there is a point of $S_{i,j}$ at least $\sqrt{\hat{r}_j^2 + K_j^2}$ from $c_{B(S_{i,j+1})}$. Thus, $\hat{r}_{j+1} \geq \sqrt{\hat{r}_j^2 + K_j^2}$. It follows that,

$$\lambda_{j+1}\hat{R} = \hat{r}_{j+1} \geq \max\{\hat{R} - K_j, \sqrt{\lambda_j^2 \hat{R}^2 + K_j^2}\}$$

The lower bound on λ_{j+1} is smallest when the above two quantities are equal, i.e., when $K_j = \frac{(1-\lambda_j^2)\hat{R}}{2}$. Thus, $\lambda_{j+1} \geq (1+\lambda_j^2)/2$.

This recurrence solves to $\lambda_j \geq 1 - \frac{1}{1+j/2}$ using the fact that $\lambda_0 = 0$. □

Now, we know that $S_{i,j} \subseteq \mathcal{C} \cap B_i^*$ and so the radius of the MEB of $S_{i,j}$ can be at most r_i^*. Hence, λ_j can be at most $1/(1+\epsilon)$ by definition. By the above claim, this maximum value is achieved when $j \geq 2/\epsilon$, and thus $j < 1 + \lceil 2/\epsilon \rceil$. It follows that after $\lceil 2/\epsilon \rceil$ points of $\mathcal{C} \cap B_i^*$ are added to S_i, $(1+\epsilon)$-expansion of $B(S_i)$ contains all the points of $\mathcal{C} \cap B_i^*$. As the radius of $B(S_i)$ is at most r_i^*, for any point $p \in B_i^*$, $\|c_{B(S_i)} - p\| \leq (1+\epsilon) \cdot r_{B(S_i)} \leq (1+\epsilon)r_i^*$. □

Next, we finish the correctness proof.

Lemma 4.5. *When the algorithm terminates, all points of $\mathcal{C}$ are covered by $\cup_{i=1}^k B(c_{B(S_i)}, (1+\epsilon)r_i^*)$.*

Proof. Suppose the statement is not true, i.e., there is a point p which is not covered by $\cup_{i=1}^k B(c_{B(S_i)}, (1+\epsilon)r_i^*)$. This means the algorithm ran for all $k \cdot \lceil 2/\epsilon \rceil$ iterations, i.e., the total size of the sets $\{S_i\}$ is $k \cdot \lceil 2/\epsilon \rceil$. Let j be the index such that $p \in B_j^*$. Now, consider the case when the algorithm makes all correct choices. Note that there must be at least one such case, as we try all possible k choices in every step. By Lemma 4.4 and the way the algorithm adds points to the sets $\{S_i\}$, the size of each S_i can be at most $\lceil 2/\epsilon \rceil$. But, this implies that the size of S_j is at least $\lceil 2/\epsilon \rceil$, as the total size of the sets $\{S_i\}$ is $k \cdot \lceil 2/\epsilon \rceil$. It follows by Lemma 4.4, $\|c_{B(S_j)} - p\| \leq (1+\epsilon)r_j^*$, which is a contradiction. □

From the above discussion, we obtain the following.

Theorem 4.6. *A $(1+\epsilon)$-approximation for Euclidean NUkC can be computed in time $2^{O((k \log k)/\epsilon)} dn$.*

References

Awasthi, P.; Charikar, M.; Krishnaswamy, R.; and Sinop, A. K. 2015. The hardness of approximation of euclidean k-means. *arXiv preprint arXiv:1502.03316*.

Badoiu, M.; and Clarkson, K. L. 2003. Smaller core-sets for balls. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms, January 12-14, 2003, Baltimore, Maryland, USA*, 801–802. ACM/SIAM.

Badoiu, M.; Har-Peled, S.; and Indyk, P. 2002. Approximate clustering via core-sets. In Reif, J. H., ed., *Proceedings on 34th Annual ACM Symposium on Theory of Computing, May 19-21, 2002, Montréal, Québec, Canada*, 250–257. ACM.

Bandyapadhyay, S. 2020. On Perturbation Resilience of Non-Uniform k-Center. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2020)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik.

Bhattacharya, A.; Goyal, D.; and Jaiswal, R. 2020. Hardness of Approximation of Euclidean k-Median. *arXiv preprint arXiv:2011.04221*.

Bhattiprolu, V. V. S. P.; and Har-Peled, S. 2016. Separating a Voronoi Diagram via Local Search. In *32nd International Symposium on Computational Geometry (SoCG 2016)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik.

Chakrabarty, D.; Goyal, P.; and Krishnaswamy, R. 2020. The non-uniform k-center problem. *ACM Transactions on Algorithms (TALG)*, 16(4): 1–19.

Chen, R. 2021. On Mentzer's Hardness of the k-Center Problem on the Euclidean Plane.

Cohen-Addad, V. 2018. A Fast Approximation Scheme for Low-Dimensional k-Means. In Czumaj, A., ed., *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*, 430–440. SIAM.

Cygan, M.; Fomin, F. V.; Kowalik, Ł.; Lokshtanov, D.; Marx, D.; Pilipczuk, M.; Pilipczuk, M.; and Saurabh, S. 2015a. *Parameterized algorithms*, volume 5. Springer.

Cygan, M.; Fomin, F. V.; Kowalik, L.; Lokshtanov, D.; Marx, D.; Pilipczuk, M.; Pilipczuk, M.; and Saurabh, S. 2015b. *Parameterized Algorithms*. Springer. ISBN 978-3-319-21274-6.

De La Vega, W. F.; Karpinski, M.; Kenyon, C.; and Rabani, Y. 2003. Approximation schemes for clustering problems. In *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, 50–58.

Dell, H.; Husfeldt, T.; Marx, D.; Taslaman, N.; and Wahlén, M. 2014. Exponential time complexity of the permanent and the Tutte polynomial. *ACM Transactions on Algorithms (TALG)*, 10(4): 1–32.

Demaine, E. D.; Fomin, F. V.; Hajiaghayi, M. T.; and Thilikos, D. M. 2005. Subexponential parameterized algorithms on bounded-genus graphs and H-minor-free graphs. *J. ACM*, 52(6): 866–893.

Dorn, F.; Fomin, F. V.; Lokshtanov, D.; Raman, V.; and Saurabh, S. 2013. Beyond bidimensionality: Parameterized subexponential algorithms on directed graphs. *Inf. Comput.*, 233: 60–70.

Feder, T.; and Greene, D. 1988. Optimal algorithms for approximate clustering. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, 434–444.

Fomin, F. V.; Lokshtanov, D.; Kolay, S.; Panolan, F.; and Saurabh, S. 2020. Subexponential Algorithms for Rectilinear Steiner Tree and Arborescence Problems. *ACM Trans. Algorithms*, 16(2): 21:1–21:37.

Fomin, F. V.; Lokshtanov, D.; Marx, D.; Pilipczuk, M.; Pilipczuk, M.; and Saurabh, S. 2016. Subexponential Parameterized Algorithms for Planar and Apex-Minor-Free Graphs via Low Treewidth Pattern Covering. In Dinur, I., ed., *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*, 515–524. IEEE Computer Society.

Goel, A.; Indyk, P.; and Varadarajan, K. R. 2001. Reductions among high dimensional proximity problems. In Kosaraju, S. R., ed., *Proceedings of the Twelfth Annual Symposium on Discrete Algorithms, January 7-9, 2001, Washington, DC, USA*, 769–778. ACM/SIAM.

Gonzalez, T. F. 1985. Clustering to Minimize the Maximum Intercluster Distance. *Theor. Comput. Sci.*, 38: 293–306.

Hochbaum, D. S.; and Shmoys, D. B. 1986. A unified approach to approximation algorithms for bottleneck problems. *Journal of the ACM (JACM)*, 33(3): 533–550.

Impagliazzo, R.; Paturi, R.; and Zane, F. 2001. Which problems have strongly exponential complexity. *Journal of Computer and System Sciences*, 63(4): 512–530.

Jansen, B. 2010. Kernelization for maximum leaf spanning tree with positive vertex weights. In *International Conference on Algorithms and Complexity*, 192–203. Springer.

Johnson, W. B.; and Lindenstrauss, J. 1984. Extensions of Lipschitz mappings into a Hilbert space 26. *Contemporary mathematics*, 26.

Kolliopoulos, S. G.; and Rao, S. 2007. A nearly linear-time approximation scheme for the Euclidean k-median problem. *SIAM Journal on Computing*, 37(3): 757–782.

Kumar, A.; Sabharwal, Y.; and Sen, S. 2010. Linear-time approximation schemes for clustering problems in any dimensions. *J. ACM*, 57(2): 5:1–5:32.

Lichtenstein, D. 1982. Planar formulas and their uses. *SIAM journal on computing*, 11(2): 329–343.

Mahajan, M.; Nimbhorkar, P.; and Varadarajan, K. R. 2012. The planar k-means problem is NP-hard. *Theor. Comput. Sci.*, 442: 13–21.

Megiddo, N.; and Supowit, K. J. 1984. On the complexity of some common geometric location problems. *SIAM journal on computing*, 13(1): 182–196.

Mentzer, S. G. 1988. Approximability of metric clustering problems. *Manuscript. https://www. academia. edu/23251714/Approximability of Metric Clustering Problems*.

Nederlof, J. 2020. Detecting and counting small patterns in planar graphs in subexponential parameterized time. In Makarychev, K.; Makarychev, Y.; Tulsiani, M.; Kamath, G.; and Chuzhoy, J., eds., *Proccedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, Chicago, IL, USA, June 22-26, 2020*, 1293–1306. ACM.

Shiloach, Y. 1976. *Linear and planar arrangements of graphs*. The Weizmann Institute of Science (Israel).