A Single-Swap Local Search Algorithm for k-means of Lines

Ting Liang[1], **Xiaoliang Wu**[1], **Junyu Huang**[1], **Jianxin Wang**[1,2], **Qilong Feng**[1,*]
[1]School of Computer Science and Engineering, Central South University
[2]The Hunan Provincial Key Lab of Bioinformatics, Central South University,
Changsha 410083, China
tingliang@csu.edu.cn, xiaoliangwucsu@163.com, junyuhuangcsu@foxmail.com,
jxwang@mail.csu.edu.cn, csufeng@mail.csu.edu.cn

Abstract

Clustering is a fundamental problem that has been extensively studied over past few decades, with most research focusing on point-based clustering such as k-means, k-median, and k-center. However, numerous real-world applications, such as motion analysis, computer vision, and missing data analysis, require clustering over structured data, including lines, time series and affine subspaces (flats), where traditional point-based clustering algorithms often fall short. In this paper, we study the k-means of lines problem, where the input is a set L of lines in $\mathbb{R}^d$, and the goal is to find k centers C in $\mathbb{R}^d$ such that the sum of squared distances from each line in L to its nearest center in C is minimized. The local search algorithm is a well-established strategy for point-based k-means clustering, known for its efficiency and provable approximation guarantees. However, extending local search algorithm to the k-means of lines problem is nontrivial, as the capture relation used in point-based clustering does not generalize to the line setting. This is because that the point-to-line distance function lack the triangle inequality property that supports geometric analysis in point-based clustering. Moreover, since lines extend infinitely in space, it is difficult to identify effective swap points that can significantly reduce the clustering cost. To overcome above obstacles, we introduce a *proportional capture relation* that links optimal and current centers based the assignment proportions of lines, enabling a refined analysis that bypasses the triangle inequality barrier. We also introduce a *CrossLine* structure, which provides a principled discretization of the geometric space around line pairs, and ensures coverage of high-quality swap points essential for local search, thereby enabling effective execution of the local search process. Consequently, based on the proposed components, we develop the first single-swap local search algorithm for the k-means of lines problem, achieving a $(500 + \varepsilon)$-approximation in polynomial time for low-dimensional Euclidean space.

1 Introduction

Clustering is one of the most popular problems in machine learning, and has lots of applications in data mining, image classification, etc. The goal of clustering is to partition a given set of points into several disjoint clusters such that similar points end up in the same cluster, and dissimilar points are separated into different clusters [23]. Among classical clustering models [11, 12, 25], k-means is one of the most extensively studied, aiming to minimize the total squared distance from each data point to

*Corresponding Author

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

its nearest center. More formally, given a set $P \subseteq \mathbb{R}^d$ of n data points in a d-dimensional Euclidean space and a positive integer k, the goal of k-means is to find a set $C \subset \mathbb{R}^d$ of k centers such that the objective $\sum_{p \in P} \min_{c \in C} \|c - p\|^2$ is minimized.

Although the k-means problem is NP-hard [21, 15], numerous approximation algorithms have been developed. One of the most widely used heuristics in practice is Lloyd's algorithm [17], even though it lacks provable worst-case approximation guarantees under general data distributions. To establish approximation guarantees, researchers introduced a series of algorithms based on primal-dual and randomized rounding techniques [2, 10], among which a primal-dual method leveraging nested quasi-independent sets [5] achieves the best-known approximation ratio of 5.912. Meanwhile, local search has emerged as a practical and theoretically grounded framework for the k-means problem. Several local search algorithms [6, 7] achieved a $(1+\varepsilon)$-approximation under the assumption of fixed dimension d or a constant number of clusters k. To further enhance the initialization, Arthur and Vassilvitskii [3] proposed an efficient seeding algorithm, known as k-means++, which achieves an $O(\log k)$-approximation in expectation. Subsequent studies [26, 1, 19] showed that k-means++ achieves constant-factor approximations when allowed to open $O(k)$ centers. Building upon this, Lattanzi and Sohler [16] proposed the LS++ algorithm by combining k-means++ seeding with local search, achieving a constant-factor approximation in $O(ndk^2 \log\log k)$ time. Choo *et al.* [4] further improved its efficiency, demonstrating that an $O(1/\varepsilon^3)$-approximation can be obtained using only $O(\varepsilon k)$ local search iterations. Under the assumption that each optimal cluster has size $\Omega(n/k)$, Huang *et al.* [13] proposed a fast and practical local search algorithm for k-means problem, and achieved a $(100+\varepsilon)$ approximation in expectation. More recently, Huang *et al.* [14] gave the first multi-swap local search algorithm with running time linearly dependent on the data size, and achieved a $\left(50\left(1+\frac{1}{t}\right)+\varepsilon\right)$-approximation with any swap size $t \geq 2$ in time $O\left(ndk^{2t+1}\log\left(\varepsilon^{-1}\log k\right)\right)$, improving the previous 509-approximation under linear-time constraints.

While these results provide strong theoretical and practical foundations for point-based clustering, many real-world applications involve structured or incomplete data that are more naturally represented as geometric objects such as one-dimensional affine subspaces (i.e., lines). For example, in computer vision [18, 20], when multiple cameras are used to observe a set of fixed objects, each camera provides a directional observation, that is, a direction vector from the camera to the object. By combining each camera with its corresponding direction, a set of lines can be obtained. Clustering these lines helps identify different objects and recover their locations, since the lines corresponding to the same object tend to intersect at or near the object's true location. This motivates the study of clustering problems where the input consists of lines rather than points. Although conceptually similar to classical point-based formulations, line clustering introduces unique algorithmic and geometric challenges, and remains relatively underexplored in the literature. Among the few existing studies, Har-Peled and Varadarajan [8] studied the problem of finding the minimum intersection radius of a set of lines or affine subspaces, and introduced a substitute for the triangle inequality based on a novel analog of Helly's theorem. Aronov *et al.* [9] studied the problem of finding k minimum-radius balls in Euclidean space that intersect all given lines, and developed a 2-approximation algorithm for $k = 2, 3$ with quasi-linear running time. This algorithm is based on a new Helly-type theorem, and provides a geometric framework for line-based clustering representations of incomplete data. Ommer *et al.* [22] considered the k-means of lines problem, and proposed the first heuristic algorithm without theoretical guarantee and time constraints. Perets *et al.* [24] proposed an approximation algorithm in $\mathbb{R}^2$ that starts with a bi-criteria approximate solution, and then uses the coresets technique, achieving a $(1+\varepsilon)$-approximation in time $O(n(\log n/\varepsilon))^{O(k)}$ for any given parameter $\varepsilon > 0$. Further, Marom *et al.* [20] computed an ε-coreset of size $O(dk^{O(k)}\log(n)/\varepsilon^2)$ with running time $O(d^3 n \log(n) k \log k + (d/\varepsilon)^2 + ndk^{O(k)})$ for the k-means of lines problem.

In this paper, we focus on the k-means of lines problem. Given a set L of n lines in $\mathbb{R}^d$ and an integer $k \in \mathbb{N}_{\geq 1}$, the objective is to find a set of k centers $C \subset \mathbb{R}^d$ that minimizes the total squared distance from each line $\ell \in L$ to its nearest center in C. In this paper, we present the first local search algorithm with provable approximation guarantees for the line-based setting. Although local search has been extensively analyzed for point-based k-means due to its simplicity and strong empirical performance, extending it to the line setting introduces several fundamental challenges. To motivate our algorithm, we first review the classical local search framework and highlight the key obstacles in adapting it to line clustering.

- The standard analysis of local search algorithm for k-means mainly relies on a capture relation, in which each optimal center is captured by the nearest center in the current solution, such that it is easy to analyze the change in clustering cost incurred when points are reassigned in the process of local search. However, the above relation is not workable for the line setting, since the assignments of lines may be determined by an optimal center that is not the closest in Euclidean distance, and reassigning lines based solely on geometric closeness can lead to a significant increase in clustering cost.
- The sampling-based method is a commonly used technique in local search. However, unlike the point setting, it is more challenging to sample a point close to the optimal center from these lines, since the optimal center is unknown and each line spans a continuous space of candidate points, even we show that the cost of a high-cost cluster can be effectively approximated by lines that are close to their optimal center in the k-means of lines problem.

Our Contributions. To overcome these obstacles, we introduce a new proportional capture relation and a structured sampling technique tailored to the geometry of lines. Specifically, we define that an optimal center is captured by a current center if it receives the largest fraction of lines from that center, enabling a more faithful representation of line-to-center influence. To facilitate effective sampling, we further design a discretization structure, called *CrossLine*, which guarantees the existence of a point near the optimal center within a bounded region defined by two sampled lines. This structure allows us to recover, with provable guarantees, a good replacement center from a finite set of representative points. We formalize the proposed *proportional capture relation* and *CrossLine* structure in Section 3, and discuss some key properties. Building on these foundations, we develop the first local search algorithm for the k-means of lines problem with provable approximation guarantees. The main contributions of this paper are summarized as follows:

- In two-dimensional space, we introduce a *proportional capture relation* and a geometry-aware discretization structure *CrossLine*, both tailored to the challenges of line-based clustering. These tools enable an accurate representation of line-to-center assignments, and support efficient sampling from continuous line domains.
- In high-dimensional space, the existence of skew lines makes the *CrossLine* structure in $\mathbb{R}^2$ inadequate. To address this issue, we redefine the *CrossLine* structure to accommodate high-dimensional geometry, ensuring that the coverage guarantees remain valid under arbitrary line orientations.
- Building upon these tools, we propose the first single-swap local search algorithm for the k-means of lines problem, achieving a $(500 + \varepsilon)$-approximation, and running in polynomial time on low-dimensional Euclidean space.
- Extensive experiments on both synthetic and real-world datasets show that our algorithm consistently outperforms coreset-based baselines in both efficiency and clustering quality.

Formally, we have the following results of this paper.

Theorem 1. *For any $\varepsilon > 0$, there exists a local search algorithm for the k-means of lines problem, which achieves a $(500 + \varepsilon)$-approximation in expectation with polynomial time on low-dimensional Euclidean space.*

2 Preliminaries

For any positive integer $m \in \mathbb{N}^{\geq 1}$, let $[m] = \{1, \ldots, m\}$. Given a set $P \subseteq \mathbb{R}^d$ of points, for any $p, q \in P$, let $\delta(p, q) = \|p - q\|_2^2$ denote the square Euclidean distance between p and q. Given any two set $A, B \subseteq \mathbb{R}^d$ of points, for any $p \in A$, let $\delta(p, B) = \min_{q \in B} \delta(p, q)$ denote the shortest square distance from p to any point in B. Further, let $\delta(A, B) = \min_{p \in A, q \in B} \delta(p, q)$ denote the square distance between A and B, and let $\Delta(A, B) = \sum_{p \in A} \delta(p, B)$ denote the sum of distances from each point in A to its nearest point in B. Given a nonempty subset $C \subseteq P$ of centers, for any $c \in C$ and a radius r, a ball $\mathcal{B}(c, r)$ is the set of points that are within a distance r from c, i.e., $\mathcal{B}(c, r) = \{p \in P \mid \delta(c, p) \leq r\}$. Given a set L of lines in $\mathbb{R}^d$ and a set $C \subseteq \mathbb{R}^d$ of points, for any $\ell \in L$, let $\delta(\ell, C) = \min_{c \in C} \delta(\ell, c)$, where $\delta(\ell, c)$ is the shortest square distance from any point on the line ℓ to c. Formally, the k-means of lines (denoted as k-ML) problem considered in this paper can be defined as follows.

Algorithm 1: SLS-k-ML

Input: An instance (L, d, k) of the k-ML problem and a parameter T
Output: A set $C \subset \mathbb{R}^d$ of at most k centers

$P \leftarrow$ CENTROID-SET(L);
$C \leftarrow$ Sample k points from P randomly;
for $i \in \{1, 2, \ldots, T\}$ **do**
  Sample two lines ℓ_1, ℓ_2 from L with probability $b_\ell = \frac{\Delta(\{\ell\}, C)}{\sum_{\ell' \in L} \Delta(\{\ell'\}, C)}$ $(\ell = \{\ell_1, \ell_2\})$;
  $\mathcal{M} \leftarrow$ the point set returned by *CrossLine* of (ℓ_1, ℓ_2);
  for each point $p \in \mathcal{M}$ **do**
    if $\exists\ q \in C$, *s.t.* $\Delta(L, C \backslash \{q\} \cup \{p\}) < \Delta(L, C)$ **then**
      $C \leftarrow C \backslash \{q\} \cup \{p\}$;
return C.

Algorithm 2: CENTROID-SET

Input: A finite set L of n lines in $\mathbb{R}^d$
Output: A set P of points

$P \leftarrow \emptyset$;
for each line $\ell \in L$ **do**
  $P_\ell \leftarrow \emptyset$;
  for each line $\ell' \in L \backslash \{\ell\}$ **do**
    $\pi_{\ell'}(\ell) \leftarrow$ the closest point on ℓ to ℓ';
    $P_\ell \leftarrow P_\ell \cup \{\pi_{\ell'}(\ell)\}$;
  $P \leftarrow P \cup P_\ell$;
return P.

Definition 1 (the k-ML problem)**.** *Given a set $L \subseteq \mathbb{R}^d$ of n lines in a d-dimensional Euclidean space and a positive integer k, the goal is to find a set $C \subset \mathbb{R}^d$ of k centers such that the objective $\Delta(L, C) = \sum_{\ell \in L} \delta(\ell, C)$ is minimized.*

In this paper, we assume that all lines in L are pairwise non-parallel (possibly intersecting or skew), ensuring distinct directional relationships and avoiding degenerate configurations. Given an instance (L, d, k) of the k-ML problem, C is called a feasible solution of this instance if $C \subseteq \mathbb{R}^d$ is a set of k points. Throughout this paper, let $C^* = \{c_1^*, \ldots, c_k^*\}$ be the optimal solution and $\{L(c_1^*), \ldots, L(c_k^*)\}$ be the corresponding optimal clusters by assigning each line in L to its closest centers in C^*. Let $\tau = \Delta(L, C^*)$ be the cost of the optimal solution C^*.

3 Local Search Algorithm with Single-Swap Strategy

The local search algorithm for the k-means problem typically begins with a candidate set of centers and iteratively replace one or more centers, as in single-swap or multi-swap strategies, to progressively reduce the clustering cost. Unlike classical k-means problem, which benefits from effective initialization algorithms such as k-means++ [16], the k-ML problem lacks a well-established algorithm for constructing a high-quality initial solution. Therefore, it is necessary to explicitly construct an initial solution for the k-ML problem. Moreover, extending local search algorithm to the k-ML problem introduces additional challenges beyond initialization. Specifically, the capture relation used in k-means problem does not hold in the line setting, as line assignments are governed by projection distances, and reassigning a line to the optimal center closest to its current center may lead to an increase in clustering cost. In addition, it is more challenging to sample a point that is close to an optimal center from given lines, since the optimal center is unknown and each line spans a continuous space of candidate points.

To overcome these challenges, we develop a single-swap local search algorithm for the k-ML problem, referred to SLS-k-ML, which is presented in Algorithm 1. Our algorithm SLS-k-ML consists of

two stages. In the first stage (steps 1–2), the algorithm starts with a centroid set P obtained by CENTROID-SET that reduces the lines in L to a set of points. Then, we obtain a feasible solution C from the constructed centroid set P with an approximation guarantee. In the second stage (steps 3-8), we execute the single-swap local search with T iterations. In each iteration, the algorithm first to sample two lines ℓ_1, ℓ_2 from L with a probability, and then constructs candidate swap pairs based on the *CrossLine* structure with respect to ℓ_1, ℓ_2 to find improvements. The *CrossLine* structure ensures that each optimal center can be effectively approximated by at least one grid point. Under the proposed *proportional capture relation*, such a grid point can be used to replace a current center, resulting in a reduction of the clustering cost.

3.1 Construct an Initial Solution

This section shows how to construct an initial solution with an approximation guarantee for a given instance of the k-ML problem in polynomial time. It is known that the local search algorithm typically begins with an initial solution. However, the k-ML problem lacks a well-established algorithm for generating a high-quality initial solution. Therefore, we first need to construct such an initial solution as a foundation for the local search process. To this end, we first construct a candidate point set P by the CENTROID-SET procedure from [20] (see Algorithm 2). Given a set L of n lines in $\mathbb{R}^d$, for each line, CENTROID-SET computes its the closest points to all other lines, and collects them into a candidate set P of $O(n^2)$ points. We then obtain the initial solution C by sampling k points uniformly at random from P. In the following, we show that the initial solution C provides a constant-factor approximation to the optimal clustering cost, and can be computed in polynomial time.

Lemma 2. *Given an instance $\mathcal{I} = (L, d, k)$ of the k-ML problem, let C be the set of points returned by step 2 of SLS-k-ML. Then, for a finite constant ρ, we have $\Delta(L, C) \leq \rho \cdot \Delta(L, C^*)$ (see Appendix A.1 for proof).*

3.2 Single-Swap Local Search in $\mathbb{R}^2$

To facilitate understanding of our proposed algorithm, this section focuses on the single-swap local search process in $\mathbb{R}^2$. Specifically, we construct a swap pair between the center in C and a candidate center in $\mathcal{M}$ obtained by the *CrossLine* structure, which is related to two lines sampled from L with a specified probability. If the clustering cost is reduced, we update the current solution by replacing the center in C with the candidate center from $\mathcal{M}$. In the following, we show that if the current solution has a high cost (greater than 500τ), then the clustering cost can be reduced with a certain probability in each iteration.

Lemma 3. *Let $C \subseteq \mathbb{R}^2$ be a set of centers with $\Delta(L, C) > 500\tau$, and let C' be the set of centers obtained in the t-th ($t \in [T]$) iteration of SLS-k-ML in step 8. Then, with probability at least $\Omega(\zeta^{-1})$, we have $\Delta(L, C') \leq (1 - \frac{1}{100k})\Delta(L, C)$, where ζ is a constant.*

Before proving Lemma 3, we first analyze a single-swap (w.l.o.g. t-th iteration) in steps 3–8 of SLS-k-ML. Let $C = \{c_1, \ldots, c_k\}$ denote the set of centers before the t-th swap operation, and let $\{L(c_1), \ldots, L(c_k)\}$ be the corresponding partition of L induced by assigning the lines to the closest center in C. For clarity, when the specific index is not relevant, we omit it and simply write, for example, $c \in C$. Motivated by the local search for k-means [16], we adopt a similar strategy to analyze the cost reduction by comparing the current and optimal center sets. Specifically, the algorithm in [16] introduces the notion of a capture relation, which states that an optimal center can be captured by the nearest center in the current solution. However, the above relation relies on a point-to-point assignment setting, and is not workable for our problem, since the data consist of lines and the clustering cost is based on projection distances. For example, in the k-means setting, for each $c^* \in C^*$, assume that c is the closest center in C to c^*. By the capture relation, the optimal center c^* is captured by c, which means that c is the nearest center to c^* among all centers in C. In the process of analyzing local search, we consider replacing c with a point near to c^*, and reassigning the points in the cluster of c, thereby achieving a reduction in clustering cost. However, in the line setting, the lines in the cluster of c may be significantly closer to another optimal center in C^*. In such case, if we reassign these lines to a point near to c^*, the clustering cost will drastically increase, indicating that the original capture relation does not generalize to the k-ML problem. Therefore, we propose a *proportional capture relation* better suited to the line setting: an optimal center $c^* \in C^*$ is captured by a center $c \in C$ if c is assigned the largest proportion of lines from the cluster of c^* among all center in C. Clearly, a center $c \in C$ may capture more than one optimal centers, and every optimal

center is captured by exactly one center from C. For a center $c \in C$, let $\mathcal{N}(c)$ be the set of centers in C^* captured by c. For each optimal center $c^* \in C^*$, let $\mathcal{N}^{-1}(c^*)$ be the center in C capturing c^*. Further, we introduce two types of matched swap pairs that correspond to the two cases encountered in local search as follows.

The type-1 matched swap pair: For any $c \in C$, if $|\mathcal{N}(c)| = 1$ (i.e., $\mathcal{N}(c)$ contains only one optimal center c^*), we define (c, c^*) as a type-1 matched swap pair. Let C_S denote all centers in C satisfying the property of type-1 matched swap pair. For simplicity, we introduce a mapping $\psi_S : C_S \to C^*$ that maps each center in C_S to an optimal center in C^* such that $(c, \psi_S(c))$ is a type-1 matched swap pair. For each type-1 matched swap pair $(c, \psi_S(c))$ $(c \in C_S)$, we replace the current center c with a point near the optimal center $\psi_S(c)$. For a type-1 matched swap pair, the lines in cluster c $(c \in C_S)$ can be divided into two groups: The first group consists of lines that belong to both the current cluster of c and the optimal cluster of $\psi_S(c)$. These lines are reassigned to a point near their optimal center, leading to a reduction in clustering cost. The second group consists of lines that belong to in the current cluster of c but not to the optimal cluster of $\psi_S(c)$. These lines are reassigned to other centers in C, which may increase the clustering cost.

The type-2 matched swap pair: For any $c \in C$, if $|\mathcal{N}(c)| = 0$ (i.e., c does not capture any optimal center), we can find another center $c' \in C$ with $|\mathcal{N}(c')| > 1$, and assign one of its captured optimal center $c^* \in \mathcal{N}(c')$ to form a type-2 matched pair (c, c^*). Let C_N denote all centers in C satisfying the property of type-2 matched swap pair. For simplicity, we introduce a mapping $\psi_N : C_N \to C^*$ that maps each center in C_N to an optimal center in C^* such that $(c, \psi_N(c))$ is a type-2 matched swap pair. For a type-2 matched swap pair $(c, \psi_N(c))$ $(c \in C_N)$, we replace the lonely center c with a point near the optimal center $\psi_N(c)$. Similar to the type-1 case, the lines in cluster c $(c \in C_N)$ also can be divided into two groups: The first group consists of lines that belong to the optimal cluster of $\psi_N(c)$. These lines are reassigned to a point near their optimal center, leading to a reduction in clustering cost. The second group consists of lines that belong to the current cluster of c. These lines are reassigned to other centers in C, which may increase the clustering cost.

Through this construction, each optimal center in C^* is paired with a current center in C via exactly one type-1 or type-2 matched swap pair. Each swap pair involves both clustering cost reduction (from lines reassigned to a sampled point near their optimal center) and clustering cost increase (from lines reassigned to potentially farther centers). In the following, we will argue that with high probability, a point can be sampled to perform the swap operation such that the resulting reduction in clustering cost significantly bigger than the corresponding increase. For a type-1 or type-2 matched swap pair, we refer to the clustering cost increase as the reassignment cost, which is formally defined below.

Definition 2. *For any $c \in C_S$, the reassignment cost of the type-1 matched swap pair $(c, \psi_S(c))$ is defined as $\eta_1(c) = \Delta(L \backslash L(c^*), C \backslash \{c\}) - \Delta(L \backslash L(c^*), C)$, where $c^* = \psi_S(c)$. Similarly, for any $c \in C_N$, the reassignment cost of the type-2 matched swap pair $(c, \psi_N(c))$ is defined as $\eta_2(c) = \Delta(L, C \backslash \{c\}) - \Delta(L, C)$.*

Lemma 4. *For any $c \in C_S$ (or $c \in C_N$), the reassignment cost $\eta_1(c)$ (or $\eta_2(c)$) is at most $\frac{1}{5}\Delta(L(c), C) + 24\Delta(L(c), C^*)$ (see Appendix A.2 for proof).*

Lemma 4 implies that there exists a good bound on the reassignment cost. To proceed, we discuss the following two cases based on the cost contributions of type-1 and type-2 matched pairs: **Case 1:** $\sum_{c \in C_S} \Delta(L(\psi_S(c)), C) > \frac{1}{3} \cdot \Delta(L, C)$. **Case 2:** $\sum_{c \in C_S} \Delta(L(\psi_S(c)), C) \leq \frac{1}{3} \cdot \Delta(L, C)$, thus $\sum_{c \in C^* \backslash \cup_{c \in C_S} \psi_S(c)} \Delta(L(c), C) \geq \frac{2}{3} \cdot \Delta(L, C)$.

3.2.1 The Analysis of Case 1

Recall that the definition of type-1 matched swap pair $(c, \psi_S(c))$ $(c \in C_S)$, we replace the current center c with a point near the optimal center $\psi_S(c)$, with the goal of achieving a significant reduction in clustering cost. We refer to such clusters that yield a significant cost reduction as *good clusters*, and we formalize this notion below.

Definition 3 (*Good type-1 cluster*). *For any $c \in C_S$, the cluster $L(c)$ is called a good type-1 cluster, if $\Delta(L(\psi_S(c)), C) - \eta_1(c) - 9\Delta(L(\psi_S(c)), \{\psi_S(c)\}) > \frac{1}{100k}\Delta(L, C)$. Otherwise, $L(c)$ is called a bad type-1 cluster with $(c, \psi_S(c))$.*

For simplicity, let C_S^g denote the set of centers in C_S for which the corresponding cluster is a good type-1 cluster. Let $C_S^b = C_S \backslash C_S^g$. The Definition 3 estimates the cost reduction of replacing c with a

point close to $\psi_S(c)$ by considering a cluster that reassigns the lines in $L(c)\backslash L(\psi_S(c))$, and assigns all lines in $L(\psi_S(c))$ to the new center. We now aim to show that a good type-1 cluster can be sampled with high probability. To this end, we first argue that the total clustering cost contributed by good type-1 clusters is sufficiently large. Then, we show that with good probability, we can sample a point that is close to the center.

Lemma 5. *If $3\sum_{c\in C_S}\Delta(L(\psi_S(c)),C) > \Delta(L,C)$, then we have $\sum_{c\in C_S\cap C_S^g}\Delta(L(\psi_S(c)),C) \geq \frac{1}{50}\cdot\Delta(L,C)$ (see Appendix A.3 for proof).*

Then, we show that with good probability, a point close to the optimal center of a good type-1 cluster is sampled. To prove this, we first establish the following the technical lemma.

Lemma 6. *Let L be a set of lines in $\mathbb{R}^d$, and let $c, c' \in \mathbb{R}^d$ be two points. Then, the total squared distance from the lines in L to c satisfies: $\Delta(L,\{c\}) \leq 3|L|\cdot d(c,c') + 6\cdot\Delta(L,\{c'\})$, where $\Delta(L,\{c\}) = \sum_{\ell\in L}\delta(\ell,c)$ and $\delta(\ell,c)$ denote the squared Euclidean distance from line ℓ to point c, respectively (see Appendix A.4 for the proof).*

This lemma, used in the proof of Lemma 6, shows that the clustering cost of any two centers is related to the distance between the centers. In the following lemma, we show that if a cluster with respect to C incurs a large clustering cost, then the subset of lines located near its center likewise contributes a comparably high cost.

Lemma 7. *Let Q be a set of lines in $\mathbb{R}^d$, let $C \subseteq \mathbb{R}^d$ be a set of k centers, and let c^* be the optimal single center for Q. Assume $\alpha \geq 9$. If the cost under C satisfies $\Delta(Q,C) \geq \alpha\cdot\Delta(Q,\{c^*\})$, then there exists a subset $R \subseteq Q$ with $\Delta(R,C) \geq \frac{\alpha-1}{24}\cdot\Delta(Q,\{c^*\})$, where $R = \{\ell\in Q \mid \delta(\ell,c^*) \leq \frac{2}{|Q|}\cdot\Delta(Q,\{c^*\})\}$ (see appendix A.5 for proof).*

To find a point near the optimal center along a line, we first evaluate the probability of sampling a line close to the optimal center. For any $c \in C_S$, we have $\Delta(L(\psi_S(c)),C) \geq 9\cdot\Delta(L(\psi_S(c)),\{\psi_S(c)\})$ due to Definition 3. Then, by applying Lemma 7 with $Q = L(\psi_S(c))$ and $\alpha = 9$, we get that $\Delta(\mathcal{A},C) \geq \frac{\alpha-1}{24}\cdot\Delta(L(\psi_S(c)),\{\psi_S(c)\}) = \frac{\alpha-1}{24\alpha}\cdot\Delta(L(\psi_S(c)),C) \geq \frac{1}{27}\cdot\Delta(L(\psi_S(c)),C)$, where $\mathcal{A}$ is the set of lines within distance $\frac{2\Delta(L(\psi_S(c)),\{\psi_S(c)\})}{|L(\psi_S(c))|}$ from $\psi_S(c)$. By Lemma 5, we can obtain that $\sum_{c\in C_S\cap C_S^g}\Delta(L(\psi_S(c)),C) \geq \frac{1}{50}\cdot\Delta(L,C)$. Thus, we conclude that the total cost over all sets $\mathcal{A}$ from good clusters satisfies $\sum_{c\in C_S\cap C_S^g}\Delta(\mathcal{A},C) \geq \frac{1}{27\cdot 50}\cdot\Delta(L,C)$, implying that the probability of sampling a line from $\bigcup_{c\in C_S\cap C_S^g}\mathcal{A}$ is at least $\frac{1}{1500}$. However, since $\mathcal{A}$ is a set of lines rather than points, we cannot directly select a line as a candidate center. Instead, we need to generate a set of points from the lines in $\mathcal{A}$, and ensure that at least one of them lies within distance $\frac{2\Delta(L(\psi_S(c)),\{\psi_S(c)\})}{|L(\psi_S(c))|}$ from the optimal center. To proceed, we introduce a geometric discretization structure, referred to as r-*CrossLine*, and prove that the set of points returned by this structure contains at least one point that lies within distance r of any point inside the structure.

Definition 4 (r-*CrossLine* in $\mathbb{R}^2$)**.** *Let ℓ_1 and ℓ_2 be two intersecting lines in $\mathbb{R}^2$, intersecting at a point $p = \ell_1\cap\ell_2$. For each line ℓ_i ($i = 1, 2$), we construct two lines parallel to ℓ_i by shifting it along its normal direction by $\pm r$. Let $\mathcal{L}_1 = \{\ell_1^-, \ell_1, \ell_1^+\}$ and $\mathcal{L}_2 = \{\ell_2^-, \ell_2, \ell_2^+\}$ denote the sets of three parallel lines derived from ℓ_1 and ℓ_2, respectively. The r-CrossLine is defined as the union of all regions enclosed by the lines in $\mathcal{L}_1$ and $\mathcal{L}_2$, and CrossLine$_r(\ell_1,\ell_2) = \{\ell_i\cap\ell_j \mid \ell_i\in\mathcal{L}_1,\ \ell_j\in\mathcal{L}_2\}$ denotes the set of all pairwise intersection points.*

Based on Definition 4, we prove that for any point inside the r-*CrossLine*, there exists at least one grid point within distance r of it.

Lemma 8. *Let ℓ_1 and ℓ_2 be two intersecting lines in $\mathbb{R}^2$, and let $r > 0$ be a constant. Then, for any point x within r-CrossLine, there exists at least one grid point from CrossLine$_r(\ell_1,\ell_2)$ whose distance to x is at most r. (see Appendix A.6 for proof).*

As mentioned earlier, $\mathcal{A}$ is the set of lines within distance $\frac{2\Delta(L(\psi_S(c)),\{\psi_S(c)\})}{|L(\psi_S(c))|}$ from $\psi_S(c)$, and the probability of sampling a line from the set $\bigcup_{c\in C_S\cap C_S^g}\mathcal{A}$ is at least $\frac{1}{1500}$. Therefore, the probability that both ℓ_1 and ℓ_2 lie in $\mathcal{A}$ is at least $\zeta = \left(\frac{1}{1500}\right)^2$. Assuming that $\ell_1,\ell_2 \in \bigcup_{c\in C_S\cap C_S^g}\mathcal{A}$, $r = \frac{2\Delta(L(\psi_S(c)),\{\psi_S(c)\})}{|L(\psi_S(c))|}$, let *CrossLine*$_r(\ell_1,\ell_2)$ be the point set obtained by Definition 4 based on ℓ_1

and ℓ_2. Then, by Lemma 8, *CrossLine*$_r(\ell_1, \ell_2)$ contains at least one point within distance r of the optimal center, which can be used as a high-quality swap candidate. However, since the radius r is defined in terms of the (unknown) optimal cost, we adopt a multi-scale, data-driven schedule: let $r_{\min}$ and $r_{\max}$ be empirical scales around the intersection (e.g., lower/upper quantiles of the distances from the intersection to the candidate set returned by Algorithm 2). We progressively explore the search radii $r_t \in [\frac{r_{\min}}{10}, r_{\max}]$, constructing an r_t-*CrossLine* around the intersection at each scale and performing a single-swap at every step until an improvement is found. By construction, at least one of the radii r_t lies within a constant factor of the theoretical radius r. At this scale, Lemma 8 guarantees a constant probability of success. By Definition 3, if such a point p is obtained, we can swap it with c to get a new clustering with cost at most $\Delta(L, C \setminus \{c_h\} \cup \{p\}) \leq \Delta(L, C) - \Delta(L(\psi_S(c)), C) + \eta_1(c) + \Delta(L(\psi_S(c)), \{c\})$. By Lemma 6, we have that $\Delta(L(\psi_S(c)), \{c\}) \leq 9 \cdot \Delta(L(\psi_S(c)), \{\psi_S(c)\})$. Thus, with a constant probability, the new clustering has cost at most $\Delta(L, C) - (\Delta(L(\psi_S(c)), C) - \eta_1(c) - 9\Delta(L(\psi_S(c)), \{\psi_S(c)\})) \leq \left(1 - \frac{1}{100k}\right) \cdot \Delta(L, C)$.

3.2.2 The Analysis of Case 2

For type-2 matched swap pair $(c, \psi_N(c))$ $(c \in C_N)$, we similarly define clusters that yield a significant reduction in assignment cost under such swaps as *good clusters*, as formalized below.

Definition 5 (***Good type-2 cluster***). *For any $c \in C_N$, the cluster $L(c)$ is called a good type-2 cluster, if $\Delta(L(\psi_N(c)), C) - \eta_2(c) - 9\Delta(L(\psi_N(c)), \{\psi_N(c)\}) > \frac{1}{100k}\Delta(L, C)$. Otherwise, $L(c)$ is called a bad type-2 cluster with $(c, \psi_N(c))$.*

For simplicity, let C_N^g denote the set of centers in C_N for which the corresponding cluster is a good type-2 cluster. Let $C_N^b = C_N \backslash C_N^g$. The above definition estimates the cost reduction of replacing c with a point close to $\psi_S(c)$ by considering a cluster that reassigns the lines in $L(c)$ and assigns all lines in $L(\psi_S(c))$ to the new center. We now aim to show that a good type-2 cluster can be sampled with high probability. To this end, we first argue that the total cost contributed by good type-2 clusters is sufficiently large.

Lemma 9. *If $\sum_{c \in C \setminus C_S} \Delta(L(\psi_N(c)), C) \geq \frac{2}{3} \cdot \Delta(L, C)$, then we have $\sum_{c \in C \setminus C_S, c \in C_N^g} \Delta(L(\psi_N(c)), C) \geq \frac{1}{50} \cdot \Delta(L, C)$ (see Appendix A.7 for proof).*

Note that in this case we can now argue similarly as in the other case that sampling according to sum of squared distances will provide us with constant probability with a good center using Lemma 7. Indeed, since the sum of squared distances of lines in good centers is at least $\frac{1}{50}\Delta(L, C)$ by Lemma 9, it follows together with Lemma 7 that we sample a point from a good cluster $L(\phi_N(c))$ that is within distance two times the average cost of the cluster with probability $\frac{1}{1500}$. By the definition of good cluster, we have that such a point improves the cost of the current clustering by at least a factor of $\left(1 - \frac{1}{100k}\right)$. Thus, by combining Case 1 and Case 2, we complete the proof of Lemma 3. Now we complete the proof of Theorem 1. Let $\hat{C}$ be the set of centers returned by step 2 of SLS-k-ML, which achieves a ρ-approximation guarantee by Lemma 2. Let C be the set of centers returned by SLS-K-ML. By Lemma 3, we know that if the current cost exceeds 500τ, then with probability at least $\Omega(1/\rho)$, a single iteration of SLS-K-ML reduces the cost by a factor of $(1 - \frac{1}{100k})$. We introduce a random process X to analyze the change of the clustering cost. The process starts at $\Delta(L, \hat{C})$, and over $T = 100k\zeta \log \rho$ steps, it decreases multiplicatively by a factor of $\left(1 - \frac{1}{100k}\right)$ with probability $1/\zeta$ at each step, and finally increases additively by 500τ. Furthermore, $\mathbb{E}[X] = 500\,\tau + \Delta(L, \hat{C}) \cdot \sum_{i=0}^{T} \binom{T}{i} (\frac{1}{\zeta})^i (\frac{\zeta - 1}{\zeta})^{T-i} (1 - \frac{1}{100k})^i = \Delta(L, \hat{C})(1 - \frac{1}{100k\zeta})^{100k\zeta \log \rho} + 500\tau \leq \frac{\Delta(L, \hat{C})}{\rho} + 500\tau$. This implies that $\mathbb{E}[\Delta(L, C) \mid \hat{C}] \leq \frac{\Delta(L, \hat{C})}{\rho} + 500\tau$. Then, we have $\mathbb{E}[\Delta(L, C)] = \sum_{\hat{C}} \mathbb{E}[\Delta(L, C) \mid \hat{C}] \cdot \Pr(\hat{C}) = \sum_{\hat{C}} \Pr(\hat{C})(\frac{\Delta(L, \hat{C})}{\rho} + 500\tau) \leq (500 + \varepsilon)\tau$.

Running Time Analysis. The overall time complexity of SLS-k-ML is $O(n^2 + nk^2\zeta \log \rho)$. The first stage of constructing the initial solution takes $O(n^2)$ time, as it involves constructing the candidate set using the CENTROID-SET procedure, which takes $O(n^2)$ time, and sampling k points from the candidate set, which takes $O(1)$ time. To achieve a $(500 + \varepsilon)$-approximate solution, SLS-k-ML requires $O(k\zeta \log \rho)$ iterations. In each iteration, sampling two lines takes $O(\log n)$ time, constructing the *CrossLine* structure takes $O(1)$ time, finding swap points from the *CrossLine* output takes $O(9^d)$ time, and updating the distances between all lines and their nearest centers takes $O(nk)$

time. Therefore, the total running time of the iterative procedure is $O(nk^2\zeta \log \rho)$, and the overall time complexity of SLS-k-ML is $O(n^2 + nk^2\zeta \log \rho)$.

3.3 Extended to $\mathbb{R}^d$

In this section, we show how to extend our algorithm to the general case in $\mathbb{R}^d$. In $\mathbb{R}^d$, the the existence of skew lines makes the *CrossLine* structure in $\mathbb{R}^2$ inadequate, as lines may not intersect and can lie on different affine subspaces. To address this, we redefine the *CrossLine* structure to accommodate high-dimensional geometry, ensuring coverage guarantees remain valid under arbitrary line orientations. Moreover, in $\mathbb{R}^d$, the reassignment argument remains valid, while the bound must be adjusted to account for the dimensional dependence in the projection geometry. Formally, we restate the following result for $\mathbb{R}^d$.

Lemma 10. *For any $c \in C_S$ (or $c \in C_N$), the reassignment cost $\eta_1(c)$ (or $\eta_2(c)$) is at most $\frac{1}{5}\Delta(L(c), C) + 24\Delta(L(c), C^*)$ (see Appendix A.8 for proof).*

Note that although Lemma 4 and Lemma 10 yield the same conclusion, they are derived in different contexts and serve distinct purposes within the analysis. The second difference lies in the construction of the *CrossLine* structure. Recall that in $\mathbb{R}^2$, we generate a constant-size candidate set based on the intersection and perturbation of line pairs. In $\mathbb{R}^d$, this structure must be extended to cover a higher-dimensional geometric region induced by two non-parallel lines. Specifically, for each pair of lines, we construct a discretized grid over the intersection of their respective r-bounded neighborhoods, which forms a d-dimensional cross-like region. The number of candidate points grows polynomially with d, but remains bounded and efficiently enumerable. Note that in the process of constructing the candidate set in $\mathbb{R}^d$, we retain the same discretization resolution as in the planar case, but extend it across all d orthogonal directions to ensure sufficient coverage for sampling high-quality swap points. We now introduce the required extensions to the candidate set construction.

Definition 6 (r-*CrossLine* in $\mathbb{R}^d$)**.** *Let $\ell_1, \ell_2 \subset \mathbb{R}^d$ be two non-parallel lines. For each line ℓ_i ($i = 1, 2$), we define the set of 3^d axis-aligned parallel shifts as $\mathcal{L}_i = \{\ell_i^{\vec{s}} = \ell_i + \sum_{j=1}^{d} s_j \cdot r \cdot e_j \mid \vec{s} \in \{-1, 0, 1\}^d\}$, where e_j is the unit vector in the j-th coordinate direction. We define the r-CrossLine as the union of all regions enclosed by the lines in $\mathcal{L}_1$ and $\mathcal{L}_2$, and CrossLine$_r(\ell_1, \ell_2) = \{\ell \cap \ell' \mid \ell \in \mathcal{L}_1,\ \ell' \in \mathcal{L}_2,\ \ell \cap \ell' \neq \emptyset\}$. Assuming each pair of lines (ℓ, ℓ') intersects (e.g., due to careful shift alignment in a common plane), the total number of grid points is at most $|\text{CrossLine}_r(\ell_1, \ell_2)| \leq 3^{2d}$.*

Based on the r-*CrossLine* in $\mathbb{R}^d$, we restate Lemma 8 as follows.

Lemma 11. *Let ℓ_1 and ℓ_2 be two intersecting lines in $\mathbb{R}^d$, and let $r > 0$ be a constant. Then, for any point x within r-CrossLine, there exists at least one grid point from CrossLine$_r(\ell_1, \ell_2)$ whose distance to x is at most r.*

Running Time Analysis. The overall time complexity of SLS-k-ML is $O(n^2d^2 + 9^d ndk^2\zeta \log \rho)$. The first stage of constructing the initial solution takes $O(n^2d^2)$ time, as it involves constructing the candidate set using the CENTROID-SET procedure, which takes $O(n^2d^2)$ time, and sampling k points from the candidate set, which takes $O(1)$ time. To achieve a $(500 + \varepsilon)$-approximate solution, SLS-k-ML requires $O(k\zeta \log \rho)$ iterations. In each iteration, sampling two lines takes $O(\log n)$ time, constructing the *CrossLine* structure takes $O(1)$ time, finding swap points from the *CrossLine* output takes $O(9^d)$ time, and updating the distances between all lines and their nearest centers takes $O(ndk)$ time. Therefore, the total running time of the iterative procedure is $O(9^d ndk^2\zeta \log \rho)$, and the overall time complexity of SLS-k-ML is $O(n^2d^2 + 9^d nk^2\zeta \log \rho)$.

4 Experiments

In this section, we give empirical evaluations on the performances of our proposed algorithms. All algorithms are implemented and executed in Python. The experiments were done on a machine with i7-14700KF processor and 256GB RAM.

Datasets. We evaluate the performance of our algorithm on both synthetic datasets (SYN 1 with $n = 5000$, $d = 10$, and SYN2 with $n = 10000$, $d = 5$) and real-world OpenStreetMap datasets (RE 1 with $n = 476$, $d = 2$, and RE 2 with $n = 418$, $d = 2$) as used in [20]. For each dataset, we run both

Table 1: Experimental results of our SLS-k-ML algorithm and the coreset-based method.

Datasets	*Method*	k	*Min_Cost*	*Max_Cost*	*Avg_Cost*	*Std*	*Time(s)*
RE 1	SLS-k-ML(Ours)	10	**8.37E-10**	**1.72E-06**	**1.01E-07**	**2.73E-07**	**1.05**
	Coreset+sampling	10	8.17E-04	3.08E-02	6.24E-03	5.89E-03	6.05
	Coreset+exhaustive search	10	-	-	-	-	Over 12 hours
RE 1	SLS-k-ML(Ours)	3	**5.67E-06**	**1.15E-03**	**1.35E-04**	**2.00E-04**	**0.71**
	Coreset+sampling	3	1.57E-02	8.78E-01	1.39E-01	1.95E-01	4.08
	Coreset+exhaustive search	3	1.10E-03	2.11E-03	1.56E-03	3.01E-04	93.97
RE 2	SLS-k-ML(Ours)	10	**1.35E-08**	**3.16E-05**	**1.60E-06**	**4.66E-06**	**1.40**
	Coreset+sampling	10	2.89E-03	1.20E-01	2.16E-02	2.13E-02	3.34
	Coreset+exhaustive search	10	-	-	-	-	Over 12 hours
RE 2	SLS-k-ML(Ours)	3	**1.45E-05**	**5.59E-03**	**1.39E-03**	**1.18E-03**	**0.43**
	Coreset+sampling	3	4.88E-02	6.11E+00	6.21E-01	1.08E+00	4.91
	Coreset+exhaustive search	3	4.69E-04	4.83E-03	2.51E-03	1.09E-03	2.59E+02
SYN 1	SLS-k-ML(Ours)	10	**1.84E+04**	**1.88E+04**	**1.86E+04**	**1.97E+02**	**4.56E+02**
	Coreset+sampling	10	3.97E+04	4.51E+04	4.24E+04	2.70E+03	2.04E+04
	Coreset+exhaustive search	10					Over 12 hours
SYN 2	SLS-k-ML(Ours)	3	**1.87E+04**	**1.98E+04**	**1.98E+04**	**5.45E+02**	**1.15E+03**
	Coreset+sampling	3	-	-	-	-	Over 12 hours
	Coreset+exhaustive search	3	-	-	-	-	Over 12 hours

algorithms 10 times and report the minimum cost (*Min_Cost*), maximum cost (*Max_Cost*), average cost (*Avg_Cost*), standard deviation (*Std*), and runtime (*Time(s)*).

Algorithms. In our experiments, we give comparisons between our local search algorithm and the coreset-based method from [20]. For coreset-based method, we compress the data with their coreset algorithm and select centers via sampling or exhaustive search. For our SLS-k-ML algorithm, we design a sampling strategy that selects a subset of 100 points from r-*CrossLine* to improve computational efficiency. Following the settings in [14], we set the number of sampling rounds to $T = 400$, and the number of clusters to $k = 3$ and $k = 10$.

Results. Table 1 shows that our proposed SLS-k-ML algorithm always achieve the best performance compared with coreset+sampling algorithm and coreset+exhaustive algorithm. In particular, on datasets containing more than 5000 input lines, our algorithm runs at least 43 times faster while maintaining or improving clustering quality. Furthermore, the consistently low standard deviation observed across all experiments demonstrates the robustness of our algorithm.

5 Conclusions

In this paper, we propose the first local search algorithm for the k-means of lines problem, based on a single-swap strategy, which achieves a $(500 + \varepsilon)$-approximation guarantee and runs in polynomial time for low-dimensional Euclidean space. To handle the lack of triangle inequality and the geometric complexity of line clustering, we design two core components: a *proportional capture relation* for aligning optimal and current centers, and a *CrossLine* structure for discretizing line interactions. Extensive experiments on both synthetic and real-world datasets demonstrate that our algorithm consistently outperforms coreset-based baselines in both efficiency and clustering quality. However, our current theoretical analysis and experimental evaluation are mainly limited to low-dimensional settings, and alleviating the d-dependence will be an interesting direction for future work.

Acknowledgments and Disclosure of Funding

This work was supported by the National Natural Science Foundation of China (Nos. 62432016, 62172446) and the Central South University Research Program of Advanced Interdisciplinary Studies (No. 2023QYJC023). This work was also carried out in part using computing resources at the High Performance Computing Center of Central South University.

References

[1] Ankit Aggarwal, Amit Deshpande, and Ravi Kannan. Adaptive sampling for k-means clustering. In *Proceedings of the 12th International Workshop on Approximation Algorithms for Combinatorial Optimization*, pages 15–28, 2009.

[2] Sara Ahmadian, Ashkan Norouzi-Fard, Ola Svensson, and Justin Ward. Better guarantees for k-means and euclidean k-median by primal-dual algorithms. *SIAM Journal on Computing*, 49(4):17–97, 2019.

[3] David Arthur and Sergei Vassilvitskii. k-means++ the advantages of careful seeding. In *Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1027–1035, 2007.

[4] Davin Choo, Christoph Grunau, Julian Portmann, and Václav Rozhoň. k-means++ few more steps yield constant approximation. In *Proceedings of the 37th International Conference on Machine Learning*, pages 1909–1917, 2020.

[5] Vincent Cohen-Addad, Hossein Esfandiari, Vahab Mirrokni, and Shyam Narayanan. Improved approximations for euclidean k-means and k-median, via nested quasi-independent sets. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, pages 1621–1628, 2022.

[6] Vincent Cohen-Addad, Philip N Klein, and Claire Mathieu. Local search yields approximation schemes for k-means and k-median in euclidean and minor-free metrics. *SIAM Journal on Computing*, 48(2):644–667, 2019.

[7] Zachary Friggstad, Mohsen Rezapour, and Mohammad R Salavatipour. Local search yields a PTAS for k-means in doubling metrics. *SIAM Journal on Computing*, 48(2):452–480, 2019.

[8] Jie Gao, Michael Langberg, and Leonard J Schulman. Analysis of incomplete data and an intrinsic-dimension helly theorem. *Discrete & Computational Geometry*, 40:537–560, 2008.

[9] Jie Gao, Michael Langberg, and Leonard J Schulman. Clustering lines in high-dimensional space: Classification of incomplete data. *ACM Transactions on Algorithms (TALG)*, 7(1):1–26, 2010.

[10] Fabrizio Grandoni, Rafail Ostrovsky, Yuval Rabani, Leonard J Schulman, and Rakesh Venkat. A refined approximation for euclidean k-means. *Information Processing Letters*, 176:106251, 2022.

[11] Pierre Hansen, Jack Brimberg, Dragan Urošević, and Nenad Mladenović. Solving large p-median clustering problems by primal–dual variable neighborhood search. *Data Mining and Knowledge Discovery*, 19:351–375, 2009.

[12] George TS Ho, WH Ip, Carman KM Lee, and WL Mou. Customer grouping for better resources allocation using GA based clustering technique. *Expert Systems with Applications*, 39(2):1979–1987, 2012.

[13] Junyu Huang, Qilong Feng, Ziyun Huang, Jinhui Xu, and Jianxin Wang. FLS: A new local search algorithm for k-means with smaller search space. In *Proceedings of the 31st International Joint Conference on Artificial Intelligence*, pages 3092–3098, 2022.

[14] Junyu Huang, Qilong Feng, Ziyun Huang, Jinhui Xu, and Jianxin Wang. Linear time algorithms for k-means with multi-swap local search. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, pages 45651–45680, 2023.

[15] Amit Kumar, Yogish Sabharwal, and Sandeep Sen. Linear-time approximation schemes for clustering problems in any dimensions. *Journal of the ACM*, 57(2):1–32, 2010.

[16] Silvio Lattanzi and Christian Sohler. A better k-means++ algorithm via local search. In *Proceedings of the 36th International Conference on Machine Learning*, pages 3662–3671, 2019.

[17] Stuart Lloyd. Least squares quantization in PCM. *IEEE transactions on information theory*, 28(2):129–137, 1982.

[18] Sagi Lotan, Ernesto Evgeniy Sanches Shayda, and Dan Feldman. Coreset for line-sets clustering. *Proceedings of the 35th International Conference on Neural Information Processing Systems*, 35:37363–37375, 2022.

[19] Konstantin Makarychev, Aravind Reddy, and Liren Shan. Improved guarantees for k-means++ and k-means++ parallel. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, pages 16142–16152, 2020.

[20] Yair Marom and Dan Feldman. k-means clustering of lines for big data. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, pages 12817–12826, 2019.

[21] Jiří Matoušek. On approximate geometric k-clustering. *Discrete & Computational Geometry*, 24(1):61–84, 2000.

[22] Björn Ommer and Jitendra Malik. Multi-scale object detection by clustering lines. In *Proceedings of the 12th International Conference on Computer Vision*, pages 484–491, 2009.

[23] Wei Pan, Xiaotong Shen, and Binghui Liu. Cluster analysis: unsupervised learning via supervised learning with a non-convex penalty. *The Journal of Machine Learning Research*, 14(1):1865–1889, 2013.

[24] Tomer Perets. *Clustering of lines*. Open University of Israel Ra'anana, Israel, 2011.

[25] R Rollet, GB Benie, W Li, S Wang, and JM Boucher. Image classification algorithm based on the rbf neural network and k-means. *International Journal of Remote Sensing*, 19(15):3003–3009, 1998.

[26] Dennis Wei. A constant-factor bi-criteria approximation guarantee for k-means++. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pages 604–612, 2016.

A The Missing Proof

A.1 Proof of Lemma 2

Proof. Let $C^* = \{c_1^*, \ldots, c_k^*\}$ denote the optimal solution of $\mathcal{I}$. For any $\ell \in L$, let $c \in C$ and $c^* \in C^*$ denote the closest centers to ℓ in C and C^*, respectively. Then, we have

$$\begin{aligned}
\Delta(L, C) = \sum_{\ell \in L} \delta(\ell, c) &\leq \sum_{\ell \in L} \left(\sqrt{\delta(\ell, c^*)} + \sqrt{\delta(\pi_\ell(c), \delta(\pi_\ell(c^*))} + \sqrt{\delta(c, c^*)} \right)^2 \\
&\leq \sum_{\ell \in L} \left(\sqrt{\delta(\ell, c^*)} + 2\sqrt{\delta(c, c^*)} \right)^2 \\
&\leq \sum_{\ell \in L} \delta(\ell, c^*) + 4\delta(c, c^*) + 4\sqrt{\delta(\ell, c^*)}\sqrt{\delta(c, c^*)} \\
&\leq \sum_{\ell \in L} 3\delta(\ell, c^*) + 6\delta(c, c^*),
\end{aligned}$$

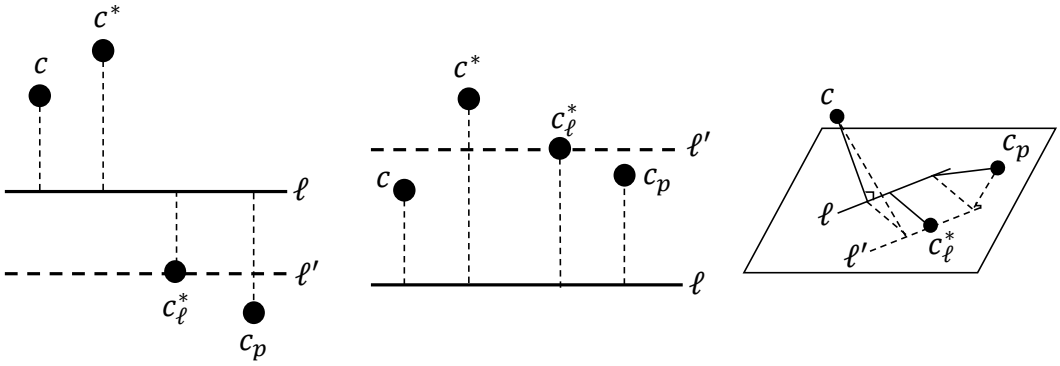
where the first inequality follows from triangle inequality, the second step follow from the fact that the distance between the projections of two points onto a line is at most their original Euclidean distance, and the last step follows the squared inequality and Cauchy inequality, respectively. Based on the Theorem in [24], the optimal solution to the k-means of lines problem can be obtained by selecting centers from the set of all pairwise line intersections (in $\mathbb{R}^2$). Since the maximum pairwise distance between any two points in the candidate set P can be explicitly computed, we have $\Delta(L, C) \leq \rho \cdot \Delta(L, C^*)$, where ρ is a constant. Moreover, the initial solution can be obtained in $O(n^2 d^2)$ time, as constructing the candidate set P takes $O(n^2 d^2)$ time and sampling k centers from P takes $O(1)$ time. □

A.2 Proof of Lemma 4

Proof. We first consider a type-1 match swap pair $(c, \psi_S(c))$ ($c \in C_S$), as the case of type-2 match swap pair is nearly identical and even simpler. Let $c^* = \psi_S(c)$. By definition 2, we have $\eta_1(c) = \Delta(L \backslash L(c^*), C \backslash \{c\}) - \Delta(L \backslash L(c^*), C)$, since lines in clusters other than $L(c)$ will still be assigned to their current center. Consider a line $\ell \in L(c) \backslash L(c^*)$, and it is easy to get that ℓ does not belong to the optimal cluster $L(c^*)$. Let c_ℓ^* ($c_\ell^* \neq c^*$) denote the closet optimal center in C^* from ℓ. Then, we construct a line ℓ' that is the translation of ℓ to c_ℓ^*, and then find a point c_p that is the closet center in C to ℓ'. It is easy to get that $c_p \neq c$, otherwise ℓ will remain assigned to its original center and no reassignment will be necessary. We now assign every lines in $L(c) \backslash L(c^*)$ to c_p, and get an estimate for the cost of reassigning these lines. Figure 1(b)-(c) shows two configurations of ℓ, c, c_ℓ^*, and c_p. From Figure 1(b), we have

$$\begin{aligned}
\eta_1(c) \leq \sum_{\ell \in L(c) \backslash L(c^*)} (\delta(\ell, c_p) - \delta(\ell, c)) &\leq \sum_{\ell \in L(c) \backslash L(c^*)} \left(\sqrt{\delta(\ell, c_\ell^*)} + \sqrt{\delta(\ell', c_p)} \right)^2 - \delta(\ell, c) \\
&\leq \sum_{\ell \in L(c) \backslash L(c^*)} \left(2\sqrt{\delta(\ell, c_\ell^*)} + \sqrt{\delta(\ell, c)} \right)^2 - \delta(\ell, c) \\
&\leq \sum_{\ell \in L(c) \backslash L(c^*)} 4\delta(\ell, c_\ell^*) + 2\sqrt{\frac{2}{\lambda}} \cdot \sqrt{2\lambda \cdot \delta(\ell, c_\ell^*) \cdot \delta(\ell, c)} \\
&\leq \sum_{\ell \in L(c) \backslash L(c^*)} (4 + \frac{2}{\lambda}) \cdot \delta(\ell, c_\ell^*) + 2\lambda \cdot \delta(\ell, c),
\end{aligned}$$

where the first inequality follows from equidistance property of parallel lines and squared inequality, the second step follow from the fact that c_p is the closet center in C to ℓ', and the last step follows



(a) (b) (c)

Figure 1: Illustrations of Line Configurations in $\mathbb{R}^2$ and $\mathbb{R}^d$.

from the Cauchy Inequality, respectively. From Figure 1(c), we have

$$\begin{aligned}\eta_1(c) \leq \sum_{\ell \in L(c) \setminus L(c^*)} (\delta(\ell, c_p) - \delta(\ell, c)) &\leq \sum_{\ell \in L(c) \setminus L(c^*)} \left(2\sqrt{\delta(\ell, c^*_\ell)} + \sqrt{\delta(\ell, c)}\right)^2 - \delta(\ell, c) \\ &\leq \sum_{\ell \in L(c) \setminus L(c^*)} 4\delta(\ell, c^*_\ell) + 2\sqrt{\frac{2}{\lambda}} \cdot \sqrt{2\lambda \cdot \delta(\ell, c^*_\ell) \cdot \delta(\ell, c)} \\ &\leq \sum_{\ell \in L(c) \setminus L(c^*)} (4 + \frac{2}{\lambda}) \cdot \delta(\ell, c^*_\ell) + 2\lambda \cdot \delta(\ell, c),\end{aligned}$$

where the first inequality follows the fact that $\delta(\ell, c_p) < \delta(\ell, c^*_\ell)$ under the configuration of Figure 1(c), the second steps and the last step follows from squared inequality and Cauchy Inequality. For the configurations of Figure 1(b)-(c), let $\lambda = \frac{1}{10}$. Then $\eta_1(c) \leq \frac{1}{5}\Delta(L(c), C) + 24\Delta(L(c), C^*)$. The cases for type-2 match swap pair $(c, \psi_N(c))$ $(c \in C_N)$ are similar to the cases for type-1 match swap pair. □

A.3 Proof of Lemma 5

Proof. By Definition 3 and Lemma 4, we have

$$\sum_{c \in C_S \cap C_S^b} \Delta(L(\phi_S(c)), C) \leq \sum_{c \in C_S} \eta_1(c) + 9 \cdot \tau + \frac{1}{100} \cdot \Delta(L, C) \leq \frac{21}{100} \cdot \Delta(L, C) + 33 \cdot \tau.$$

Based on the assumption that $\Delta(L, C) \geq 500 \cdot \tau$, we obtain $\sum_{c \in C_S \cap C_S^b} \Delta(L(\phi_S(c)), C) \leq \frac{69}{250} \cdot \Delta(L, C)$, and thus $\sum_{c \in C_S \cap C_S^b} \Delta(L(\phi_S(c)), C) \geq \frac{3}{50} \cdot \Delta(L, C)$. □

A.4 Proof of Lemma 6

Proof. For any point $c \in \mathbb{R}^2$ and line $\ell \in Q$, let $\pi_\ell(c)$ denote the projection of c on ℓ. let c be a center in C. We have

$$\begin{aligned}
\Delta(L, \{c\}) \leq \sum_{\ell \in L} \delta(\ell, c) \leq & \sum_{\ell \in L} \left(\sqrt{\delta(\ell, c^*)} + \sqrt{\delta(\pi_\ell(c), \pi_\ell(c^*))} + \sqrt{\delta(c, c^*)} \right)^2 \\
\leq & \sum_{\ell \in L} \left(\sqrt{\delta(\ell, c^*)} + 2\sqrt{\delta(c, c^*)} \right)^2 \\
\leq & \sum_{\ell \in L} \delta(\ell, c^*) + 4\delta(c, c^*) + 4\sqrt{\delta(\ell, c^*)}\sqrt{\delta(c, c^*)} \\
\leq & \sum_{\ell \in L} 3\delta(\ell, c^*) + 6\delta(c, c^*) \\
\leq & 3\Delta(L, \{c^*\}) + 6|L|\delta(c, c^*)
\end{aligned}$$

where the first inequality follows from the triangle inequality, the second step follow from the fact that the distance between the projections of two points onto a line is at most their original Euclidean distance, respectively. □

A.5 Proof of Lemma 7

Proof. Based on the Lemma 6, we know that the closest center in C to c^* has distance at least $\frac{\alpha-3}{6|Q|} \cdot \Delta(Q, \{c^*\})$ as otherwise $\Delta(Q, C) < \alpha \cdot \Delta(Q, \{c^*\})$. Hence, the distance of every lines in R to C is at least

$$\left(\sqrt{2} - 2\sqrt{\frac{\alpha - 3}{6}} \right)^2 \cdot \frac{\text{cost}(Q, \{c^*\})}{|Q|} \geq \frac{\alpha - 1}{24} \cdot \frac{\text{cost}(Q, \{c^*\})}{|Q|},$$

where we use that $\alpha \geq 9$. Furthermore, by averaging we get $|R| \geq |Q|/2$, which together with the inequality above implies the result. □

A.6 Proof of Lemma 8

Proof. The 9 grid points defined in $\text{CrossLine}_r(\ell_1, \ell_2)$ form a regular 3×3 grid in a parallelogram arrangement. The lines in $\mathcal{L}_1$ are equally spaced at distance r, and so are the lines in $\mathcal{L}_2$. Thus, the grid divides the local region around the intersection point into parallelogram cells, each defined by two adjacent lines from $\mathcal{L}_1$ and two from $\mathcal{L}_2$. Any point x within $\text{CrossLine}_r(\ell_1, \ell_2)$ must lie in one of these parallelogram cells. Since each parallelogram has diameter (i.e., maximal corner-to-center distance) less than or equal to r, any point x must be within distance r of at least one vertex of the cell, which is a grid point of $\text{CrossLine}_r(\ell_1, \ell_2)$. Hence, the lemma follows. □

A.7 Proof of Lemma 9

Proof. Note that $|C \backslash C_S| \leq 2|C_N|$. By the Definition 5 and Lemma 4, we have

$$\begin{aligned}
\sum_{c \in C \backslash C_S, c \in C_N^b} \Delta(L(\psi_N(c)), C) \leq & 2|C_N| \min_{c \in C_N} \eta_2(c) + 9\tau + \frac{1}{100}\Delta(L, C) \\
\leq & 2 \sum_{c \in C_N} \eta_2(c) + 9\tau + \frac{1}{100}\Delta(L, C) \\
\leq & \frac{41}{100}\Delta(L, C) + 57\tau.
\end{aligned}$$

Based on the assumption that $\Delta(L, C) \geq 500 \cdot \tau$, we obtain $\sum_{c \in C \backslash C_S, c \in C_N^b} \Delta(L(\psi_N(c)), C) \leq \frac{131}{250} \cdot \Delta(L, C)$, and thus $\sum_{c \in C \backslash C_S, c \in C_N^g} \Delta(L(\phi_N(c)), C) \geq \frac{7}{50} \cdot \Delta(L, C)$. □

A.8 Proof of Lemma 10

Proof. Similar to $\mathbb{R}^d$, we also construct a line ℓ' that is the translation of ℓ to c^*_ℓ, and then find a point c_p that is the closet center in C to ℓ'. We now assign every lines in $L(c)\backslash L(c^*)$ to c_p, and get an estimate for the cost of reassigning these lines. From Figure 1(c), we have

$$\begin{aligned}
\eta_1(c) \leq \sum_{\ell \in L(c)\backslash L(c^*)} (\delta(\ell, c_p) - \delta(\ell, c)) &\leq \sum_{\ell \in L(c)\backslash L(c^*)} \left(\sqrt{\delta(\ell, c^*_\ell)} + \sqrt{\delta(\ell', c_p)}\right)^2 - \delta(\ell, c) \\
&\leq \sum_{\ell \in L(c)\backslash L(c^*)} \left(2\sqrt{\delta(\ell, c^*_\ell)} + \sqrt{\delta(\ell, c)}\right)^2 - \delta(\ell, c) \\
&\leq \sum_{\ell \in L(c)\backslash L(c^*)} 4\delta(\ell, c^*_\ell) + 2\sqrt{\frac{2}{\lambda}} \cdot \sqrt{2\lambda \cdot \delta(\ell, c^*_\ell) \cdot \delta(\ell, c)} \\
&\leq \sum_{\ell \in L(c)\backslash L(c^*)} (4 + \frac{2}{\lambda}) \cdot \delta(\ell, c^*_\ell) + 2\lambda \cdot \delta(\ell, c),
\end{aligned}$$

where the first inequality follows from equidistance property of parallel lines and triangle inequality, the second step follow from the fact that c_p is the closet center in C to ℓ', and the last step follows from the Cauchy Inequality, respectively. For the configuration of Figure 1(c), let $\lambda = \frac{1}{10}$. Then $\eta_1(c) \leq \frac{1}{5}\Delta(L(c), C) + 24\Delta(L(c), C^*)$. The cases for type-2 match swap pair $(c, \psi_N(c))$ $(c \in C_N)$ are similar to the cases for type-1 match swap pair. The lemma follows. □

A.9 Proof of Lemma 11

Proof. By construction, each $\mathcal{L}_i$ contains 3^d lines, generated by shifting ℓ_i along all combinations of $\{-1, 0, 1\}^d$ at step size r along coordinate axes. This induces a discrete grid of intersection points, where each point lies at the intersection of one line from $\mathcal{L}_1$ and one from $\mathcal{L}_2$. The resulting grid forms a regular structure embedded within the union of two axis-aligned hypercubes of side length $2r$, centered near a common point x^*. Since each axis-aligned hypercube is of side length $2r$, the distance between adjacent grid points along any axis is at most r, and each hypercube cell (in which grid points are located at vertices) has diameter at most $\sqrt{d} \cdot r$. Then, for any point x lying within the convex hull of the grid points (i.e., the r-CrossLine region), there exists a hypercube cell that contains x. By geometry of axis-aligned cubes, any point within such a cell lies within distance r from at least one of its 2^d corners, all of which are grid points. Hence, the lemma follows. □

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The main claims made in the abstract and introduction accurately reflect the contributions of the paper, which proposes the first single-swap local search algorithm for the k-means of lines problem, leveraging a novel proportional capture relation and CrossLine structure to achieve a $(500 + \varepsilon)$-approximation in polynomial time. see Section 1.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes] .

   Justification: The limitations are discussed in the last section of this paper.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: All theoretical results are stated with full assumptions, and complete proofs are provided in the main paper and supplementary material (see Section 3 and appendix).

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The paper provides sufficient experimental details to allow reproduction of the main results and validation of the conclusions.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: The codes (including dataset generation and experimental code) are available upon request via email.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: The experimental settings have been clearly stated in the paper.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: In the experiment comparison results, each algorithm is executed 10 times on each dataset, and the average results are reported as the final results.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The paper provides sufficient details about the computational resources used.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We have reviewed the NeurIPS Code of Ethics and confirm that our research complies fully. The paper does not involve human subjects, sensitive data, or applications with known dual-use or misuse risks.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: This paper mainly focuses on the theoretical aspect of the k-means of lines problem. The primary purpose is to provide algorithmic insights, and we do not foresee any specific societal consequences related to the proposed method.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.

- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: This paper poses no such risks about safeguards.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [NA]

    Justification: This paper does not use any external assets such as datasets, models, or third-party code. Therefore, licensing considerations are not applicable.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: This paper does not introduce or release any new assets such as datasets, models, or software code. Therefore, documentation for new assets is not applicable.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: This paper does not involve any crowdsourcing experiments or research with human subjects. Therefore, this question is not applicable.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: This paper does not involve any research with human subjects, and therefore IRB or equivalent ethical approval is not applicable.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: This paper does not involve the use of large language models (LLMs) as part of the core research methodology. Therefore, no declaration is necessary.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.