Simple KNN-Based Outlier Detection Achieves Robust Clustering

Tianle Jiang, Yufa Zhou
Duke University
{tianle.jiang,yufa.zhou}@duke.edu

Abstract

Being robust to the presence of outliers is crucial for applying clustering algorithms in practice. In the *robust k-Means* problem (i.e., k-Means with outliers), the goal is to remove z outliers and minimize the k-Means cost on the remaining points. Despite the close connection between robust k-Means and outlier detection, both theoretical and empirical understanding of the effectiveness of *classic outlier detection heuristics* for robust k-Means remains limited. In this paper, we prove that under a practical assumption on the optimal cluster sizes, simply removing points with large K-Nearest-Neighbor distances achieves performance comparable to prior work in terms of approximation guarantees: it yields a constant-factor reduction from robust k-Means to standard k-Means, without introducing additional centers or discarding extra outliers, as is commonly required by existing approaches. Empirically, experiments on real-world datasets show that our method outperforms or matches several more sophisticated algorithms in terms of clustering cost and runtime[1]. These results demonstrate that simple KNN-based heuristics can be surprisingly effective for robust clustering, highlighting new opportunities to bridge techniques from outlier detection and clustering.

1 Introduction

Clustering is a fundamental problem in unsupervised learning and has been studied extensively in machine learning, data mining, and theoretical computer science. However, it is well known that widely used clustering objectives such as k-Median and k-Means are highly sensitive to noise and outliers [10, 12]. In particular, even a single data point that lies far from the underlying clusters can significantly distort the solution and lead standard clustering algorithms to select poor centers. Since noise and corrupted points are ubiquitous in real-world datasets, the design of efficient and practical clustering algorithms that are robust to outliers has attracted sustained attention [12, 27, 29, 30].

We study the robust k-Clustering problem (also known as k-Clustering with Outliers), first formalized by Charikar et al. [10]. In this problem, the algorithm is given an outlier budget z and aims to minimize a standard clustering objective, such as k-Median or k-Means, after removing at most z points from the dataset. Although polynomial-time constant-factor approximation algorithms are known for both robust k-Median and robust k-Means [13, 26, 37], recent research has increasingly emphasized scalability. Specifically, there is a strong interest in finding robust k-Means algorithms that run in near-linear time in the number of data points n.

Existing efficient algorithms for robust k-Means can be broadly categorized into two classes: (1) *Outlier-removal-based methods* [12, 24, 27, 30] explicitly identify and discard suspected outliers, and then apply a standard clustering algorithm to the remaining points. These approaches typically have super-linear runtime and, in practice, rely on coreset constructions to scale to large datasets.

[1] Code: https://github.com/MasterZhou1/Robust-Clustering

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

(2) *Sampling-based methods* [6, 17, 24, 29] avoid explicit outlier removal and instead use adaptive sampling strategies to select centers while reducing the probability of sampling outliers.

To achieve a favorable trade-off between efficiency and solution quality, algorithms in both categories generally resort to bicriteria approximations. That is, they allow the use of more than k centers, the removal of more than z outliers, or both. For instance, one of the most efficient recent algorithms by Huang et al. [29] runs in $O(\epsilon^{-2}ndk \log\log n)$ time and achieves a 4-approximation, but uses $O(k/\epsilon)$ centers and discards $(1+\epsilon)z$ outliers. We refer readers to Huang et al. [29] for a comprehensive comparison of runtime and approximation guarantees.

Outlier detection (or anomaly detection) is a closely related but distinct problem, where the objective is to identify anomalous points rather than to optimize a clustering cost. A classic and effective approach in this literature ranks data points by their distance to the K-th nearest neighbor [45], which we name as the *KNN heuristic*. Owing to its simplicity and strong empirical performance, this heuristic has become a widely used baseline across many application domains [8, 40, 51]. Despite the conceptual connection between outlier detection and robust clustering, to the best of our knowledge, there has been no theoretical or systematic empirical study of applying the KNN heuristic as an outlier-removal step for the robust k-Means problem. While more sophisticated outlier detection techniques have been considered in Huang et al. [29], they typically result in substantially worse clustering costs compared to algorithms specifically designed for robust k-Means.

1.1 Our Contributions

In this paper, we investigate whether the KNN heuristic for outlier removal can achieve rigorous theoretical approximation guarantees and empirical practicality for the robust k-Means problem, which, surprisingly, is an overlooked direction in the robust clustering literature.

Theoretical Results. Under the assumption that every optimal cluster contains at least cz points for some constant $c > 1$, we show that the vanilla KNN heuristic with $K = \frac{c+1}{2}z$ yields a constant-factor reduction to standard k-Means.

Theorem 1.1 (Approximation guarantee of vanilla KNN, Theorem 4.1)**.** *If each cluster in the optimal solution has size at least cz for $c > 1$, then using the vanilla KNN heuristic to remove z outliers gives a $\Phi(c)$-approximate reduction to standard k-Means, where $\Phi(\cdot)$ is a monotone decreasing function.* [2]

We note that several prior works [3, 29, 30] adopt the assumption on the input instance that the size of each cluster in the optimal solution is lower bounded by $3z$. This is a reasonable practical assumption, since the number of outliers is typically small compared to the number of inliers, and inliers are often well clustered. The datasets that are commonly used to evaluate robust clustering algorithms also have the property that the true clusters are larger than the number of outliers. Of course, there can be real-world datasets that contain underrepresented but valid groups of points, but from a theoretical perspective, without any knowledge to true labels, if some points form a very small cluster, it is hard to distinguish whether they are true inliers or a group of adversarial outliers.

Under this assumption, these works show that one can substantially reduce the violation in the number of centers used or the number of outliers removed. Table 1 compares our results with prior work under the same assumption.

We prove that under the same assumption, the KNN heuristic, despite its extreme simplicity, already achieves strong theoretical guarantees. In particular, when $c = 3$, applying the vanilla KNN heuristic with $K = 2z$ to identify and remove z outliers yields a $\Phi(3) = 9$-approximate reduction from robust k-Means to standard k-Means. We emphasize that, while the parameter K in the KNN heuristic is traditionally chosen to be a constant, our result requires K to scale linearly with z in order to obtain provable guarantees. Compared to Im et al. [30], whose method also gives a 9-approximate reduction but requires access to the optimal cost OPT and removes $2z$ outliers, and to Huang et al. [29], who achieves a 4-approximation but must use $O(k)$[3] centers, the KNN heuristic **strictly satisfies the constraints on both k and z**. Although the assumption of knowing OPT can be removed via geometric guessing, doing so introduces significant overhead in practice. Finally, the assumed lower bound of $3z$ in prior work is somewhat arbitrary, and they do not provide a justification for the

[2] The best approximation ratio we can obtain for each c depends on the solution to a complex equation and has no succinct closed-form representation.

[3] Although not explicitly stated in their paper, the hidden constant in $O(\cdot)$ is very large.

Table 1: Comparison with previous theoretical results. Here $|P^*|$ denotes the size of the smallest cluster in the optimal solution.

Algorithm	Approximation	Centers Used	Outliers Removed	Assumption	Source
NKMeans	9 (reduction)	k	$(3k+2)z$	Know OPT	Im et al. [30]
NKMeans	9 (reduction)	k	$2z$	$\|P^*\| \geq 3z$, Know OPT	Im et al. [30]
TIKMeans	4	$O(k/\epsilon)$	$(1+\epsilon)z$	No	Huang et al. [29]
TIKMeans	4	$O(k)$	z	$\|P^*\| \geq 3z$	Huang et al. [29]
OKMeans	9 (reduction)	k	z	$\|P^*\| \geq 3z$	Theorem 4.1
OKMeans2	5.98 (reduction)	k	z	$\|P^*\| \geq 3z$	Theorem 4.6

specific choice of 3. Our results extend to arbitrary $c > 1$, and agrees with the intuition that the approximation ratio improves as c increases, i.e., the inliers are more well-clustered, making outliers easier to identify.

Inspired by the proof of Theorem 1.1, we propose a slight variant of the vanilla KNN heuristic that aggregates distances to multiple neighbors, rather than relying on a single neighbor, as the outlier score. This variant can be analyzed using a similar proof strategy and yields a better approximation guarantee in Theorem 1.2, further demonstrating the versatility and potential of KNN-based outlier detection heuristics for the robust clustering problem. When each optimal cluster has size at least $3z$, the approximation ratio **improves from** 9 **to** 5.98.

Theorem 1.2 (Approximation guarantee of KNN variant, Theorem 4.6)**.** *If each cluster in the optimal solution has size at least cz for $c > 1$, then using a variant of the KNN heuristic to remove z outliers gives a $\Psi(c)$-approximate reduction to standard k-Means, where $\Psi(\cdot)$ is a monotone decreasing function.*

Empirical Results. We evaluate our proposed Algorithms 1a and 1b on four real-world datasets ranging from small to large scale (up to 5 million points), comparing them against six baselines. Our experiments demonstrate that: (1) **Clustering Quality**: Our methods achieve competitive clustering costs and often match or outperform strong robust clustering baselines, especially on large-scale datasets such as SUSY and KDDFULL; notably, on SUSY-10, OKMeans improves upon the previous SOTA method, TIKMeans. (2) **Scalability**: By leveraging coresets and efficient nearest-neighbor search implementations, our approach scales effectively to millions of data points without sacrificing solution quality, validating that simple KNN heuristics can outperform complex sampling-based algorithms in practice. (3) **Parameter Sensitivity**: Empirically, the performance of our methods is not particularly sensitive to the choice of the parameter c. However, on some datasets, choosing K as a small fixed constant in the KNN heuristic can lead to noticeably worse performance. Although choosing K as a small constant is common practice when applying KNN to outlier detection, our result suggests that this choice may be suboptimal for the robust clustering objective.

2 Related Work

2.1 (Robust) Clustering

Two of the most classical clustering objectives are k-Median and k-Means. Both problems are NP-hard, but there exist constant-factor approximation algorithms for general metrics [9, 11, 16, 23, 34], and improved approximation ratios [2, 15] and even polynomial-time approximation schemes (PTAS) [4, 42] in Euclidean space.

The class of robust clustering problems was first proposed by Charikar et al. [10]. For robust k-Median, they gave a $4(1+1/\gamma)$-approximation algorithm when the algorithm is allowed to remove $(1+\gamma)z$ outliers. In subsequent work, the only known polynomial-time, non-bicriteria, constant-factor approximation algorithms [13, 26, 37] are based on linear programming relaxations and iterative rounding, but their runtimes are high-order polynomials in n. The best approximation ratios known so far are a $(6.994+\epsilon)$-approximation for robust k-Median [26] and a $(53.002+\epsilon)$-approximation for robust k-Means [37], both with running time $n^{O(\mathrm{poly}(1/\epsilon))}$. Another recent line of research direction studies fixed-parameter tractable (FPT) approximation algorithms [1, 14, 20, 21, 31], where k, z, or both are treated as parameters. However, the running times of these algorithms remain impractical even for moderate values of k and z.

2.2 Outlier Detection and KNN

Nearest-neighbor–based methods form a classical paradigm for outlier detection. The idea of identifying outliers based on neighborhood distances first appeared in Knorr and Ng [35], and the KNN heuristic based on a global ranking was later formalized by Ramaswamy et al. [45]. Subsequently, numerous variants of the original KNN heuristic have been proposed [19, 36, 43, 48, 50], which either make more extensive use of K-nearest-neighbor information or extend KNN beyond purely numerical attributes. There has also been substantial effort devoted to optimizing the runtime of computing or approximating KNN distances [19, 22, 32, 33, 38, 41, 46, 47, 49]. More recently, KNN-based retrieval has been successfully incorporated into retrieval-augmented generation (RAG) systems [7, 25, 39] to enhance the performance of large language models.

3 Preliminaries

3.1 Problem Definition

Standard k-Clustering. Let $[n] := \{1, 2, 3, \ldots, n\}$ for $n \in \mathbb{N}^+$. In the standard k-clustering problem, the input consists of a set X of n points in a metric space and an integer parameter $k \in [n]$. The goal is to compute a set C of k centers, then each input point is assigned to its closest center. The quality of a clustering solution is measured by an objective function that depends on the distances between points and their assigned centers. Let $d(\cdot, \cdot)$ denote the distance function of the metric space, and define $d(x, C) = \min_{c \in C} d(x, c)$. The three most common k-clustering objectives are: (1) k**-Median:** $\mathrm{cost}(X, C) = \sum_{x \in X} d(x, C)$; (2) k**-Means:** $\mathrm{cost}(X, C) = \sum_{x \in X} d(x, C)^2$; (3) k**-Center:** $\mathrm{cost}(X, C) = \max_{x \in X} d(x, C)$.

In this paper, we focus on the k-Means objective. We will briefly discuss how our algorithms can be extended to k-Median and k-Center in Appendix B. Note that there are different variants of the k-clustering problem regarding what points can be chosen as centers. The most common setting allows any point in the metric space to be chosen as a center, while there are also variants that require C to be a subset of X, or a subset of a given set of candidate centers. The main theoretical result in this paper is indifferent among these settings.

Clustering with Outliers. In the robust k-Means problem, an instance $I = (X, k, z)$ consists of a dataset X of n points in a metric space and parameters $k \in [n]$ and $z \in [n]$. The goal is to select a subset $Z \subseteq X$ of z outliers such that the k-Means cost of the remaining points is minimized, formally $\min_{Z \subseteq X,\ |Z|=z} \min_{C:|C|=k} \sum_{x \in X \setminus Z} d(x, C)^2$. Equivalently, the problem can be viewed as selecting a set C of k centers, such that after removing the z points in X with the largest $d(x, C)$ as outliers, the k-Means cost of the remaining points is minimized.

In practice, the appropriate value of z may be unknown a priori, and its choice depends on the specific application and domain knowledge. Specifying this budget and optimizing the resulting clustering objective are separate questions. Following the standard formulation [10, 12, 29, 30], this paper studies the approximation guarantees and computational efficiency of the latter, treating z as a prescribed input.

3.2 Notations

Robust k-Means Objective. We focus on the robust k-Means problem. For a dataset X, let $f(X) := \min_{C:|C|=k} f(X, C)$ denote the optimal k-Means cost over all choices of k centers, where $f(X, C)$ is the k-Means cost induced by a fixed center set C on X. Given an outlier budget z, we similarly define $f_z(X) := \min_{C:|C|=k} f_z(X, C)$ as the optimal robust k-Means cost, where $f_z(X, C) := \min_{Z \subseteq X, |Z|=z} \sum_{x \in X \setminus Z} d(x, C)^2$ denotes the robust k-Means cost achieved by a fixed set of centers C on X. We use g in place of f when referring to analogous definitions for the k-Median objective.

Approximation Guarantee. An algorithm is said to achieve an α-approximation if, for any instance $I = (X, k, z)$, it outputs a set C of k centers such that $f_z(X, C) \leq \alpha \cdot f_z(X)$. An algorithm is said to provide an α-approximate reduction from robust k-Means to standard k-Means, if for any instance $I = (X, k, z)$, outputs a set of points $X' \subseteq X$ such that $|X'| = |X| - z$ and $f(X') \leq \alpha \cdot f_z(X)$.

Algorithm 1 Two KNN-based outlier detection algorithms for robust k-means. X is the input point set, k is the number of clusters, z is the outlier budget, $A(\cdot, k)$ is a standard k-means algorithm, and $c > 1$ is the constant in the cluster size assumption.

(a) OKMeans(X, k, z)

for each $x \in X$ **do**
  $r_x \leftarrow d(x, \mathrm{NN}_X(x, \frac{c+1}{2}z))$ [4]
  ▷ radius to the $2z$-th neighbor
end for
Let $O \subseteq X$ be the z points with largest r_x
return $A(X \setminus O, k)$

(b) OKMeans2(X, k, z)

for each $x \in X$ **do**
  $s_x \leftarrow \sum_{i=z+1}^{cz} d(x, \mathrm{NN}_X(x, i))$
  ▷ aggregate distance to mid-range neighbors
end for
Let $O \subseteq X$ be the z points with largest s_x
return $A(X \setminus O, k)$

Other Useful Notations. Let X be a dataset and let C be a set of k centers. Given a prescribed number of outliers z, let $Z(C)$ denote the set of outliers induced by C, which consists of the z points in X with the largest distances to their closest centers in C. The corresponding set of inliers is $X_z(C) := X \setminus Z(C)$. For any point x and radius $r > 0$, let $B_X(x, r) := \{y \in X \mid d(x, y) \le r\}$ denote the set of points in X that are within distance r from x, and let $B^\circ_X(x, r) := \{y \in X \mid d(x, y) < r\}$. Let $\mathrm{NN}_X(x, K)$ denote the K-th nearest-neighbor of x among X. If multiple neighbors are at the same distance to x, the ranking breaks ties arbitrarily. For any inlier $x \in X_z(C)$, we define its closest center as $\mathrm{closest}(x, C) := \arg\min_{c \in C} d(x, c)$, breaking ties arbitrarily. The (robust) cluster of x induced by C is then defined as $\mathrm{cluster}_z(x, C) := \{y \in X_z(C) \mid \mathrm{closest}(y, C) = \mathrm{closest}(x, C)\}$.

4 Main Results

In this section, we present our main theoretical contribution: the classic and simple idea of identifying outliers based on nearest-neighbor distance gives a constant-factor reduction from robust k-Means to standard k-Means, under the practical assumption that the optimal clusters are large enough. We first analyze the vanilla version of the KNN heuristic, then propose and analyze a slight variant inspired by the first analysis, which gives a better approximation ratio.

4.1 Analysis of Vanilla KNN

Theorem 4.1 (Restatement of Theorem 1.1)**.** *Let $I = (X, k, z)$ be an instance of robust k-Means and C^* be the optimal set of centers. Suppose $\mathcal{A}$ is a β-approximation algorithm for k-Means, and each cluster induced by C^* has size at least cz, then there is an algorithm that calls $\mathcal{A}$ once and computes a set of centers C such that $f_z(X, C) \le \Phi(c)\beta \cdot f_z(X, C^*)$, where $\Phi(c) = \frac{(1+\sqrt{(c-1)/2})^2}{(c-1)/2}(x^*)^2$ and x^* is the unique real root greater than 1 of the following equation:*

$$\left[\left(1 + \sqrt{\frac{c-1}{2}}\right)^2 x^2 - \frac{c-1}{2}\right](x-1)^2 - 2x + 1 = 0\,.$$

Here we only present a proof for the special case of $c = 3$ (where $\Phi(3) = 9$), which aligns with the assumptions from prior work [29, 30] and is numerically simpler. The proof for general c follows the same structure but requires careful tuning of the intermediate parameters used in the analysis and is deferred to Appendix A.1. The formal algorithm is presented in Algorithm 1a. In the special case of $c = 3$, for each point x, we simply compute the distance to its $2z$-th nearest-neighbor, and remove the z points with largest $2z$-NN distances.

We fix an optimal set of centers C^*, and call the points in $X_z(C^*)$ "true inliers" and the points in $X \setminus X_z(C^*)$ "true outliers". Define $Y = O \cap X_z(C^*)$ as the set of true inliers removed by the algorithm by mistake, and $Z' = (X \setminus O) \cap (X \setminus X_z(C^*))$ as the set of true outliers included by the algorithm by mistake, then we have $|Y| = |Z'|$. Below we assume that $|Y| \ge 1$, otherwise we have already identified all outliers correctly and the reduction is costless. Let $T = \min_{x \in O}\{r_x\}$, i.e., the threshold separating the r_x of $X \setminus O$ and O. We start by proving two lower bounds on the cost of the optimal solution.

[4] We assume without loss of generality that $\frac{c+1}{2}z$ is an integer. Our implementation uses $\lfloor \frac{c+1}{2}z \rfloor$, which has a negligible impact on the analysis.

Lemma 4.2. *Suppose each optimal cluster has size at least $3z$, and $\min_{y\in Y} d(y, C^*) = \lambda T$ for some $\lambda \geq 0$, then $f_z(X, C^*) \geq |Y|T^2 \cdot \max\{\lambda^2, (1-\lambda)^2\}$.*

Proof. Since every $y \in Y$ satisfies $d(y, C^*) \geq \lambda T$, we have

$$f_z(X, C^*) \geq f(Y, C^*) = \sum_{y\in Y} d(y, C^*)^2 \geq |Y|T^2 \cdot \lambda^2 \,.$$

If $\lambda \geq 1$ then $\lambda^2 \geq (1-\lambda)^2$ and we are done. Otherwise, fix a $y \in Y$ such that $d(y, C^*) = \lambda T$ for some $\lambda < 1$. Since $y \in O$, we have $r_y \geq T$, which implies $|B^\circ_X(y, T)| \leq 2z$. By the assumption that each optimal cluster has size at least $3z$, at least z points in $\mathrm{cluster}_z(y, C^*)$ are not in $B^\circ_X(y, T)$, and for each such point x, by the triangle inequality, we have

$$\begin{aligned} d(x, \mathrm{closest}(y, C^*)) &\geq d(x, y) - d(y, \mathrm{closest}(y, C^*)) \\ &\geq T - \lambda T = T(1-\lambda) \,, \end{aligned}$$

therefore,

$$\begin{aligned} f_z(X, C^*) &\geq f_z(\mathrm{cluster}_z(y, C^*) \setminus B^\circ_X(y, T), C^*) \\ &\geq zT^2 \cdot (1-\lambda)^2 \geq |Y|T^2 \cdot (1-\lambda)^2 \,. \end{aligned}$$

The last inequality follows from the fact that $|Y| \leq |O| = z$. □

Lemma 4.3. *Suppose each optimal cluster has size at least $3z$, and $\max_{o\in Z'} d(o, C^*) = \delta T$ for some $\delta > 1$, then $f_z(X, C^*) \geq |Y|T^2 \cdot (\delta-1)^2$.*

Proof. Let $o = \arg\max_{o'\in Z'}\{d(o', C^*)\}$, so $d(o, C^*) = \delta T$. Since $o \in X \setminus O$, we have $r_o \leq T$, which implies $|B_X(o, T)| \geq 2z$. Therefore, there must be at least z true inliers in $B_X(o, T)$. For each such inlier x, we must have $d(x, C^*) \geq (\delta-1)T$, otherwise there would be the following contradiction:

$$d(o, C^*) \leq d(o, x) + d(x, C^*) < T + (\delta-1)T = \delta T \,.$$

Thus, we must have

$$f_z(X, C^*) \geq \sum_{x\in B_X(o,T)\cap X_z(C^*)} d(x, C^*)^2 \geq zT^2 \cdot (\delta-1)^2 \geq |Y|T^2 \cdot (\delta-1)^2 \,. \qquad \square$$

Lemma 4.4. *Suppose each optimal cluster has size at least $3z$, then $f(X \setminus O, C^*) \leq 9 \cdot f_z(X, C^*)$.*

Proof. We distinguish between the following three cases:

Case 1 If $\max_{o\in Z'} d(o, C^*) \leq \frac{3}{2}T$, then by Lemma 4.2, assuming $\min_{y\in Y} d(y, C^*) = \lambda T$ for some $\lambda \geq 0$, then we have

$$\begin{aligned} f(X \setminus O, C^*) &\leq f_z(X, C^*) + f(Z', C^*) - f(Y, C^*) \\ &\leq f_z(X, C^*) + |Z'| \cdot (3T/2)^2 - |Y| \cdot (\lambda T)^2 \\ &\leq f_z(X, C^*) \cdot \left(1 + \frac{\left(\frac{3}{2}\right)^2 - \lambda^2}{\max\{\lambda^2, (1-\lambda)^2\}}\right) \,, \end{aligned}$$

which is maximized at $\lambda = \frac{1}{2}$ and takes value $9 \cdot f_z(X, C^*)$.

Case 2 If $\max_{o\in Z'} d(o, C^*) = \delta T$ for some $\delta > \frac{3}{2}$, and $d(y, C^*) \geq (\delta-1)T$ for all $y \in Y$, then by Lemma 4.2, we have

$$\begin{aligned} f(X \setminus O, C^*) &\leq f_z(X, C^*) + f(Z', C^*) - f(Y, C^*) \\ &\leq f_z(X, C^*) + |Z'| \cdot (\delta T)^2 - |Y| \cdot ((\delta-1)T)^2 \\ &\leq f_z(X, C^*) \cdot \left(1 + \frac{2\delta - 1}{(\delta-1)^2}\right) \,, \end{aligned}$$

which is at most $9 \cdot f_z(X, C^*)$ when $\delta > \frac{3}{2}$.

Case 3 If $\max_{o\in Z'} d(o, C^*) = \delta T$ for some $\delta > \frac{3}{2}$, and $\min_{y\in Y} d(y, C^*) = \lambda T$ for some $0 \le \lambda < \delta - 1$, then by Lemma 4.2 and Lemma 4.3, we have

$$\begin{aligned} f(X \setminus O, C^*) &\le f_z(X, C^*) + f(Z', C^*) - f(Y, C^*) \\ &\le f_z(X, C^*) + |Z'| \cdot (\delta T)^2 - |Y| \cdot (\lambda T)^2 \\ &\le f_z(X, C^*) \cdot \left(1 + \frac{\delta^2 - \lambda^2}{\max\{(1-\lambda)^2, (\delta-1)^2\}}\right) , \end{aligned}$$

which is at most $9 \cdot f_z(X, C^*)$ when $\delta > \frac{3}{2}$ and $\lambda < \delta - 1$. □

Finally, we apply the β-approximate k-Means algorithm on $X \setminus O$, and obtain a set of centers C such that $f(X \setminus O, C) \le 9\beta \cdot f_z(X, C^*)$. If we then recompute the set of inliers with respect to C, the clustering cost does not increase, therefore,

$$f_z(X, C) \le f(X \setminus O, C) \le 9\beta \cdot f_z(X, C^*) .$$

This concludes the proof of Theorem 4.1 for $c = 3$.

Remark 4.5 (Small optimal clusters)**.** *Let s be the total number of inliers in optimal clusters of size less than cz. Keeping the neighborhood parameters unchanged, remove the $z + s$ largest-scoring points, provided $z + s \le n - k$. The same approximation factor continues to hold: among the removed points, the number of inliers from large optimal clusters is at least the number of retained true outliers. We can therefore pair each retained outlier with a distinct removed inlier from a large cluster and apply the original charging argument. The additional removed inliers only decrease the retained clustering cost.*

4.2 A New Variant of KNN

Numerous variants that utilize more information about the K-neighborhood have emerged after the proposal of the original KNN heuristic. For example, instead of using the distance to the K-th nearest neighbor, Eskin et al. [19] used the sum of distances to the K nearest neighbors.

We propose another variant that can be analyzed using the same framework as Theorem 4.1 and admits better approximation ratios for all c. This variant is inspired by an attempt to analyze the aforementioned variant that identifies outliers based on the sum of distances to the K nearest neighbors. We observe that it is not robust to adversarially generated input: the true outliers can be very close to each other, gaining an advantage when comparing the sum of distances, which leads to an even worse approximation ratio. As a result, we propose to omit the distances to the nearest z neighbors, so that outliers cannot "help" each other achieve comparatively smaller total distances. We present the formal algorithm in Algorithm 1b: for each point x, we compute the total distance s_x to its K-nearest-neighbors for all $K \in [z + 1, cz]$, and remove the z points with largest s_x as outliers.

Theorem 4.6 (Restatement of Theorem 1.2)**.** *Let $I = (X, k, z)$ be an instance of robust k-Means and C^* be the optimal set of centers. Suppose $\mathcal{A}$ is a β-approximation algorithm for k-Means, and each cluster induced by C^* has size at least cz, then there is an algorithm that computes a set of centers C such that $f_z(X, C) \le \Psi(c)\beta \cdot f_z(X, C^*)$, where $\Psi(c) = \frac{(1+\sqrt{c-1})^2}{c-1}(x^*)^2$ and x^* is the unique real root greater than 1 of the following equation:*

$$\left[\left(1 + \sqrt{c-1}\right)^2 x^2 - (c-1)\right](x-1)^2 - 2x + 1 = 0 .$$

The proof is deferred to Appendix A.2. In particular, for $c = 3$ this gives a $\Psi(3) \approx 5.98$-approximate reduction from robust k-Means to standard k-Means, which is significantly better than the 9-approximate reduction given by Theorem 4.1. In Table 4 we list and compare the approximation ratios obtained by Theorem 4.1 and Theorem 4.6 for some values of c. The approximation ratios improve steadily as the "large-cluster" assumption becomes stronger.

4.3 Coreset and Runtime Analysis

The runtime of our proposed methods is $O(dn^2)$ when they are directly applied to the full dataset, dominated by the computation of KNN distances, which is impractical for large datasets. Therefore, computing a coreset to reduce the data size is useful. We call a subset of points an α-coreset if any β-approximate solution on the coreset yields an $\alpha\beta$-approximate solution on the original dataset.

Theorem 4.7 ([30]). *There is an algorithm that computes an $O(1)$-coreset of size $O(k \log n)$ in $O(dkn \log n)$ time.*

Thus, the full algorithm first runs the above coreset construction and then applies our proposed methods, Algorithm 1a or 1b, on the coreset to obtain the k centers. The KNN computation on the coreset takes $O(d(k \log n)^2)$ time, which is dominated by the coreset construction cost. Therefore, the overall runtime is $O(dkn \log n)$, which is near-linear in the input size.

5 Experiments

5.1 Experimental Setup

Datasets. We evaluate our method on four real-world datasets: SKIN (245,057 $\times$ 3, $k = 10$), SUSY (5,000,000 $\times$ 18, $k = 10$), SHUTTLE (43,500 $\times$ 9, $k = 10$), and KDDFULL (4,898,431 $\times$ 37, $k = 3$), following Deshpande et al. [17], Huang et al. [29], Im et al. [30]. For outlier generation, we adopt the experiment settings from Huang et al. [29]. We first normalize each dataset so that every dimension has mean 0 and standard deviation 1. For SKIN and SUSY, we inject 1% synthetic outliers uniformly sampled from the hypercube $[-\xi, \xi]^d$ with $\xi \in \{5, 10\}$, yielding the datasets SKIN-5, SKIN-10, SUSY-5, and SUSY-10. For SHUTTLE, which contains 43,500 points with 99.6% belonging to the four largest classes, we treat the two smallest classes (17 points total) as outliers. For KDDFULL, which has 4,898,431 points with 99.07% concentrated in the three largest classes, we designate the remaining 23 smaller classes (45,747 points) as outliers.

Baseline Algorithms. We compare our algorithm with the following baselines:

- TIKMeans [29], which runs $O(k/\varepsilon)$ sampling iterations and then applies a center-reduction procedure with the greedy weighted clustering method of Bhaskara et al. [6].
- IKMeans [29], which performs $O(k/\varepsilon)$ sampling iterations and then applies weighted k-means++.
- RobustKMeans++ [17], which samples $O(kn/z)$ candidate centers and applies weighted k-means++ to obtain the final clustering.
- NKMeans [30], a density-based sampling method that filters points far from high-density regions and runs clustering on the resulting coreset.
- KMeans++ [5], which is the most standard outlier-unaware heuristic for k-Means.

Each method is implemented to return k final centers. Methods that initially generate additional candidate centers apply the stated center-reduction procedure before evaluation. Given the final center set C, we remove exactly the z points with largest distances to C and evaluate the remaining points on the full dataset using $f_z(X, C)$.

Implementation Details. All experiments are conducted on a server equipped with an AMD EPYC 7532 32-Core Processor and 768 GB of RAM. We reimplement all baseline algorithms from scratch and open-source the code. To ensure a fair comparison, we optimize all implementations using standard acceleration techniques, including vectorization, multi-threading, and BLAS-optimized matrix multiplication [28] (computing distances via $\|x - y\|^2 = \|x\|^2 + \|y\|^2 - 2x^\top y$ with precomputed norms). To accelerate computationally intensive methods (specifically, OKMeans (both variants), NKMeans, and KMeans++), we utilize coreset construction via simple uniform sampling. This reduces the input to a smaller representative set, with the target outlier count scaled proportionally to the sampling ratio. The coreset sizes are set to 10,000 for SHUTTLE, 24,000 for the SKIN datasets, and 70,000 for the SUSY datasets and KDDFULL.

Experimental Details. We conduct two sets of experiments to demonstrate the effectiveness of our proposed methods, OKMeans (Algorithm 1a) and OKMeans2 (Algorithm 1b). (1) We benchmark all algorithms on the six datasets described above. For the baseline algorithms, we follow exactly the experimental settings from prior works [12, 17, 29, 30]. Our Algorithms 1a and 1b rely on K-nearest-neighbor search to compute pointwise distances, and we employ Scikit-Learn [44] for modest datasets (SKIN-5, SKIN-10, SHUTTLE) and FAISS [18] for larger datasets (SUSY-5, SUSY-10, KDDFULL) to obtain better runtime. When comparing with baseline algorithms, we set the parameter c to be 3 (recall that c affects the choice of K in KNN). (2) We perform a parameter sensitivity analysis by varying the parameter c. We use the FAISS implementation for our two algorithms in this

Table 2: Comparison results of robust k-means algorithms on each dataset. Best values are in bold, and second-best values are underlined. See more details in Section 5.2.

Method	Cost Best ↓	Cost Mean ± Std ↓	Recall ↑	Time(s) ↓	Method	Cost Best ↓	Cost Mean ± Std ↓	Recall ↑	Time(s) ↓
	SKIN-5					**SKIN-10**			
TIKMeans	5.854×10^4	$(\mathbf{6.151} \pm \mathbf{0.194}) \times \mathbf{10^4}$	0.742	3.314	TIKMeans	$\mathbf{6.303 \times 10^4}$	$(\mathbf{6.608} \pm \mathbf{0.131}) \times \mathbf{10^4}$	0.940	2.965
IKMeans	6.218×10^4	$(7.109 \pm 0.885) \times 10^4$	**0.767**	1.444	IKMeans	6.507×10^4	$(7.512 \pm 1.133) \times 10^4$	0.950	1.368
RobustKmeans++	6.679×10^4	$(7.620 \pm 0.711) \times 10^4$	0.741	1.618	RobustKmeans++	7.391×10^4	$(8.857 \pm 1.237) \times 10^4$	0.936	1.617
NKMeans	5.830×10^4	$(6.887 \pm 0.646) \times 10^4$	0.744	4.402	NKMeans	6.628×10^4	$(7.057 \pm 0.275) \times 10^4$	0.944	4.375
KMeans++	5.829×10^4	$(7.522 \pm 1.464) \times 10^4$	0.749	**0.476**	KMeans++	7.910×10^4	$(1.018 \pm 0.149) \times 10^5$	0.891	**0.480**
OKMeans (Ours)	6.060×10^4	$(6.494 \pm 0.312) \times 10^4$	0.757	2.056	OKMeans (Ours)	6.472×10^4	$(6.982 \pm 0.671) \times 10^4$	0.949	2.045
OKMeans2 (Ours)	$\mathbf{5.768 \times 10^4}$	$(6.410 \pm 0.300) \times 10^4$	0.746	2.395	OKMeans2 (Ours)	6.490×10^4	$(6.707 \pm 0.166) \times 10^4$	**0.955**	2.441
	SUSY-5					**SUSY-10**			
TIKMeans	5.103×10^7	$(5.165 \pm 0.048) \times 10^7$	0.759	94.824	TIKMeans	5.197×10^7	$(5.272 \pm 0.040) \times 10^7$	0.975	89.186
IKMeans	5.145×10^7	$(5.279 \pm 0.104) \times 10^7$	0.782	46.489	IKMeans	5.194×10^7	$(5.334 \pm 0.087) \times 10^7$	0.976	42.134
RobustKmeans++	5.640×10^7	$(5.791 \pm 0.132) \times 10^7$	0.708	74.735	RobustKmeans++	5.759×10^7	$(5.966 \pm 0.130) \times 10^7$	0.973	74.092
NKMeans	4.795×10^7	$(4.825 \pm 0.022) \times 10^7$	0.739	17.346	NKMeans	4.905×10^7	$(4.941 \pm 0.028) \times 10^7$	0.972	17.529
KMeans++	4.789×10^7	$(4.885 \pm 0.087) \times 10^7$	**0.831**	**9.725**	KMeans++	4.900×10^7	$(5.173 \pm 0.235) \times 10^7$	0.980	**9.728**
OKMeans (Ours)	$\mathbf{4.756 \times 10^7}$	$(4.830 \pm 0.064) \times 10^7$	0.799	12.840	OKMeans (Ours)	$\mathbf{4.836 \times 10^7}$	$(\mathbf{4.873} \pm \mathbf{0.023}) \times \mathbf{10^7}$	0.982	12.919
OKMeans2 (Ours)	4.756×10^7	$(\mathbf{4.801} \pm \mathbf{0.034}) \times \mathbf{10^7}$	0.798	13.469	OKMeans2 (Ours)	4.848×10^7	$(4.876 \pm 0.020) \times 10^7$	**0.982**	13.407
	SHUTTLE					**KDDFULL**			
TIKMeans	$\mathbf{5.710 \times 10^4}$	$(\mathbf{6.122} \pm \mathbf{0.260}) \times \mathbf{10^4}$	0.141	0.543	TIKMeans	3.318×10^7	$(\mathbf{3.320} \pm \mathbf{0.002}) \times \mathbf{10^7}$	0.601	49.850
IKMeans	6.220×10^4	$(7.277 \pm 0.503) \times 10^4$	0.141	**0.243**	IKMeans	3.324×10^7	$(3.937 \pm 0.861) \times 10^7$	0.615	21.419
RobustKmeans++	6.246×10^4	$(7.830 \pm 1.193) \times 10^4$	0.094	0.363	RobustKmeans++	3.448×10^7	$(4.845 \pm 1.260) \times 10^7$	0.611	40.355
NKMeans	6.487×10^4	$(7.260 \pm 0.353) \times 10^4$	**0.171**	4.785	NKMeans	3.323×10^7	$(3.466 \pm 0.178) \times 10^7$	**0.624**	14.859
KMeans++	6.048×10^4	$(7.055 \pm 0.705) \times 10^4$	0.082	0.294	KMeans++	3.321×10^7	$(6.123 \pm 2.152) \times 10^7$	0.553	**6.894**
OKMeans (Ours)	6.262×10^4	$(6.823 \pm 0.625) \times 10^4$	0.171	0.785	OKMeans (Ours)	$\mathbf{3.308 \times 10^7}$	$(4.019 \pm 0.770) \times 10^7$	0.616	9.510
OKMeans2 (Ours)	6.262×10^4	$(6.823 \pm 0.625) \times 10^4$	0.171	0.776	OKMeans2 (Ours)	3.308×10^7	$(4.250 \pm 0.919) \times 10^7$	0.613	10.269

experiment, setting $c \in \{3, 4, 5, 7, 10, 15, 20\}$. We further experiment with the classic setting of the KNN heuristic, where K is usually chosen to be a constant, using $K \in \{2, 5, 10\}$.

5.2 Results and Analysis

Table 2 reports the clustering cost, outlier recall, and running time of all methods across datasets. For each algorithm, we define the *Cost Best* as the minimum clustering cost achieved over 10 independent runs. The means of *Recall* ($|\text{detected} \cap \text{true outliers}|/|\text{true outliers}|$) and *Time* are reported in the table. We also report *Cost Mean* $\pm$ *Std* to demonstrate the stability of the solutions found. We provide the full table of cost, recall, and runtime (means and stds) in Table 6.

Performance Comparison. As shown in Table 2, our *OKMeans* and *OKMeans2* demonstrate a good balance of clustering quality and computational efficiency.

- **Clustering Quality:** Our methods achieve competitive clustering costs across all datasets and often match or outperform strong robust clustering baselines, especially on the large-scale SUSY and KDDFULL settings. For instance, on the large-scale SUSY-10 dataset, *OKMeans* achieves a mean cost of 4.873×10^7, improving upon *TIKMeans* (5.272×10^7) and *NKMeans* (4.941×10^7).
- **Runtime Efficiency:** While the classic *KMeans++* is generally the fastest, it is highly unstable and fails to optimize the robust objective effectively. This instability motivates the need for robust algorithms. On large-scale datasets, our methods are substantially faster than several robust baselines. For instance, on SUSY-10, OKMeans requires only ≈ 12.9 seconds, compared with RobustKmeans++ (≈ 74.1s) and TIKMeans (≈ 89.2s). On KDDFULL, OKMeans is nearly $5\times$ faster than TIKMeans while achieving a lower best cost.
- **Scalability:** The results on large datasets (SUSY, KDDFULL) highlight the effectiveness of our FAISS-accelerated implementation. While *TIKMeans* performs well on small data (SKIN), its runtime degrades severely on larger datasets. In contrast, *OKMeans* maintains high scalability without sacrificing solution quality while maintaining competitive clustering quality.

Parameter Sensitivity. For the KNN heuristic with *constant values* of K, we observe a significant increase in cost on the KDDFULL dataset, as demonstrated in Table 3. The performance on other datasets is comparable with OKMeans and is deferred to Table 7. This suggests that for the robust k-Means objective, selecting K to be a small fixed constant might

Table 3: Performance of the KNN heuristic on KDDFULL with constant values of K.

Metric	OKMeans	KNN ($K = 2$)	KNN ($K = 5$)	KNN ($K = 10$)
Cost	4.019×10^7	4.612×10^7	4.371×10^7	4.631×10^7
Recall	0.616	0.5605	0.5208	0.5693
Time (s)	9.510	9.141	9.273	9.104

be suboptimal in certain cases. The empirical results align with our theoretical finding that K should scale linearly in the number of outliers to guarantee consistent good performance.

In Appendix C, Table 8 presents the sensitivity of our methods to the parameter c. We provide additional visualizations in Figures 1 to 3. We observe that *performance is robust to the choice of* c. Moderate values (e.g., $c \in [3, 7]$) are generally sufficient to reach the "sweet spot" where the cost is minimized. For example, on SUSY-5, OKMeans achieves its best cost at $c = 5$ (4.814×10^7), with minimal variance across other settings. Increasing c further to 15 or 20 yields diminishing returns and does not significantly improve the clustering objective.

6 Discussions

Our findings suggest that outlier detection and robust clustering should be studied in a unified framework, leading to several immediate directions for future work.

Tighten the analysis of KNN. Our current analysis of the KNN heuristic is not tight. The exact approximation guarantees for our KNN-based methods remain open.

Many inequalities in our proof are tight only when the points are located at specific extremal positions, and it is unclear whether these worst cases are incompatible and can lead to a substantially improved bound. Another main source of slack is that we bound the clustering cost of the output of our algorithm assuming that the centers are the same as the centers in the optimal clustering. While this makes it convenient to track how much the cost changes due to mistakenly included outliers, the inliers identified by our algorithm may admit a better set of centers which may induce a better clustering cost.

Outlier detection for robust clustering. It would be interesting to analyze other widely used outlier detection heuristics, particularly variants of the KNN heuristic, and investigate whether they also admit strong theoretical guarantees or empirical performance for the Robust k-Means objective.

Robust clustering for outlier detection. Another promising direction is to adapt advanced techniques from the robust clustering literature to design more effective outlier detection algorithms. To the best of our knowledge, this direction remains largely underexplored.

Beyond worst-case analysis. Our results focus on worst-case approximation guarantees, where both cluster structure and outliers are adversarial. Exploring refined analyses under additional, practically motivated assumptions on the data generation process may yield significantly stronger guarantees.

7 Conclusion

In this paper, we establish a direct connection between the classical KNN-based outlier detection heuristic and robust clustering. We provide the first rigorous theoretical analysis for the approximation guarantee of KNN in the robust clustering setting, explaining its strong empirical performance, and show experimentally that it matches more sophisticated methods. Our results highlight that a surprisingly simple heuristic can be highly effective, offering a new perspective on robust clustering algorithm design.

Acknowledgments and Disclosure of Funding

Tianle Jiang is supported by NSF grant IIS-2402823.

References

[1] Akanksha Agrawal, Tanmay Inamdar, Saket Saurabh, and Jie Xue. Clustering what matters: Optimal approximation for clustering with outliers. *J. Artif. Intell. Res.*, 78:143–166, 2023.

[2] Sara Ahmadian, Ashkan Norouzi-Fard, Ola Svensson, and Justin Ward. Better guarantees for k-means and euclidean k-median by primal-dual algorithms. In *FOCS*, pages 61–72. IEEE Computer Society, 2017.

[3] Matteo Almanza, Alessandro Epasto, Alessandro Panconesi, and Giuseppe Re. k-clustering with fair outliers. In *WSDM*, pages 5–15. ACM, 2022.

[4] Sanjeev Arora, Prabhakar Raghavan, and Satish Rao. Approximation schemes for euclidean k-medians and related problems. In *STOC*, pages 106–113. ACM, 1998.

[5] David Arthur and Sergei Vassilvitskii. k-means++ the advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035, 2007.

[6] Aditya Bhaskara, Sharvaree Vadgama, and Hong Xu. Greedy sampling for approximate clustering in the presence of outliers. In *NeurIPS*, pages 11146–11155, 2019.

[7] Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George van den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, Diego de Las Casas, Aurelia Guy, Jacob Menick, Roman Ring, Tom Hennigan, Saffron Huang, Loren Maggiore, Chris Jones, Albin Cassirer, Andy Brock, Michela Paganini, Geoffrey Irving, Oriol Vinyals, Simon Osindero, Karen Simonyan, Jack W. Rae, Erich Elsen, and Laurent Sifre. Improving language models by retrieving from trillions of tokens. In *ICML*, volume 162 of *Proceedings of Machine Learning Research*, pages 2206–2240. PMLR, 2022.

[8] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM Comput. Surv.*, 41(3):15:1–15:58, 2009.

[9] Moses Charikar, Sudipto Guha, Éva Tardos, and David B. Shmoys. A constant-factor approximation algorithm for the k-median problem (extended abstract). In *STOC*, pages 1–10. ACM, 1999.

[10] Moses Charikar, Samir Khuller, David M. Mount, and Giri Narasimhan. Algorithms for facility location problems with outliers. In *SODA*, pages 642–651. ACM/SIAM, 2001.

[11] Moses Charikar, Vincent Cohen-Addad, Ruiquan Gao, Fabrizio Grandoni, Euiwoong Lee, and Ernest van Wijland. An improved greedy approximation for (metric) k-means. In *FOCS*, pages 233–240. IEEE, 2025.

[12] Sanjay Chawla and Aristides Gionis. k-means-: A unified approach to clustering and outlier detection. In *SDM*, pages 189–197. SIAM, 2013.

[13] Ke Chen. A constant factor approximation algorithm for k-median clustering with outliers. In *SODA*, pages 826–835. SIAM, 2008.

[14] Xianrun Chen, Lu Han, Dachuan Xu, Yicheng Xu, and Yong Zhang. k-median/means with outliers revisited: A simple fpt approximation. In *COCOON (2)*, volume 14423 of *Lecture Notes in Computer Science*, pages 295–302. Springer, 2023.

[15] Vincent Cohen-Addad, Hossein Esfandiari, Vahab S. Mirrokni, and Shyam Narayanan. Improved approximations for euclidean k-means and k-median, via nested quasi-independent sets. In *STOC*, pages 1621–1628. ACM, 2022.

[16] Vincent Cohen-Addad, Fabrizio Grandoni, Euiwoong Lee, Chris Schwiegelshohn, and Ola Svensson. A $(2+\epsilon)$-approximation algorithm for metric k-median. In *STOC*, pages 615–624. ACM, 2025.

[17] Amit Deshpande, Praneeth Kacham, and Rameshwar Pratap. Robust k-means++. In *UAI*, volume 124 of *Proceedings of Machine Learning Research*, pages 799–808. AUAI Press, 2020.

[18] Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. The faiss library. *IEEE Transactions on Big Data*, 2025.

[19] Eleazar Eskin, Andrew Arnold, Michael J. Prerau, Leonid Portnoy, and Salvatore J. Stolfo. A geometric framework for unsupervised anomaly detection. In *Applications of Data Mining in Computer Security*, Advances in Information Security, pages 77–101. Springer, 2002.

[20] Qilong Feng, Zhen Zhang, Ziyun Huang, Jinhui Xu, and Jianxin Wang. Improved algorithms for clustering with outliers. In *ISAAC*, volume 149 of *LIPIcs*, pages 61:1–61:12. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.

[21] Zachary Friggstad, Kamyar Khodamoradi, Mohsen Rezapour, and Mohammad R. Salavatipour. Approximation schemes for clustering with outliers. *ACM Trans. Algorithms*, 15(2):26:1–26:26, 2019.

[22] Amol Ghoting, Srinivasan Parthasarathy, and Matthew Eric Otey. Fast mining of distance-based outliers in high-dimensional datasets. *Data Min. Knowl. Discov.*, 16(3):349–364, 2008.

[23] Kishen N. Gowda, Thomas W. Pensyl, Aravind Srinivasan, and Khoa Trinh. Improved bi-point rounding algorithms and a golden barrier for *k*-median. In *SODA*, pages 987–1011. SIAM, 2023.

[24] Christoph Grunau and Václav Rozhon. Adapting k-means algorithms for outliers. In *ICML*, volume 162 of *Proceedings of Machine Learning Research*, pages 7845–7886. PMLR, 2022.

[25] Ruiqi Guo, Philip Sun, Erik Lindgren, Quan Geng, David Simcha, Felix Chern, and Sanjiv Kumar. Accelerating large-scale inference with anisotropic vector quantization. In *ICML*, volume 119 of *Proceedings of Machine Learning Research*, pages 3887–3896. PMLR, 2020.

[26] Anupam Gupta, Benjamin Moseley, and Rudy Zhou. Structural iterative rounding for generalized k-median problems. In *ICALP*, volume 198 of *LIPIcs*, pages 77:1–77:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

[27] Shalmoli Gupta, Ravi Kumar, Kefu Lu, Benjamin Moseley, and Sergei Vassilvitskii. Local search methods for k-means with outliers. *Proc. VLDB Endow.*, 10(7):757–768, 2017.

[28] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020. doi: 10.1038/s41586-020-2649-2. URL `https://doi.org/10.1038/s41586-020-2649-2`.

[29] Junyu Huang, Qilong Feng, Ziyun Huang, Jinhui Xu, and Jianxin Wang. Near-linear time approximation algorithms for k-means with outliers. In *Proceedings of the 41st International Conference on Machine Learning*, pages 19723–19756, 2024.

[30] Sungjin Im, Mahshid Montazer Qaem, Benjamin Moseley, Xiaorui Sun, and Rudy Zhou. Fast noise removal for k-means clustering. In *AISTATS*, volume 108 of *Proceedings of Machine Learning Research*, pages 456–466. PMLR, 2020.

[31] Ragesh Jaiswal and Amit Kumar. Clustering what matters in constrained settings: Improved outlier to outlier-free reductions. In *ISAAC*, volume 283 of *LIPIcs*, pages 41:1–41:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023.

[32] Suhas Jayaram Subramanya, Fnu Devvrit, Harsha Vardhan Simhadri, Ravishankar Krishnawamy, and Rohan Kadekodi. Diskann: Fast accurate billion-point nearest neighbor search on a single node. *Advances in neural information processing Systems*, 32, 2019.

[33] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with gpus. *IEEE Trans. Big Data*, 7(3):535–547, 2021.

[34] Tapas Kanungo, David M. Mount, Nathan S. Netanyahu, Christine D. Piatko, Ruth Silverman, and Angela Y. Wu. A local search approximation algorithm for k-means clustering. *Comput. Geom.*, 28(2-3):89–112, 2004.

[35] Edwin M. Knorr and Raymond T. Ng. Algorithms for mining distance-based outliers in large datasets. In *VLDB*, pages 392–403. Morgan Kaufmann, 1998.

[36] Yufeng Kou, Chang-Tien Lu, and Dechang Chen. Spatial weighted outlier detection. In *SDM*, pages 614–618. SIAM, 2006.

[37] Ravishankar Krishnaswamy, Shi Li, and Sai Sandeep. Constant approximation for k-median and k-means with outliers via iterative rounding. In *STOC*, pages 646–659. ACM, 2018.

[38] Ravishankar Krishnaswamy, Magdalen Dobson Manohar, and Harsha Vardhan Simhadri. The diskann library: Graph-based indices for fast, fresh and filtered vector search. *IEEE Data Eng. Bull.*, 48(3):20–42, 2024.

[39] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *NeurIPS*, 2020.

[40] Yihua Liao and V. Rao Vemuri. Use of k-nearest neighbor classifier for intrusion detection. *Comput. Secur.*, 21(5):439–448, 2002.

[41] Yury A. Malkov and Dmitry A. Yashunin. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE Trans. Pattern Anal. Mach. Intell.*, 42(4):824–836, 2020.

[42] Jirí Matousek. On approximate geometric k-clustering. *Discret. Comput. Geom.*, 24(1):61–84, 2000.

[43] Matthew Eric Otey, Amol Ghoting, and Srinivasan Parthasarathy. Fast distributed outlier detection in mixed-attribute data sets. *Data Min. Knowl. Discov.*, 12(2-3):203–228, 2006.

[44] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[45] Sridhar Ramaswamy, Rajeev Rastogi, and Kyuseok Shim. Efficient algorithms for mining outliers from large data sets. In *SIGMOD Conference*, pages 427–438. ACM, 2000.

[46] Yufei Tao, Xiaokui Xiao, and Shuigeng Zhou. Mining distance-based outliers from large databases in any metric space. In *KDD*, pages 394–403. ACM, 2006.

[47] Mengzhao Wang, Xiaoliang Xu, Qiang Yue, and Yuxiang Wang. A comprehensive survey and experimental comparison of graph-based approximate nearest neighbor search. *Proc. VLDB Endow.*, 14(11):1964–1978, 2021.

[48] Li Wei, Weining Qian, Aoying Zhou, Wen Jin, and Jeffrey Xu Yu. HOT: hypergraph-based outlier test for categorical data. In *PAKDD*, volume 2637 of *Lecture Notes in Computer Science*, pages 399–410. Springer, 2003.

[49] Mingxi Wu and Chris Jermaine. Outlier detection by sampling with accuracy guarantees. In *KDD*, pages 767–772. ACM, 2006.

[50] Ji Zhang and Hai H. Wang. Detecting outlying subspaces for high-dimensional data: the new task, algorithms, and performance. *Knowl. Inf. Syst.*, 10(3):333–355, 2006.

[51] Yang Zhang, Nirvana Meratnia, and Paul J. M. Havinga. Outlier detection techniques for wireless sensor networks: A survey. *IEEE Commun. Surv. Tutorials*, 12(2):159–170, 2010.

Appendix

A Deferred Proofs

A.1 Full Proof of Theorem 4.1

We extend the analysis to general $c > 1$. To justify the choice of $r_x = d(x, \mathrm{NN}_X(x, \frac{c+1}{2}z))$ in Algorithm 1a, we temporarily assume $r_x := d(x, \mathrm{NN}_X(x, \alpha z))$ for some $1 < \alpha < c$ to be optimized later.

Recall that we fix an optimal set of centers C^*, and call the points in $X_z(C^*)$ "true inliers" and the points in $X \setminus X_z(C^*)$ "true outliers". Define $Y = O \cap X_z(C^*)$ as the set of true inliers removed by the algorithm by mistake, and $Z' = (X \setminus O) \cap (X \setminus X_z(C^*))$ as the set of true outliers included by the algorithm by mistake. We have $|Y| = |Z'|$, and again we assume that $|Y| \geq 1$. Let $T = \min_{x \in O}\{r_x\}$, i.e., the threshold separating the r_x of $X \setminus O$ and O.

The following two lemmas are analogous to Lemma 4.2 and Lemma 4.3.

Lemma A.1. *Suppose each optimal cluster has size at least cz, and $\min_{y \in Y} d(y, C^*) = \lambda T$ for some $\lambda \geq 0$, then $f_z(X, C^*) \geq |Y|T^2 \cdot \max\{\lambda^2, (c-\alpha)(1-\lambda)^2\}$ when $\lambda \in [0, 1]$, and $f_z(X, C^*) \geq |Y|T^2 \cdot \lambda^2$ when $\lambda > 1$.*

Proof. Since every $y \in Y$ satisfies $d(y, C^*) \geq \lambda T$, we have

$$f_z(X, C^*) \geq f(Y, C^*) = \sum_{y \in Y} d(y, C^*)^2 \geq |Y|T^2 \cdot \lambda^2 \,.$$

When $\lambda \in [0, 1]$, fix a point $y \in Y$ such that $d(y, C^*) = \lambda T$. Since $y \in O$, we have $r_y \geq T$, which implies $|B_X^\circ(y, T)| \leq \alpha z$. By the assumption that each optimal cluster has size at least cz, at least $(c - \alpha)z$ points in $\mathrm{cluster}_z(y, C^*)$ are not in $B_X^\circ(y, T)$, and for each such point x, by the triangle inequality, we have

$$\begin{aligned} d(x, \mathrm{closest}(y, C^*)) &\geq d(x, y) - d(y, \mathrm{closest}(y, C^*)) \\ &\geq T - \lambda T = T \cdot (1 - \lambda) \,, \end{aligned}$$

therefore,

$$\begin{aligned} f_z(X, C^*) &\geq f(\mathrm{cluster}_z(y, C^*) \setminus B_X^\circ(y, T), C^*) \\ &\geq (c - \alpha)zT^2 \cdot (1 - \lambda)^2 \geq |Y|T^2 \cdot (c - \alpha)(1 - \lambda)^2 \,. \end{aligned}$$

The last inequality follows from the fact that $|Y| \leq |O| = z$. □

Lemma A.2. *Suppose each optimal cluster has size at least cz, and $\max_{o \in Z'} d(o, C^*) = \delta T$ for some $\delta > 1$, then $f_z(X, C^*) \geq |Y|T^2 \cdot (\alpha - 1)(\delta - 1)^2$.*

Proof. Let $o = \arg\max_{o' \in Z'}\{d(o', C^*)\}$, so $d(o, C^*) = \delta T$. Since $o \in X \setminus O$, we have $r_o \leq T$, which implies $|B_X(o, T)| \geq \alpha z$. Therefore, since there are at most z outliers, there must be at least $(\alpha - 1)z$ true inliers in $B_X(o, T)$. For each such inlier x, we must have $d(x, C^*) \geq (\delta - 1)T$, otherwise there would be the following contradiction:

$$d(o, C^*) \leq d(o, x) + d(x, C^*) < T + (\delta - 1)T = \delta T \,.$$

Thus, we must have

$$\begin{aligned} f_z(X, C^*) &\geq \sum_{x \in B_X(o,T) \cap X_z(C^*)} d(x, C^*)^2 \geq (\alpha - 1)zT^2 \cdot (\delta - 1)^2 \\ &\geq |Y|T^2 \cdot (\alpha - 1)(\delta - 1)^2 \,. \end{aligned}$$

□

The following lemma is analogous to Lemma 4.4, but in the proof we parameterize the thresholds between the three cases to obtain a tight analysis.

Lemma A.3. *Suppose each optimal cluster has size at least cz, then*

$$f(X \setminus O) \leq f_z(X, C^*) \cdot \frac{(1+\sqrt{(c-1)/2})^2}{(c-1)/2}(x^*)^2 \,.$$

where x^* is the unique real solution greater than 1 to the following equation:

$$\left[\left(1+\sqrt{\frac{c-1}{2}}\right)^2 x^2 - \frac{c-1}{2}\right](x-1)^2 - 2x + 1 = 0 \,.$$

Proof. We distinguish between the following three cases: let $\delta^* > 1$,

Case 1 If $\max_{o\in Z'} d(o, C^*) \leq \delta^* T$, then by Lemma A.1, assuming $\min_{y\in Y} d(y, C^*) = \lambda T$ for some $\lambda \geq 0$, then we have

$$\begin{aligned} f(X \setminus O, C^*) &\leq f_z(X, C^*) + f(Z', C^*) - f(Y, C^*) \\ &\leq f_z(X, C^*) + |Z'| \cdot (\delta^* T)^2 - |Y| \cdot (\lambda T)^2 \\ &\leq \begin{cases} f_z(X, C^*) \cdot \left(1 + \frac{(\delta^*)^2 - \lambda^2}{\max\{\lambda^2, (c-\alpha)(1-\lambda)^2\}}\right), \text{if } 0 \leq \lambda \leq 1 \\ f_z(X, C^*) \cdot (\delta^*)^2/\lambda^2, \text{if } \lambda > 1 \end{cases} . \end{aligned}$$

Case 2 If $\max_{o\in Z'} d(o, C^*) = \delta T$ for some $\delta > \delta^*$, and $d(y, C^*) \geq (\delta - 1)T$ for all $y \in Y$, then by Lemma A.1, we have

$$\begin{aligned} f(X \setminus O, C^*) &\leq f_z(X, C^*) + f(Z', C^*) - f(Y, C^*) \\ &\leq f_z(X, C^*) + |Z'| \cdot (\delta T)^2 - |Y| \cdot ((\delta - 1)T)^2 \\ &\leq f_z(X, C^*) \cdot \left(1 + \frac{2\delta - 1}{\max\{1, \alpha - 1\} \cdot (\delta - 1)^2}\right) \\ &\leq f_z(X, C^*) \cdot \left(1 + \frac{2\delta^* - 1}{\max\{1, \alpha - 1\} \cdot (\delta^* - 1)^2}\right) . \end{aligned}$$

The last inequality is due to the monotonicity of $\frac{2\delta-1}{(\delta-1)^2}$.

Case 3 If $\max_{o\in Z'} d(o, C^*) = \delta T$ for some $\delta > \delta^*$, and $\min_{y\in Y} d(y, C^*) = \lambda T$ for some $\lambda < \delta - 1$, then by Lemma A.1 and Lemma A.2, we have

$$\begin{aligned} f(X \setminus O, C^*) &\leq f_z(X, C^*) + f(Z', C^*) - f(Y, C^*) \\ &\leq f_z(X, C^*) + |Z'| \cdot (\delta T)^2 - |Y| \cdot (\lambda T)^2 \\ &\leq \begin{cases} f_z(X, C^*) \cdot \left(1 + \frac{\delta^2 - \lambda^2}{\max\{(c-\alpha)(1-\lambda)^2, (\alpha-1)(\delta-1)^2\}}\right), \text{if } 0 \leq \lambda \leq 1 \\ f_z(X, C^*) \cdot \left(1 + \frac{\delta^2 - \lambda^2}{\max\{\lambda^2, (\alpha-1)(\delta-1)^2\}}\right), \text{if } \lambda > 1 \end{cases} . \end{aligned}$$

Here in the case of $0 \leq \lambda \leq 1$, we omit the λ^2 term in the denominator as indicated by Lemma A.1. This simplifies the analysis and does not affect the optimality of the derived bound.

It remains to optimize over the choice of α and δ^* such that the maximum over the upper bounds in the three cases is minimized. It can be verified that the optimal choice is $\alpha = \frac{c+1}{2}$, then the upper bound in Case 1 is maximized at $\lambda^* = \frac{\sqrt{(c-1)/2}}{1+\sqrt{(c-1)/2}}$ and takes value $U_1(\delta^*) = \frac{(1+\sqrt{(c-1)/2})^2}{(c-1)/2}(\delta^*)^2$; the upper bound in Case 2 is $U_2(\delta^*) = 1 + \frac{2\delta^*-1}{(c-1)/2\cdot(\delta^*-1)^2}$; the upper bound in Case 3 turns out to be dominated by Case 2. When $\delta^* > 1$, $U_1(\delta^*)$ is increasing in δ^* while $U_2(\delta^*)$ is decreasing in δ^*, so the optimal choice of δ^* is given by $U_1(\delta^*) = U_2(\delta^*)$, which simplifies to

$$\left[\left(1+\sqrt{\frac{c-1}{2}}\right)^2 (\delta^*)^2 - \frac{c-1}{2}\right](\delta^* - 1)^2 - (2\delta^* - 1) = 0 \,.$$

Then in all three cases we have $f(X \setminus O, C^*) \leq f_z(X, C^*) \cdot U_1(\delta^*)$. □

A.2 Proof of Theorem 4.6

To justify using the sum of all K-nearest neighbors for $K \in [z+1, cz]$ instead of another interval, we further parameterize the algorithm: suppose we use the sum of distances to the $[lz+1, rz]$-th neighbors as a score (such that $1 \le l \le r \le c$). Assume $(r-l)zT$ is the threshold that separates the scores of $X \setminus O$ and O. We have the following two lower bounds on the cost of the optimal solution, which are analogous to Lemma A.1 and Lemma A.2. The only difference is that the bounds are now stronger since we use a sum over multiple neighbors as the outlier score.

Lemma A.4. *Suppose each optimal cluster has size at least cz, and $\min_{y\in Y} d(y, C^*) = \lambda T$ for some $\lambda \ge 0$, then we have $f_z(X, C^*) \ge |Y|T^2 \cdot \max\{\lambda^2, (c-l)(1-\lambda)^2\}$ when $\lambda \in [0,1]$, and $f_z(X, C^*) \ge |Y|T^2 \cdot \lambda^2$ when $\lambda > 1$.*

Proof. Since every $y \in Y$ satisfies $d(y, C^*) \ge \lambda T$, we have

$$f_z(X, C^*) \ge f(Y, C^*) = \sum_{y\in Y} d(y, C^*)^2 \ge |Y|T^2 \cdot \lambda^2 \,.$$

When $\lambda \in [0,1]$, fix a point $y \in Y$ such that $d(y, C^*) = \lambda T$. Let $x_i = \mathrm{NN}_{\mathrm{cluster}_z(y,C^*)}(y, lz+i)$. Using a similar argument as in the proof of Lemma A.1, each of the $[rz+1, cz]$-th nearest neighbors of y is at distance at least $T \cdot |1-\lambda|$ from C^*, therefore,

$$\sum_{i=(r-l)z+1}^{(c-l)z} d(x_i, C^*)^2 \ge (c-r)zT^2 \cdot (1-\lambda)^2 \,.$$

Further, by the assumption that $y \in O$, we have $\sum_{i=1}^{(r-l)z} d(x_i, y) \ge (r-l)zT$. Then, by the triangle inequality and Jensen's inequality, we have

$$\begin{aligned}
\sum_{i=1}^{(r-l)z} d(x_i, C^*)^2 &\ge \sum_{i=1}^{(r-l)z} (d(x_i, y) - d(y, C^*))^2 \\
&\ge (r-l)z \cdot \left(\frac{1}{(r-l)z} \sum_{i=1}^{(r-l)z} (d(x_i, y) - d(y, C^*)) \right)^2 \\
&\ge (r-l)z \cdot (T - d(y, C^*))^2 \\
&\ge (r-l)zT^2 \cdot (1-\lambda)^2
\end{aligned}$$

Therefore,

$$\begin{aligned}
f_z(X, C^*) &\ge \sum_{i=1}^{(c-l)z} d(x_i, C^*)^2 = \sum_{i=1}^{(r-l)z} d(x_i, C^*)^2 + \sum_{i=(r-l)z+1}^{(c-l)z} d(x_i, C^*)^2 \\
&\ge (c-l)zT^2 \cdot (1-\lambda)^2 \ge |Y|T^2 \cdot (c-l)(1-\lambda)^2 \,.
\end{aligned}$$

The last inequality follows from the fact that $|Y| \le |O| = z$. □

Lemma A.5. *Suppose each optimal cluster has size at least cz, and $\max_{o\in Z'} d(o, C^*) = \delta T$ for some $\delta > 1$, then $f_z(X, C^*) \ge |Y|T^2 \cdot (r-l)(\delta-1)^2$.*

Proof. Let $o = \arg\max_{o'\in Z'}\{d(o', C^*)\}$, so $d(o, C^*) = \delta T$. Let x_i denote the i-th closest true inlier to o, then since there are at most z true outliers, we have

$$\sum_{i=(l-1)z+1}^{(r-1)z} d(x_i, o) \le \sum_{i=lz+1}^{rz} d(o, \mathrm{NN}_X(o, i)) \le (r-l)zT \,.$$

Let $C_i = \mathrm{closest}(x_i, C^*)$, then by the triangle inequality, we have

$$d(C_i, x_i) + d(x_i, o) \ge d(C_i, o), \ \forall i \in [cz] \,,$$

therefore, by Jensen's inequality,

$$\begin{aligned}\sum_{i=(l-1)z+1}^{(r-1)z} d(C_i,x_i)^2 &\geq \sum_{i=(l-1)z+1}^{(r-1)z} \left(\max\{d(C_i,o)-d(x_i,o),0\}\right)^2 \\ &\geq (r-l)z\cdot\left(\max\left\{\frac{1}{(r-l)z}\sum_{i=(l-1)z+1}^{(r-1)z}(d(C_i,o)-d(x_i,o)),0\right\}\right)^2 \\ &\geq (r-l)z\cdot\left(\max\left\{d(o,C^*)-d(x_i,o),0\right\}\right)^2 \\ &\geq (r-l)z\cdot(\delta T-T)^2=(r-l)zT^2\cdot(\delta-1)^2\,.\end{aligned}$$

Therefore,

$$f_z(X,C^*)\geq\sum_{i=(l-1)z+1}^{(r-1)z} d(C_i,x_i)^2\geq(r-l)zT^2\cdot(\delta-1)^2\geq|Y|T^2\cdot(r-l)(\delta-1)^2\,.$$

□

Lemma A.6. *Suppose each optimal cluster has size at least cz, then*

$$f(X\setminus O)\leq f_z(X,C^*)\cdot\frac{(1+\sqrt{c-1})^2}{c-1}(x^*)^2$$

where x^* is the unique real root greater than 1 of the following equation:

$$\left[\left(1+\sqrt{c-1}\right)^2x^2-(c-1)\right](x-1)^2-2x+1=0\,.$$

Proof. We again distinguish between the following three cases: let $\delta^*>1$,

Case 1 If $\max_{o\in Z'}d(o,C^*)\leq\delta^*T$, then by Lemma A.4, for all $\lambda>0$, assume $\min_{y\in Y}d(y,C^*)=\lambda T$ then we have

$$\begin{aligned}f(X\setminus O,C^*)&\leq f_z(X,C^*)+f(Z',C^*)-f(Y,C^*)\\&\leq f_z(X,C^*)+|Z'|\cdot(\delta^*T)^2-|Y|\cdot(\lambda T)^2\\&\leq\begin{cases}f_z(X,C^*)\cdot\left(1+\frac{(\delta^*)^2-\lambda^2}{\max\{\lambda^2,(c-l)(1-\lambda)^2\}}\right),\text{if }0\leq\lambda\leq1\\f_z(X,C^*)\cdot(\delta^*)^2/\lambda^2,\text{if }\lambda>1\end{cases}.\end{aligned}$$

Case 2 If $\max_{o\in Z'}d(o,C^*)=\delta T$ for some $\delta>\delta^*$, and $d(y,C^*)\geq(\delta-1)T$ for all $y\in Y$, then by Lemma A.4, we have

$$\begin{aligned}f(X\setminus O,C^*)&\leq f_z(X,C^*)+f(Z',C^*)-f(Y,C^*)\\&\leq f_z(X,C^*)+|Z'|\cdot(\delta T)^2-|Y|\cdot((\delta-1)T)^2\\&\leq f_z(X,C^*)\cdot\left(1+\frac{2\delta-1}{\max\{1,r-l\}\cdot(\delta-1)^2}\right)\\&\leq f_z(X,C^*)\cdot\left(1+\frac{2\delta^*-1}{\max\{1,r-l\}\cdot(\delta^*-1)^2}\right)\,.\end{aligned}$$

Case 3 If $\max_{o\in Z'}d(o,C^*)=\delta T$ for some $\delta>\delta^*$, and $\min_{y\in Y}d(y,C^*)=\lambda T$ for some $\lambda<\delta-1$, then by Lemma A.4 and Lemma A.5, we have

$$\begin{aligned}f(X\setminus O,C^*)&\leq f_z(X,C^*)+f(Z',C^*)-f(Y,C^*)\\&\leq f_z(X,C^*)+|Z'|\cdot(\delta T)^2-|Y|\cdot(\lambda T)^2\\&\leq\begin{cases}f_z(X,C^*)\cdot\left(1+\frac{\delta^2-\lambda^2}{\max\{(c-l)(1-\lambda)^2,(r-l)(\delta-1)^2\}}\right),\text{if }0\leq\lambda\leq1\\f_z(X,C^*)\cdot\left(1+\frac{\delta^2-\lambda^2}{\max\{\lambda^2,(r-l)(\delta-1)^2\}}\right),\text{if }\lambda>1\end{cases}.\end{aligned}$$

It is straightforward to see that to minimize the upper bounds, we should choose $r = c$ and $l = 1$. It remains to optimize the choice of δ^* such that the maximum over the upper bounds in the three cases is minimized. Similar to Lemma A.3, the upper bound in Case 1 is maximized at $\lambda^* = \frac{\sqrt{c-1}}{1+\sqrt{c-1}}$ and takes value $V_1(\delta^*) = \frac{(1+\sqrt{c-1})^2}{c-1}(\delta^*)^2$; the upper bound in Case 2 is $V_2(\delta^*) = 1 + \frac{2\delta^*-1}{(c-1)\cdot(\delta^*-1)^2}$, and the upper bound in Case 3 is dominated by Case 2. The optimal choice of δ^* is given by $V_1(\delta^*) = V_2(\delta^*)$, which simplifies to

$$\left[\left(1+\sqrt{c-1}\right)^2(\delta^*)^2 - (c-1)\right](\delta^*-1)^2 - (2\delta^*-1) = 0\,.$$

Then in all three cases we have $f(X \setminus O, C^*) \le f_z(X, C^*) \cdot V_1(\delta^*)$. □

Finally, we present a comparison of the approximation ratios in Theorem 4.1 and Theorem 4.6

Table 4: Approximation ratio for different c for the Robust k-Means objective.

c-value	2	3	4	5	10
Theorem 4.1	14.30	9	7.04	5.98	3.99
Theorem 4.6	9	5.98	4.84	4.21	2.96

B Extensions to k-Median and k-Center

This section analyzes the performance of our proposed methods when applied to the Robust k-Median and Robust k-Center objective. The extension to Robust k-Median is a straightforward adaptation of the proof strategy for Robust k-Means. The purpose of this section is to demonstrate that the proposed framework is not specific to Robust k-Means, and can simultaneously yield non-trivial theoretical guarantees across multiple robust clustering objectives.

B.1 Robust k-Median

Our proofs for Theorem 4.1 and Theorem 4.6 can be directly extended to the Robust k-Median objective, by replacing all the squared distances by unsquared distances in the analysis. Here we present a formal upper bound of applying Algorithm 1b to the Robust k-Median objective, whose proof is similar to Theorem 4.6 and is omitted.

Theorem B.1. *Let $I = (X, k, z)$ be an instance of robust k-Median and C^* be the optimal set of centers. Suppose $\mathcal{A}$ is a β-approximation algorithm for k-Median, and each cluster induced by C^* has size at least cz, then there is an algorithm that calls $\mathcal{A}$ once and computes a set of centers C such that*

$$g_z(X, C) \le \zeta(c)\beta \cdot g_z(X, C^*)\,,$$

where $\zeta(c) = \max\left\{\frac{2c-1+\sqrt{4c+1}}{2(c-1)}, \frac{c+1}{c-1}\right\}$.

In the following table we present the value of $\zeta(c)$ for some values of c.

Table 5: Approximation ratio for different c for the Robust k-Median objective.

c-value	2	3	4	5	10
Theorem B.1	3	2.15	1.85	1.70	1.41

B.2 Robust k-Center

For the k-Center objective, both our methods achieve a 3-approximate reduction. We give a brief justification for Algorithm 1a. Let $r_{\max}$ be the largest radius among the optimal clusters, we have the following two properties.

Observation B.2. *We have $r_x \le 2r_{\max}$ for each true inlier x.*

Proof. $B(x, 2r_{\max})$ already completely contains $\text{cluster}_z(x, C^*)$, which consists of at least $3z$ points. □

Observation B.3. *If $r_o \leq 2r_{\max}$ for some true outlier o, then $d(r_o, C^*) \leq 3r_{\max}$.*

Proof. Since there are only z outliers, $B(o, r_o)$ must contain at least one true inlier x, so if $r_o \leq 2r_{\max}$ then we must have

$$d(o, \text{closest}(x, C^*)) \leq d(o, x) + d(x, \text{closest}(x, C^*)) \leq 2r_{\max} + r_{\max} = 3r_{\max} \,. \qquad \square$$

The analysis for Algorithm 1b is similar.

However, since Charikar et al. [10] already established an optimal 3-approximate algorithm for robust k-Center without assumption on the optimal cluster sizes (they also showed that better-than-3 approximation is NP-hard), we chose not to elaborate on this aspect of our algorithm in the current paper.

C Additional Experimental Results

This section provides more experimental results and plots.

Table 6: Comparison results of robust k-means algorithms on each dataset.

Method	Cost Mean ± Std ↓	Recall ↑	Time(s) ↓	Method	Cost Mean ± Std ↓	Recall ↑	Time(s) ↓
	SKIN-5				**SKIN-10**		
TIKMeans	$(\mathbf{6.151 \pm 0.194}) \times \mathbf{10^4}$	0.742 ± 0.016	3.314 ± 0.495	TIKMeans	$(\mathbf{6.608 \pm 0.131}) \times \mathbf{10^4}$	0.940 ± 0.002	2.965 ± 0.178
IKMeans	$(7.109 \pm 0.885) \times 10^4$	$\mathbf{0.767 \pm 0.020}$	1.444 ± 0.161	IKMeans	$(7.512 \pm 1.133) \times 10^4$	0.950 ± 0.013	1.368 ± 0.060
RobustKmeans++	$(7.620 \pm 0.711) \times 10^4$	0.741 ± 0.033	1.618 ± 0.012	RobustKmeans++	$(8.857 \pm 1.237) \times 10^4$	0.936 ± 0.010	1.617 ± 0.034
NKMeans	$(6.887 \pm 0.646) \times 10^4$	0.744 ± 0.015	4.402 ± 0.105	NKMeans	$(7.057 \pm 0.275) \times 10^4$	0.944 ± 0.016	4.375 ± 0.069
KMeans++	$(7.522 \pm 1.464) \times 10^4$	0.749 ± 0.029	$\mathbf{0.476 \pm 0.037}$	KMeans++	$(1.018 \pm 0.149) \times 10^5$	0.891 ± 0.013	$\mathbf{0.480 \pm 0.039}$
OKMeans (Ours)	$(6.494 \pm 0.312) \times 10^4$	0.757 ± 0.011	2.056 ± 0.052	OKMeans (Ours)	$(6.982 \pm 0.671) \times 10^4$	0.949 ± 0.013	2.045 ± 0.045
OKMeans2 (Ours)	$(6.410 \pm 0.300) \times 10^4$	0.746 ± 0.014	2.395 ± 0.022	OKMeans2 (Ours)	$(6.707 \pm 0.166) \times 10^4$	$\mathbf{0.955 \pm 0.011}$	2.441 ± 0.057
	SUSY-5				**SUSY-10**		
TIKMeans	$(5.165 \pm 0.048) \times 10^7$	0.759 ± 0.035	94.824 ± 1.276	TIKMeans	$(5.272 \pm 0.040) \times 10^7$	0.975 ± 0.002	89.186 ± 4.053
IKMeans	$(5.279 \pm 0.104) \times 10^7$	0.782 ± 0.049	46.489 ± 2.589	IKMeans	$(5.334 \pm 0.087) \times 10^7$	0.976 ± 0.004	42.134 ± 2.536
RobustKmeans++	$(5.791 \pm 0.132) \times 10^7$	0.708 ± 0.072	74.735 ± 2.058	RobustKmeans++	$(5.966 \pm 0.130) \times 10^7$	0.973 ± 0.006	74.092 ± 1.903
NKMeans	$(4.825 \pm 0.022) \times 10^7$	0.739 ± 0.014	17.346 ± 0.281	NKMeans	$(4.941 \pm 0.028) \times 10^7$	0.972 ± 0.002	17.529 ± 0.456
KMeans++	$(4.885 \pm 0.087) \times 10^7$	$\mathbf{0.831 \pm 0.012}$	$\mathbf{9.725 \pm 0.111}$	KMeans++	$(5.173 \pm 0.235) \times 10^7$	0.980 ± 0.004	$\mathbf{9.728 \pm 0.152}$
OKMeans (Ours)	$(4.830 \pm 0.064) \times 10^7$	0.799 ± 0.020	12.840 ± 0.353	OKMeans (Ours)	$(\mathbf{4.873 \pm 0.023}) \times \mathbf{10^7}$	0.982 ± 0.002	12.919 ± 0.432
OKMeans2 (Ours)	$(\mathbf{4.801 \pm 0.034}) \times \mathbf{10^7}$	0.798 ± 0.018	13.469 ± 0.450	OKMeans2 (Ours)	$(4.876 \pm 0.020) \times 10^7$	$\mathbf{0.982 \pm 0.002}$	13.407 ± 0.286
	SHUTTLE				**KDDFULL**		
TIKMeans	$(\mathbf{6.122 \pm 0.260}) \times \mathbf{10^4}$	0.141 ± 0.142	0.543 ± 0.008	TIKMeans	$(\mathbf{3.320 \pm 0.002}) \times \mathbf{10^7}$	0.601 ± 0.003	49.850 ± 1.626
IKMeans	$(7.277 \pm 0.503) \times 10^4$	0.141 ± 0.142	$\mathbf{0.243 \pm 0.003}$	IKMeans	$(3.937 \pm 0.861) \times 10^7$	0.615 ± 0.022	21.419 ± 0.834
RobustKmeans++	$(7.830 \pm 1.193) \times 10^4$	0.094 ± 0.121	0.363 ± 0.011	RobustKmeans++	$(4.845 \pm 1.260) \times 10^7$	0.611 ± 0.040	40.355 ± 1.482
NKMeans	$(7.260 \pm 0.353) \times 10^4$	$\mathbf{0.171 \pm 0.061}$	4.785 ± 0.095	NKMeans	$(3.466 \pm 0.178) \times 10^7$	$\mathbf{0.624 \pm 0.013}$	14.859 ± 0.127
KMeans++	$(7.055 \pm 0.705) \times 10^4$	0.082 ± 0.127	0.294 ± 0.082	KMeans++	$(6.123 \pm 2.152) \times 10^7$	0.553 ± 0.113	$\mathbf{6.894 \pm 0.104}$
OKMeans (Ours)	$(6.823 \pm 0.625) \times 10^4$	0.171 ± 0.116	0.785 ± 0.074	OKMeans (Ours)	$(4.019 \pm 0.770) \times 10^7$	0.616 ± 0.010	9.510 ± 0.062
OKMeans2 (Ours)	$(6.823 \pm 0.625) \times 10^4$	0.171 ± 0.116	0.776 ± 0.058	OKMeans2 (Ours)	$(4.250 \pm 0.919) \times 10^7$	0.613 ± 0.013	10.269 ± 0.574

Table 7: Performance of the KNN heuristic with constant K.

Dataset	Metric	KNN ($K = 2$)	KNN ($K = 5$)	KNN ($K = 10$)	OKMeans
SKIN-5	Cost	$(6.290 \pm 0.275) \times 10^4$	$(6.822 \pm 0.417) \times 10^4$	$(6.577 \pm 0.214) \times 10^4$	$(6.494 \pm 0.312) \times 10^4$
	Recall	0.760 ± 0.008	0.758 ± 0.016	0.758 ± 0.012	0.757 ± 0.011
	Time (s)	1.907 ± 0.178	1.859 ± 0.130	1.929 ± 0.139	2.056 ± 0.052
SKIN-10	Cost	$(6.838 \pm 0.385) \times 10^4$	$(6.911 \pm 0.585) \times 10^4$	$(6.773 \pm 0.449) \times 10^4$	$(6.982 \pm 0.671) \times 10^4$
	Recall	0.953 ± 0.014	0.950 ± 0.016	0.953 ± 0.012	0.949 ± 0.013
	Time (s)	1.888 ± 0.151	1.864 ± 0.149	1.927 ± 0.175	2.045 ± 0.045
SUSY-5	Cost	$(4.815 \pm 0.024) \times 10^7$	$(4.804 \pm 0.022) \times 10^7$	$(4.790 \pm 0.030) \times 10^7$	$(4.830 \pm 0.064) \times 10^7$
	Recall	0.832 ± 0.009	0.826 ± 0.011	0.824 ± 0.010	0.799 ± 0.020
	Time (s)	11.935 ± 0.203	11.867 ± 0.272	11.892 ± 0.213	12.840 ± 0.353
SUSY-10	Cost	$(4.890 \pm 0.039) \times 10^7$	$(4.891 \pm 0.065) \times 10^7$	$(4.897 \pm 0.062) \times 10^7$	$(4.873 \pm 0.023) \times 10^7$
	Recall	0.982 ± 0.002	0.982 ± 0.002	0.982 ± 0.002	0.982 ± 0.002
	Time (s)	11.805 ± 0.242	11.968 ± 0.231	11.841 ± 0.262	12.919 ± 0.432
SHUTTLE	Cost	$(6.717 \pm 0.679) \times 10^4$	$(7.152 \pm 0.459) \times 10^4$	$(6.711 \pm 0.707) \times 10^4$	$(6.823 \pm 0.625) \times 10^4$
	Recall	0.124 ± 0.116	0.171 ± 0.113	0.147 ± 0.124	0.171 ± 0.116
	Time (s)	1.769 ± 0.153	1.752 ± 0.172	1.741 ± 0.139	0.785 ± 0.074
KDDFULL	Cost	$(4.612 \pm 1.244) \times 10^7$	$(4.371 \pm 1.087) \times 10^7$	$(4.631 \pm 1.366) \times 10^7$	$(4.019 \pm 0.770) \times 10^7$
	Recall	0.560 ± 0.089	0.521 ± 0.120	0.569 ± 0.074	0.616 ± 0.010
	Time (s)	9.141 ± 0.568	9.273 ± 0.458	9.104 ± 0.605	9.510 ± 0.062

Table 8: Parameter sensitivity analysis of OKMeans and OKMeans2 under different values of c.

Dataset	Method	$c = 3$	$c = 4$	$c = 5$	$c = 7$	$c = 10$	$c = 15$	$c = 20$
SKIN-5	OKMeans	6.688×10^4	6.479×10^4	$\underline{6.347 \times 10^4}$	6.605×10^4	6.613×10^4	$\mathbf{6.257 \times 10^4}$	6.736×10^4
	OKMeans2	6.670×10^4	$\mathbf{6.367 \times 10^4}$	6.624×10^4	$\underline{6.533 \times 10^4}$	6.544×10^4	6.691×10^4	6.708×10^4
SKIN-10	OKMeans	7.053×10^4	7.013×10^4	$\underline{6.821 \times 10^4}$	$\mathbf{6.778 \times 10^4}$	6.963×10^4	7.144×10^4	6.836×10^4
	OKMeans2	6.925×10^4	$\mathbf{6.682 \times 10^4}$	$\underline{6.824 \times 10^4}$	6.980×10^4	7.006×10^4	7.114×10^4	6.996×10^4
SUSY-5	OKMeans	$\underline{4.830 \times 10^7}$	4.861×10^7	$\mathbf{4.814 \times 10^7}$	4.843×10^7	4.845×10^7	4.847×10^7	4.864×10^7
	OKMeans2	$\mathbf{4.801 \times 10^7}$	4.863×10^7	$\underline{4.823 \times 10^7}$	4.849×10^7	4.823×10^7	4.841×10^7	4.830×10^7
SUSY-10	OKMeans	4.873×10^7	4.868×10^7	4.868×10^7	$\mathbf{4.865 \times 10^7}$	$\underline{4.868 \times 10^7}$	4.891×10^7	4.882×10^7
	OKMeans2	4.876×10^7	4.873×10^7	$\underline{4.866 \times 10^7}$	4.867×10^7	$\mathbf{4.866 \times 10^7}$	4.880×10^7	4.888×10^7
SHUTTLE	OKMeans	$\mathbf{6.823 \times 10^4}$	$\underline{6.823 \times 10^4}$	6.823×10^4	6.823×10^4	6.889×10^4	6.998×10^4	6.998×10^4
	OKMeans2	$\mathbf{6.823 \times 10^4}$	$\underline{6.823 \times 10^4}$	6.823×10^4	6.823×10^4	6.823×10^4	6.823×10^4	6.998×10^4
KDDFULL	OKMeans	4.019×10^7	4.053×10^7	4.395×10^7	$\mathbf{3.586 \times 10^7}$	4.143×10^7	4.251×10^7	$\underline{3.640 \times 10^7}$
	OKMeans2	4.250×10^7	4.082×10^7	3.988×10^7	4.302×10^7	$\underline{3.967 \times 10^7}$	$\mathbf{3.770 \times 10^7}$	4.087×10^7

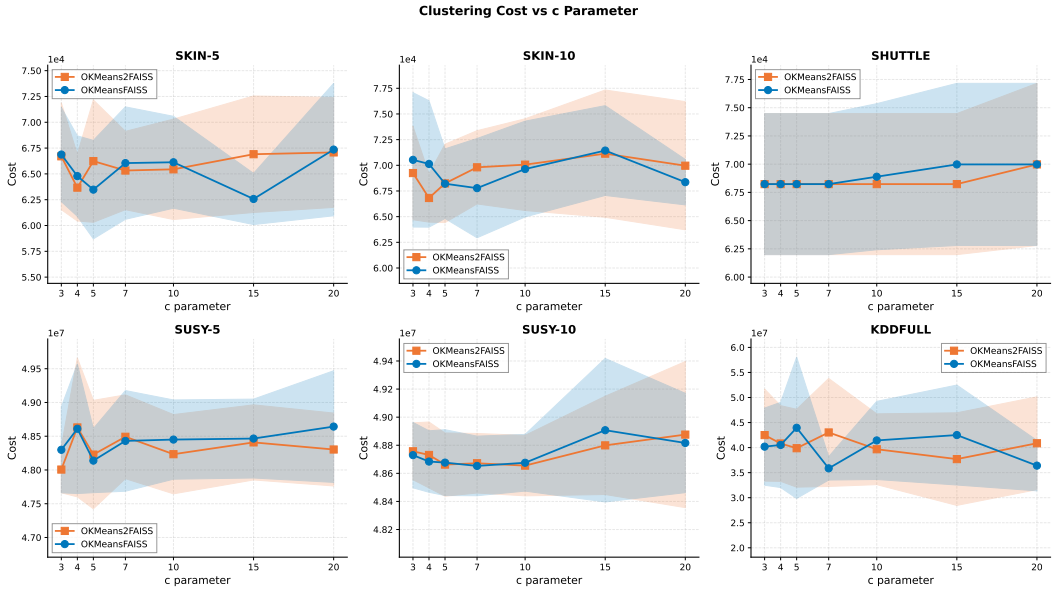


Figure 1: Parameter Sensitivity Cost Comparison

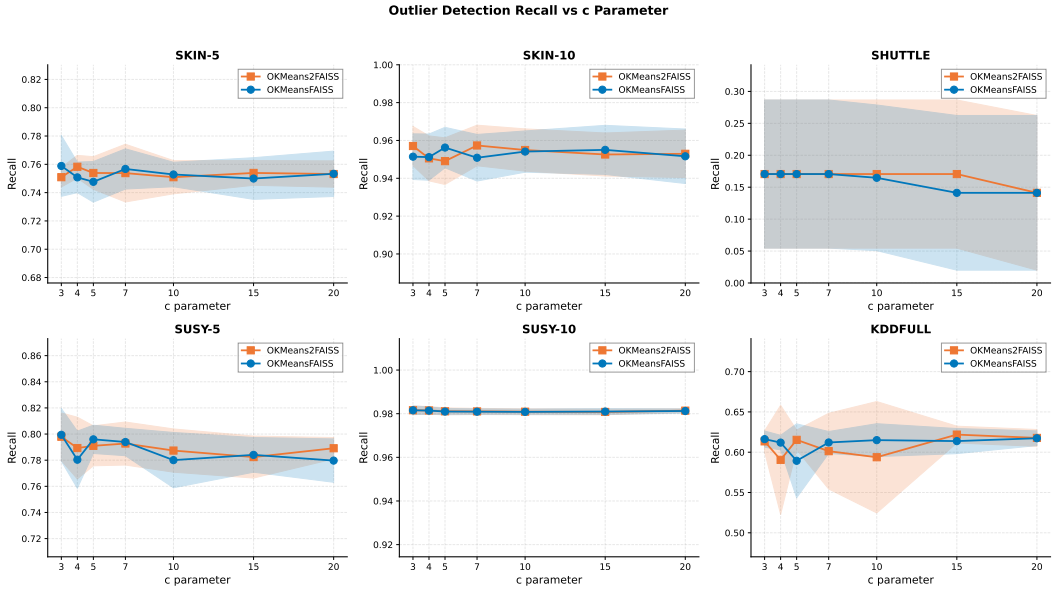


Figure 2: Parameter Sensitivity Recall Comparison

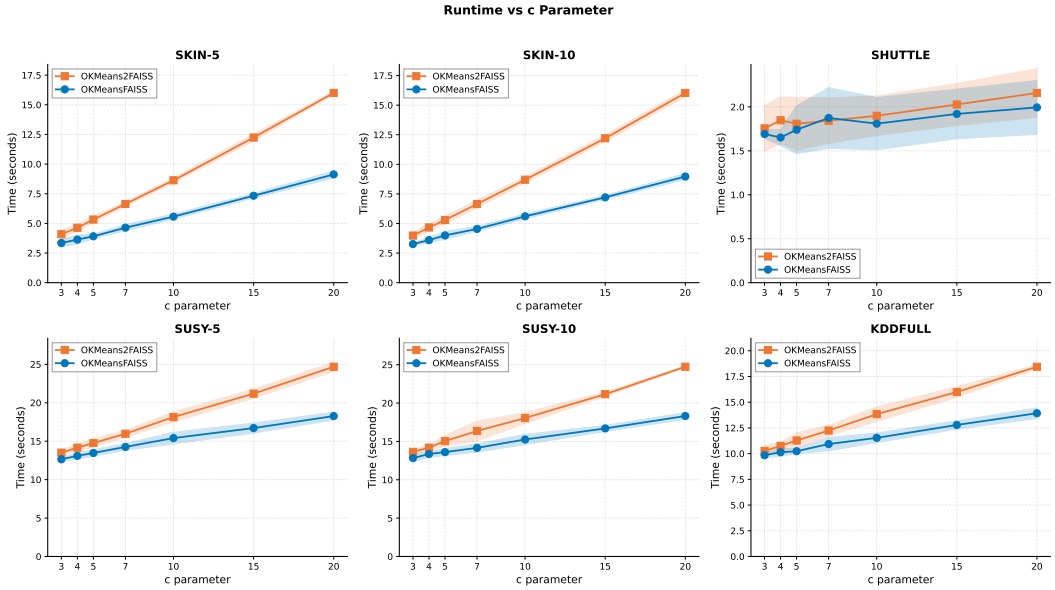


Figure 3: Parameter Sensitivity Time Comparison

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 are aligned with the actual results and supported by rigorous theoretical analysis or experiments in the paper.

   Guidelines:

   - The answer [N/A] 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 [N/A] 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 and future work are included in Section 6.

   Guidelines:

   - The answer [N/A] 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: We provide the full assumptions, theory, and proofs in Section 4 and Appendix A.

Guidelines:

- The answer [N/A] 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: We provide the detailed experiment setup in Section 5.

   Guidelines:

   - The answer [N/A] 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 dataset and algorithms we used are open-source.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.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://neurips.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) necessary to understand the results?

   Answer: [Yes]

   Justification: We provide detailed experiment settings in Section 5.

   Guidelines:

   - The answer [N/A] 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: We report error bars in our Tables 2 and 6 to 8.

   Guidelines:

   - The answer [N/A] 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: We explained the computational resources we used in Section 5.

   Guidelines:

   - The answer [N/A] 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: Research conducted in the paper conforms with NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] 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: [N/A]

    Justification: Not applicable to our research, as our research is mainly theoretical.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] 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., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: Not applicable to our research, as our research is mainly theoretical.

    Guidelines:

    - The answer [N/A] 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: [Yes]

    Justification: We originally implemented our method and cited their papers.

    Guidelines:

    - The answer [N/A] 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: [Yes]

    Justification: We introduced in the Section 5 and open-source our codes.

    Guidelines:

    - The answer [N/A] 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: [N/A]

    Justification: We are not related to any human subjects in our work.

    Guidelines:

    - The answer [N/A] 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: [N/A]

    Justification: We are not related to any human subjects in our work.

    Guidelines:

    - The answer [N/A] 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 rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: We did not use LLMs for the development of our algorithmic methodology.

Guidelines:

- The answer [N/A] 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 in the NeurIPS handbook for what should or should not be described.