Near-Linear Time Approximation Algorithms for k-means with Outliers

Junyu Huang [1 2] **Qilong Feng** [1 2] **Ziyun Huang** [3] **Jinhui Xu** [4] **Jianxin Wang** [1 2 5]

Abstract

The k-means with outliers problem is one of the most extensively studied clustering problems in the field of machine learning, where the goal is to discard up to z outliers and identify a minimum k-means clustering on the remaining data points. Most previous results for this problem have running time dependent on the aspect ratio Δ (the ratio between the maximum and the minimum pairwise distances) to achieve fast approximations. To address the issue of aspect ratio dependency on the running time, we propose sampling-based algorithms with almost linear running time in the data size, where a crucial component of our approach is an algorithm called Fast-Sampling. Fast-Sampling algorithm can find inliers that well approximate the optimal clustering centers without relying on a guess for the optimal clustering costs, where a 4-approximate solution can be obtained in time $O(\frac{ndk \log\log n}{\epsilon^2})$ with $O(\frac{k}{\epsilon})$ centers opened and $(1+\epsilon)z$ outliers discarded. To reduce the number of centers opened, we propose a center reduction algorithm, where an $O(\frac{1}{\epsilon})$-approximate solution can be obtained in time $O(\frac{ndk \log\log n}{\epsilon^2} + dpoly(k, \frac{1}{\epsilon})\log(n\Delta))$ with $(1+\epsilon)z$ outliers discarded and exactly k centers opened. Empirical experiments suggest that our proposed sampling-based algorithms outperform state-of-the-art algorithms for the k-means with outliers problem.

[1]School of Computer Science and Engineering, Central South University, Changsha 410083, China [2]Xiangjiang Laboratory, Changsha 410205, China [3]Department of Computer Science and Software Engineering, Penn State Erie, The Behrend College [4]Department of Computer Science and Engineering, State University of New York at Buffalo, NY, USA [5]The Hunan Provincial Key Lab of Bioinformatics, Central South University, Changsha 410083, China. Correspondence to: Qilong Feng <csufeng@mail.csu.edu.cn>, Jianxin Wang <jxwang@mail.csu.edu.cn>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

1. Introduction

Clustering is a fundamental unsupervised learning problem that has been extensively studied over the past decades. Among various mathematical characterizations, the k-means clustering is one of the most popular formulations, with numerous applications in decision-making and data analysis. However, a recurring problem for the k-means clustering is how to handle the data noise. This issue arises because the performance of the k-means clustering is well known to be highly sensitive to outliers, as pointed out in the literature (Deshpande et al., 2020; Gupta et al., 2017; Im et al., 2020). Charikar et al. (Charikar et al., 2001) formulated the k-means with outliers problem, in which a number z of data points can be discarded as outliers when minimizing the clustering cost of the given instances.

The k-means with outliers. Given a dataset $X \subset \mathbb{R}^d$ of size n, a number of clusters k, and a number of outliers z, the goal of the problem is to find a set $C \subset \mathbb{R}^d$ of k centers and partition X into X_{in} and X_{out} such that $|X_{in}| \geq n - z$, and the clustering cost of X_{in} with respect to C is minimized.

The k-means with outliers problem is much more challenging than the standard k-means problem due to the additional task of identifying z outliers. The only two constant approximation schemes are based on complex linear programming rounding techniques with high-order polynomial running time (Chen, 2008; Krishnaswamy et al., 2018), which are difficult to implement in practice. Furthermore, as pointed out in the literature (Grunau & Rozhoň, 2022), any constant approximate solution with exactly z outliers discarded requires a running time of $\Omega(z^2)$. In the case when there is heavy noise in the datasets, i.e., $z = \Omega(n)$, the running time becomes quadratic, which may limit the scalability of the algorithms for handling large-scale datasets.

On the practical side, developing simple and fast approximation schemes for the k-means with outliers problem has attracted much attention in recent years. By relaxing the number of outliers discarded or the number of centers opened, there is a line of research aiming at designing fast and practical algorithms with good theoretical guarantees. Charikar et al. (Charikar et al., 2001) presented a polynomial-time $O(1/\epsilon)$-approximation algorithm using a reduction-based method, where $(1+\epsilon)z$ outliers are discarded to guarantee the approximation loss. Gupta et al. (Gupta et al., 2017)

proposed a local search algorithm that gives a $(274+\epsilon)$-approximate solution with $O(zk\log(n\Delta))$ outliers discarded in time $O(\frac{d}{\epsilon}k^2n^2\log(n\Delta))$, where Δ is the aspect ratio of the given instance (the ratio between the maximum and the minimum pairwise distances). By using k-means++ (Arthur & Vassilvitskii, 2007) to construct a weighted $O(1)$-approximation coreset, the running time of their algorithm can be improved to $O(\frac{d}{\epsilon}k^2(k+z)^2\log(n\Delta)+nz)$. Zhang et al. (Zhang et al., 2021) showed that a combination of Lagrange relaxation and local search methods gives an $O(1)$-approximate solution with $O(z)$ outliers discarded in time $O(n(k+z)(d+\frac{1}{\epsilon}k^2\log(n\Delta)))$. Grunau and Rozhoň (Grunau & Rozhoň, 2022) proposed an improved reduction-based local search method, which gives an $O(1/\epsilon)$-approximation by discarding $(1+\epsilon)z$ outliers in time $O(\frac{d}{\epsilon}nk\log k(\log\log k+\frac{1}{\epsilon}\log\frac{1}{\epsilon})\log(n\Delta))$. By using Metropolis-Hastings technique, the running time of (Grunau & Rozhoň, 2022) can be further improved to $\tilde{O}(\frac{ndk^2}{z})$ [1] with the assumption that $\Delta = poly(n)$, where an $O(1)$-approximate solution can be obtained with $O(z)$ outliers discarded. Bhaskara et al. (Bhaskara et al., 2019) presented a modified k-means++ sampling method which yields an $O(\log k)$-approximation with $O(z\log k)$ outliers discarded. The algorithm requires a guess for the optimal clustering cost to trim the distances of data points to the centers opened, ensuring that outliers are sampled with smaller probabilities. With this technique, approximation results can be obtained in time $O(ndk\log(n\Delta))$. Under the assumption that each optimal cluster has size $\Omega(z)$, Im et al. (Im et al., 2020) proposed a density-based method that can filter most data points far away from the optimal clustering centers. Then, by calling a standard k-means approximation algorithm on the remaining data points, an $O(1)$-approximate solution with $2z$ outliers discarded can be obtained. However, the filtering process also requires a guess for the optimal clustering cost, which results in a running time of $O(n^2\log(n\Delta))$.

As pointed out in the literature (Grunau & Rozhoň, 2022), for the k-means with outliers problem, $\Omega(nk^2/z)$ is the lower bound of running time for achieving a constant approximation with $O(z)$ outliers discarded in the general metric space query model. Thus, near-linear time approximation can only be achieved when $O(z)$ outliers are allowed to be discarded. To obtain linear/sub-linear time approximation, certain data distribution assumptions must be introduced. In (Deshpande et al., 2020), the number of outliers z is assumed to be $\Theta(\delta n)$, such that a solution with $O(k/\delta)$ centers opened and $(1+\delta)z$ outliers discarded can be obtained in time $O(ndk/\delta)$. In (Ding & Huang, 2021), to design a sub-linear time framework, it was assumed that both the number of outliers z and the size of each optimal cluster should be $\Omega(n/k)$, where a solution with $k+O(\log k)$ centers opened can be obtained, though with a non-constant approximation on clustering quality. Consequently, it is challenging to design fast approximation schemes without any data distribution assumptions.

In this paper, we aim to present fast sampling-based algorithms without any data distribution assumptions. Previously, a number of sampling-based methods have been proposed. For non-uniform sampling methods, most algorithms (Bhaskara et al., 2019; Grunau & Rozhoň, 2022) have running time dependent on the aspect ratio Δ, since a guess for the optimal clustering cost is required to trim the distances between data points and their centers. In (Bhaskara et al., 2019; Charikar et al., 2001; Grunau & Rozhoň, 2022; Zhang et al., 2021), Δ is assumed to be polynomially bounded. In (Cohen-Addad, 2020), a more general case was considered, where Δ can be as large as $2^{n^{o(1)}}$. However, in the worst case, Δ can be arbitrarily large as pointed out in (Nguyen et al., 2022), which may constrain the scalability of the algorithms with running time dependent on Δ. In (Deshpande et al., 2020), a 5-approximate solution can be obtained by opening $O(k/\delta)$ centers and discarding $O(1+\delta)z$ outliers using a sampling-based method, where $\delta=\Theta(z/n)$. For the case when outliers take a very tiny fraction of the dataset, the algorithm may open $O(n^2/z)$ centers with quadratic running time. For uniform sampling methods, although approximation schemes with sub-linear running time have been presented in (Ding & Huang, 2021), the theoretical bounds are guaranteed under strict assumptions. Other aforementioned practical approximation algorithms, such as those based on local search or radius query techniques (Charikar et al., 2001; Im et al., 2020; Zhang et al., 2021), also depend on a guess for the optimal clustering cost, making them less competitive compared to sampling-based algorithms.

To further improve the running time and approximation guarantee, our main objective is to design fast algorithms with $(1+\epsilon)z$ outliers discarded. From a theoretical perspective, an algorithm that discards an additional ϵz outliers is the best we can hope when designing approximation algorithms with near-linear running time in the data size, according to the lower bounds given in (Grunau & Rozhoň, 2022). On the other hand, the number of additional outliers discarded can be arbitrarily small by adjusting the value of ϵ, which is acceptable from a practical perspective. To obtain fast approximation results with $(1+\epsilon)$z outliers discarded and near-linear running time in the data size, we first propose a sampling-based algorithm called Fast-Sampling. Fast-Sampling can avoid guessing the optimal clustering cost during the sampling process and achieve a good approximation with almost linear running time in the data size.

To address the aspect ratio dependency issue on the running time, a two-stage sampling strategy is proposed, which mainly consists of probability boosting and probability normalization stages. In probability boosting stage, an oversam-

[1] $\tilde{O}(.)$ hides polylogarithmic factors in n

Table 1. Comparison results for the k-means with outliers problem.

Results without Assumptions	Approximation	Centers Opened	Outliers Discarded	Assumption	Running Time
(Chen, 2008)	$O(1)$	k	z	-	$d \cdot poly(n,k)$
(Krishnaswamy et al., 2018)	$53.002+\epsilon$	k	z	-	$n^{O(\frac{1}{\epsilon^3})}d$
(Friggstad et al., 2019)	$1+\epsilon$	$k(1+\epsilon)$	z	doubling metrics	$poly(n,d,k)$
(Charikar et al., 2001)	$O(\frac{1}{\epsilon})$	k	$(1+\epsilon)z$	-	$d \cdot poly(n,k)$
(Gupta et al., 2017)	$274+\epsilon$	k	$O(zk\log(n\Delta))$	-	$O(\frac{d}{\epsilon}k^2n^2\log(n\Delta))$
(Zhang et al., 2021)	$O(1)$	k	$O(z)$	-	$O(n(k+z)(d+\frac{1}{\epsilon}k^2\log(n\Delta)))$
(Grunau & Rozhoň, 2022)	$O(\frac{1}{\epsilon})$	k	$(1+\epsilon)z$	-	$O(\frac{d}{\epsilon}nk\log k(\log\log k+\frac{1}{\epsilon}\log\frac{1}{\epsilon})\log(n\Delta))$
(Bhaskara et al., 2019)	$O(\log k)$	k	$O(zlogk)$	-	$O(ndk\log(n\Delta))$
	$64+\epsilon$	$(1+c)k$	$\frac{(1+c)(1+\epsilon)z}{c(1-\mu)}$		$O(ndk^{\frac{\log(n\Delta)}{\epsilon}})$
This Paper	4	$O(\frac{k}{\epsilon})$	$(1+\epsilon)z$	-	$O(\frac{ndk\log\log n}{\epsilon^2})$
	$O(\frac{1}{\epsilon})$	k	$(1+\epsilon)z$		$O(\frac{ndk\log\log n}{\epsilon^2}+dpoly(k,\frac{1}{\epsilon})\log(n\Delta))$
Results with Assumptions	**Approximation**	**Centers Opened**	**Outliers Discarded**	**Assumption**	**Running Time**
(Deshpande et al., 2020)	5	$O(\frac{kn}{z})$	$O(z)$	$z=\Theta(n)$	$O(ndk)$
(Im et al., 2020)	$O(1)$	k	$2z$	$\lvert P_j^*\rvert \geq 3z$	$O((n^2d+T(n))\log(n\Delta))$
(Ding & Huang, 2021)	$O(1)+O(L^2)$	$k+O(\log k)$	z	$\lvert P_j^*\rvert=\Omega(\frac{n}{k}),\ z=\Omega(\frac{n}{k})$	$d \cdot poly(k)$
	$O(1)+O(L^2)$	k	z		$d \cdot poly(k)$
This Paper	4	$O(k)$	z	$\lvert P_j^*\rvert \geq 3z$	$O(ndk\log\log n)$

pling factor is found to boost the distance-based sampling probabilities, enabling independent sampling of $\Theta((1+\epsilon)z)$ points in expectation. This guarantees that inliers can take a certain fraction (i.e., $\frac{\epsilon}{1+\epsilon}$) of the sampled data points. Then, in the second stage, a probability normalization step is used to guarantee that exactly one inlier that is close enough to some optimal clustering center can be picked with certain probability. By repeating the sampling process $O(k/\epsilon)$ rounds, a 4-approximate solution with $O(k/\epsilon)$ centers opened and $(1+\epsilon)z$ outliers discarded can be obtained. To further reduce the number of centers opened, we propose an algorithm called Center-Reduction, which can find most mistakenly discarded inliers back during the sampling process. With this technique, an $O(1/\epsilon)$-approximate solution can be obtained by opening exactly k centers and discarding $(1+\epsilon)z$ outliers. The experimental results demonstrate that our proposed sampling-based algorithms can achieve significant improvements on both the running time and clustering quality compared with other algorithms for the k-means with outliers problem. Putting all these together, we have the following results.

Theorem 1.1. *For the k-means with outliers problem, there exists an algorithm that runs in time $O(\frac{ndk\log\log n}{\epsilon^2})$ and outputs a 4-approximate solution with constant probability by opening $O(\frac{k}{\epsilon})$ centers and discarding $(1+\epsilon)z$ outliers.*

Theorem 1.2. *For the k-means with outliers problem, there exists an algorithm that runs in time $O(\frac{ndk\log\log n}{\epsilon^2}+dpoly(k,\frac{1}{\epsilon})\log(n\Delta))$ and outputs an $O(\frac{1}{\epsilon})$-approximate solution with constant probability by opening exactly k centers and discarding $(1+\epsilon)z$ outliers.*

Following the assumption in (Im et al., 2020) that each optimal cluster has size at least $3z$, a 4-approximate solution with $O(k)$ centers opened and exactly z outliers discarded can be obtained using our proposed Fast-Sampling method in almost linear running time in the data size.

Theorem 1.3. *For the k-means with outliers problem, under the assumption that each optimal cluster has size at least $3z$, there exists an algorithm that runs in time $O(ndk\log\log n)$ and outputs a 4-approximate solution with constant probability by opening $O(k)$ centers and discarding z outliers.*

Table 1 shows a detailed comparison of the results for the k-means with outliers problem, where n is the data size, d is the dimension, k is the number of clusters, Δ is the aspect ratio, $T(n)$ is the running time of any constant approximation algorithm for the standard k-means problem, L is the maximum diameter of all optimal clusters, and ϵ is the parameter to control the approximation guarantee. It can be seen that linear/sub-linear time approximation with running time independent of Δ can only be obtained under certain data distribution assumptions (Ding & Huang, 2021; Deshpande et al., 2020). It is worth mentioning that although the result of (Deshpande et al., 2020) has linear running time if $z=\Theta(n)$, the running time will deteriorate to $O(n^2dk)$ when z is a constant. For approximation without any assumptions, several near-linear time results can be obtained with running time independent of z when Δ can be bounded by $poly(n)$ (Bhaskara et al., 2019; Grunau & Rozhoň, 2022). Compared with these results, our sampling-based method is the first one that can achieve a running time independent of Δ. The current best result (Grunau & Rozhoň, 2022) is an $O(1/\epsilon)$-approximation with exactly k centers opened and $(1+\epsilon)z$ outliers discarded. Compared with the results given in (Grunau & Rozhoň, 2022), our proposed method can achieve the same theoretical guarantee on clustering quality and the number of outliers discarded, i.e., an $O(1/\epsilon)$-approximation with k centers opened and $(1+\epsilon)z$ outliers discarded . Moreover, even under the case (Cohen-Addad et al., 2022) when $\Delta=2^{n^{o(1)}}$, the running time of our method is better than that in (Grunau & Rozhoň, 2022). In general, when Δ is larger than $\Omega(\log n)$, our

method is better than that in (Grunau & Rozhoň, 2022). In real-world applications, Δ is usually assumed to be bounded by $poly(n)$, where our method is still much better than that in (Grunau & Rozhoň, 2022). By relaxing the number of centers opened, our proposed algorithm can achieve the best constant approximation on clustering quality with only $(1+\epsilon)z$ outliers discarded in almost linear time in the data size, compared with other algorithms with running time dependent on Δ (Bhaskara et al., 2019).

Due to space limit, we leave the discussions on the running time lower bounds in Appendix A.

2. Preliminaries

We use $X \subset \mathbb{R}^d$ and k to denote the given dataset and the number of clusters, respectively. Let z be the number of outliers. We use $C^* = \{c_1^*, c_2^*, ..., c_k^*\}$ and $\mathcal{P}(C^*) = \{P_1^*, P_2^*, ..., P_k^*\}$ to denote an optimal solution and its clustering partition, respectively. For any two points $p, q \in \mathbb{R}^d$, let $d(p,q)$ denote their squared distance. Given a point $p \in \mathbb{R}^d$ and a set $C \subset \mathbb{R}^d$ of centers, let $d(p,C) = \min_{c_i \in C} d(p,c_i)$ be the squared distance from p to its closest center in C. We define the clustering cost of X with respect to C as $d(X,C) = \sum_{x \in X} d(x,C)$. Let OPT be the optimal clustering cost. For each $P_j^* \in \mathcal{P}(C^*)$, let $OPT_j = d(P_j^*, \{c_j^*\})$ be the cost of data points in P_j^* to c_j^*. We use Z^* to denote the set of true outliers, which is the set of the farthest z data points from X to C^*. Let $N = X \backslash Z^*$ be the set of true inliers. For any optimal cluster P_j^*, define $T_\alpha(P_j^*) = \{p \in P_j^* : d(p,c_j^*) < \alpha d\left(P_j^*, \{c_j^*\}\right)/|P_j^*|\}$ as the set of data points in P_j^* close to c_j^*, where α is a constant with $\alpha > 1$. Given a set C of centers and a non-negative real number l, for each $x \in X$, let $t(l,x,C) = \min\{ld(x,C)/d(X,C), 1\}$ be the trimmed distance-based sampling probability by l and C. The following lemma is a folklore result for the k-means problem.

Lemma 2.1. (Aggarwal et al., 2009) Let $P_j^* \in \mathcal{P}(C^*)$ be an optimal cluster, and c_j^* be its center. For any data point $s \in \mathbb{R}^d$, we have $d(P_j^*, \{s\}) = d(P_j^*, \{c_j^*\}) + |P_j^*| d(s, c_j^*)$.

3. Almost Linear Time Algorithms for k-means with Outliers

The general idea of our proposed sampling-based algorithms is to find inliers that can well approximate the optimal clustering centers. Since outliers are usually far away from the optimal clustering centers, distance-based sampling strategies (such as the D^2-Sampling strategy (Arthur & Vassilvitskii, 2007)) are prone to directly pick outliers as centers, which may deteriorate the theoretical guarantees on clustering quality if the number of centers to be opened is required to be roughly bounded by $O(k)$. To address this issue, our approach aims to increase the distance-based sampling probabilities for inliers while decreasing those for outliers. By enhancing the summation of distance-based sampling probabilities to at least $(1+\epsilon)z$, we ensure that outliers only contribute a tiny fraction of the overall probability summation. Thus, with a normalization step, a linear relationship of distance-based sampling distribution can be maintained, while exactly one inlier close enough to the optimal clustering centers can be picked with certain probability in each sampling iteration. Finally, the sampled inliers can be used for constructing a weighted instance for final clustering.

3.1. The Fast-Sampling Algorithm

In this section, we present a 4-approximation algorithm with $O(k/\epsilon)$ centers opened and $(1+\epsilon)z$ outliers discarded, which can avoid guessing the optimal clustering cost while choosing good inliers as clustering centers during the sampling process. The Fast-Sampling algorithm is described in Algorithm 1, where an intuitive idea is to use a two-stage sampling strategy to sample inliers that are close to the optimal clustering centers. In the first stage (steps 3-7 of Algorithm 1), an oversampling factor is used to boost the distance-based sampling probabilities, enabling an independent sampling of $\Theta((1+\epsilon)z)$ data points in expectation. This guarantees that inliers account for a significant proportion (at least $\frac{\epsilon}{1+\epsilon}$) of the sampled data points. Then, in the second stage (step 8 of Algorithm 1), a probability normalization step is used to guarantee that exactly one inlier well approximating an optimal clustering center can be picked with certain probability. Due to space limit, all the proofs are given in Appendix B.

In Algorithm 1, a key stage is to find an oversampling factor to boost the distance-based sampling probabilities, which is given in steps 3-7 of Algorithm 1. Let $R = (1+\epsilon)z$ be the probability summation threshold. Denote C_i as the set of centers obtained before executing the i-th iteration of step 8 in Algorithm 1. The true oversampling factor l_{true}^i is defined as the non-negative real number such that $\sum_{x \in X} t(l_{true}^i, x, C_i) = R$. We first show that an accurate estimation for l_{true}^i can be obtained in time $O(\frac{nd \log\log n}{\epsilon})$.

For each data point $x \in X$, we use $t(l_{true}^i, x, C_i) = \min\{l_{true}^i d(x,C_i)/d(X,C_i), 1\}$ to denote the trimmed probability of x by l_{true}^i and C_i. The exact value for l_{true}^i can be obtained trivially by sorting the distances between data points in X to C_i in non-increasing order, and then iteratively guessing the points in X whose boosted probabilities by l_{true}^i and C_i exceed 1. However, the distance calculation and sorting process take time $O(nd \log n)$ in each iteration. To further accelerate the oversampling factor finding process, a fast estimation method is proposed (Algorithm 2). In the i-th iteration of Algorithm 1, we denote $T_i = \{x \in X : t(l_{true}^i, x, C_i) = 1\}$ as the set of data points in X whose boosted probabilities by l_{true}^i and C_i are larger

than 1. In the j-th iteration of Algorithm 2, starting from step 3 in Algorithm 2, it iteratively tries to guess the furthest ϵz data points from X to C_i as the set of data points in X whose boosted probabilities by l^i_{true} and C_i are larger than 1. Then, an estimation factor l^j_f can be obtained in each step 5 of Algorithm 2.

Algorithm 1 Fast-Sampling($X, k, z, d, \eta, \epsilon$)

Input: An instance (X, k, z, d) of the k-means with outliers, parameters η and ϵ.
Output: A set $C \subset \mathbb{R}^d$.

1: Randomly and uniformly sample a point x_0 from X, and initialize $C = \{x_0\}$.
2: **for** $i = 1$ to $O(\frac{k}{\epsilon}\log\frac{1}{\eta})$ **do**
3: l_f = OSE$(X, k, z, d, \epsilon, (1+\epsilon)z, C)$.
4: $l_{min} = l_f$, $l_{max} = \max\{2l_f, \epsilon z l_f\}$.
5: $\mathcal{S} = \{(1+\epsilon)^j : j \in \mathbb{Z}, l_{min} \le (1+\epsilon)^j \le l_{max}\}$.
6: $\mathcal{S} = \mathcal{S} \cup \{l_{min}\} \cup \{l_{max}\}$.
7: Use binary search to find a $l' \in \mathcal{S}$ s.t. $(1+\epsilon)z \le \sum_{x\in X} t(l', x, C) \le (1+\epsilon)^2 z$.
8: Sample a data point $x \in X$ with probability $\frac{t(l',x,C)}{\sum_{x\in X} t(l',x,C)}$, and add it to C.
9: **end for**
10: **return** C.

Algorithm 2 OSE($X, k, z, d, \epsilon, R, C$)

Input: An instance (X, k, z, d) of the k-means with outliers, a parameter ϵ, a probability summation threshold R, and a set C of centers opened.
Output: An estimation of the oversampling factor.

1: $l_f = -1$, $Q = \emptyset$.
2: Let F be the set of the furthest $(1+\epsilon)z$ data points from X to C.
3: **for** $j = 1$ to $\lfloor \frac{1+\epsilon}{\epsilon} \rfloor$ **do**
4: Let $F' \subseteq F$ be the furthest ϵz points from F to C.
5: $Q = Q \cup \{F'\}$, $F = F \backslash F'$, $l^j_f = \frac{R-|Q|}{1-\frac{d(Q,C)}{d(X,C)}}$.
6: $l_f = \max\{l_f, l^j_f\}$.
7: **end for**
8: Let x be the nearest data point from F to C, and set $Q' = Q \cup (F \backslash \{x\})$.
9: $l'_f = \frac{1}{1-\frac{d(Q',C)}{d(X,C)}}$, $l_f = \max\{l_f, l'_f\}$.
10: **return** l_f.

We argue that such estimation can be used to obtain a lower bound and an upper bound for l^i_{true}. In the j-th iteration of Algorithm 2, denote Q_j as the set of the guessed data points obtained in step 5 of Algorithm 2. Let Q' be the set of the guessed data points obtained in step 8 of Algorithm 2, which contains the furthest $(1+\epsilon)z - 1$ data points from X to C_i. We use l^j_f and l'_f to denote the estimations obtained in step 5 and step 9 of Algorithm 2, respectively. There are two cases that may happen: (1) underestimation of the number of data points whose probabilities boosted by l^i_{true} and C_i are larger than 1, i.e., $Q_j \subseteq T_i$ or $Q' \subseteq T_i$; (2) overestimation of the number of data points whose boosted probabilities are larger than 1, i.e., $T_i \subset Q_j$ or $T_i \subset Q'$. In the i-th iteration of Algorithm 1, let $\lambda(i) = \{l^j_f : j = 1, 2, ..., \lfloor \frac{1+\epsilon}{\epsilon} \rfloor\} \cup \{l'_f\}$ be the set of the estimations for the true oversampling factor l^i_{true}. The following lemma shows that, for both cases, a lower bound for l^i_{true} can be obtained.

Lemma 3.1. *For each $l_\gamma \in \lambda(i)$, it holds that $l_\gamma \le l^i_{true}$.*

By Lemma 3.1, for both cases, any $l_\gamma \in \lambda(i)$ can serve as a lower bound for the true oversampling factor l^i_{true}. Next, we will show that l^i_{true} can be upper bounded by $l^i_{true} \le \max\{\epsilon z l_f, 2l_f\}$, where l_f is the output of Algorithm 2.

Lemma 3.2. *Let l_f be the estimation of the oversampling factor returned by Algorithm 2. Then, $l^i_{true} \le \max\{\epsilon z l_f, 2l_f\}$.*

By Lemma 3.1 and Lemma 3.2, we can obtain the bounds for the true oversampling factor l^i_{true} such that $l_f \le l^i_{true} \le \max\{\epsilon z l_f, 2l_f\}$. Then, in steps 5-7 of Algorithm 1, to obtain an estimation for l^i_{true}, we only need to consider those values that are integer powers of $(1+\epsilon)$ within $[l_f, \max\{\epsilon z l_f, 2l_f\}]$. The number of such values is $O(\frac{\log z}{\epsilon})$. Using binary search on those values, we can obtain an estimation l'_i (step 7 of Algorithm 1) of l^i_{true} such that $\sum_{x\in X} t(l'_i, x, C_i) \ge (1+\epsilon)z$ and $\sum_{x\in X} t(l'_i, x, C_i) \le (1+\epsilon)^2 z$ with $O(\epsilon^{-1}\log\log z)$ binary searching steps.

To analyze the approximation guarantee, we start by dividing the optimal clusters into several groups based on the following definitions of good and bad clusters. We define $G_i = \left\{P^*_j : d(P^*_j, C_i) \le 3.5 OPT_j\right\}$ and $B_i = \{P^*_1, P^*_2, ..., P^*_k\} \backslash G_i$ as the set of good and bad clusters, respectively. Intuitively speaking, bad clusters are those optimal clusters whose centers are not well approximated by the data points in C_i. Our objective is to make most bad clusters good within $O(k/\epsilon)$ rounds. For any optimal cluster P^*_j, recall that $T_\alpha(P^*_j) = \{p \in P^*_j : d(p, c^*_j) < \alpha d\left(P^*_j, \{c^*_j\}\right)/|P^*_j|\}$ is the set of data points in P^*_j close to the optimal clustering center c^*_j, where α is a constant larger than 1. The following lemmas argue that data points in $T_\alpha(P^*_j)$ take a large fraction of the whole data points in P^*_j, and data points in $T_\alpha(P^*_j)$ can well approximate the clustering cost of P^*_j.

Lemma 3.3. *For any $P^*_j \in \mathcal{P}(C^*)$, it holds that $|T_\alpha(P^*_j)| \ge (1 - 1/\alpha)|P^*_j|$.*

Lemma 3.4. *Let P^*_j be an optimal cluster with $d(P^*_j, C_i) = \beta d(P^*_j, \{c^*_j\})$ for some $\beta \ge 3.5$. Then, $d(T_\alpha(P^*_j), C_i) \ge \frac{1}{200}(\beta - 1)d(P^*_j, \{c^*_j\})$ with $\alpha = 2$.*

If the sampled data point x in step 8 of Algorithm 1 is close enough to the center of some bad optimal cluster $P^*_j \in B_i$

(i.e., $x \in T_2(P_j^*)$), then a good approximation for P_j^* can be obtained by adding x to C_i. By Lemma 2.1, it holds that $d(P_j^*, \{x\}) = d(P_j^*, \{c_j^*\}) + |P_j^*| d(x, c_j^*) \leq 3d(P_j^*, \{c_j^*\})$ if $x \in T_2(P_j^*)$. Hence, at least one bad cluster becomes good in that iteration. To analyze the success probability of sampling, we first show that an inlier $x \in N$ can be sampled in each step 8 of Algorithm 1 with probability $\Omega(\epsilon)$.

Let $P_j^* \in B_i$ be an arbitrary bad optimal cluster. We use $\chi(j) = \{x \in T_2(P_j^*) : t(l_i', x, C_i) = 1\}$ to denote the set of data points in $T_2(P_j^*)$ whose probabilities boosted by l_i' and C_i are larger than 1, where l_i' is the estimation for true oversampling factor l_{true}^i obtained in step 7 of Algorithm 1. Then, we further divide the bad optimal clusters into two categories: $\mathcal{L}_i = \{P_j^* : P_j^* \in B_i, |\chi(j)| \geq 0.5|T_2(P_j^*)|\}$ and $\mathcal{S}_i = \{P_j^* : P_j^* \in B_i, |\chi(j)| < 0.5|T_2(P_j^*)|\}$. In the i-th iteration of Algorithm 1, define $\mathcal{L}_i^n = \sum_{P_j^* \in \mathcal{L}_i} |P_j^*|$ and $\mathcal{S}_i^n = \sum_{P_j^* \in \mathcal{S}_i} |P_j^*|$ as the total number of data points in bad optimal clusters that belong to $\mathcal{L}_i$ and $\mathcal{S}_i$, respectively. Then, there are two cases that may happen: (1) $\mathcal{L}_i^n \geq 0.5\epsilon z$; (2) $\mathcal{L}_i^n < 0.5\epsilon z$. If case (1) happens, by the definition of $\mathcal{L}_i$, we can get that $|\chi(j)| \geq 0.5|T_2(P_j^*)| \geq 0.25|P_j^*|$ holds for each $P_j^* \in \mathcal{L}_i$ using Lemma 3.3. Then, by taking a summation over all the data points in $\chi(j)$ for each $P_j^* \in \mathcal{L}_i$, we have $\sum_{P_j^* \in \mathcal{L}_i} |\chi(j)| \geq \sum_{P_j^* \in \mathcal{L}_i} 0.5|T_2(P_j^*)| \geq \sum_{P_j^* \in \mathcal{L}_i} 0.25|P_j^*| = 0.25\mathcal{L}_i^n \geq \frac{\epsilon z}{8}$. Since $\sum_{x \in X} t(l_i', x, C_i) \leq (1+\epsilon)^2 z$, the probability of sampling a data point $x \in \chi(j)$ for some bad cluster $P_j^* \in \mathcal{L}_i$ is at least $\frac{\sum_{x \in \chi(j): P_j^* \in \mathcal{L}_i} t(l_i', x, C_i)}{\sum_{x \in X} t(l_i', x, C_i)} \geq \frac{\epsilon z}{8(1+\epsilon)^2 z} = \frac{\epsilon}{8(1+\epsilon)^2} \geq \frac{\epsilon}{32}$, where the first inequality follows from the definition of $\chi(j)$, and the last inequality follows from the fact that $0 < \epsilon \leq 1$. Then, by adding the sampled data point x to C_i, a bad optimal cluster P_j^* becomes a good cluster.

Then, we consider a more complicated case when $\mathcal{L}_i^n < 0.5\epsilon z$. Let $N' = N \backslash (\cup_{P_j^* \in \mathcal{L}_i} P_j^*)$ be the set of inliers obtained by removing data points in bad optimal clusters of $\mathcal{L}_i$. We will show that, with probability $\Omega(\epsilon)$, the sampled data point x in step 8 of Algorithm 1 belongs to N'. Let E_1 and E_2 be the events that the sampled data point x belongs to N' and $X \backslash N'$, respectively. We use $P_r(E_1)$ and $P_r(E_2)$ to denote the probabilities that events E_1 and E_2 happen, respectively. It holds that $P_r(E_1) = 1 - P_r(E_2) = 1 - \frac{\sum_{x \in Z^*} t(l_i', x, C_i)}{\sum_{x \in X} t(l_i', x, C_i)} - \frac{\sum_{x \in (X \backslash N') \backslash Z^*} t(l_i', x, C_i)}{\sum_{x \in X} t(l_i', x, C_i)} \geq 1 - \frac{z}{(1+\epsilon)z} - \frac{\mathcal{L}_i^n}{(1+\epsilon)z} \geq 1 - \frac{1}{1+\epsilon} - \frac{\epsilon}{2(1+\epsilon)} \geq \frac{\epsilon}{2(1+\epsilon)}$, where the third step follows from the fact that $X \backslash N' = (\bigcup_{P_j^* \in \mathcal{L}_i} P_j^*) \cup Z^*$, and the fourth step follows from the assumption that $\mathcal{L}_i^n < 0.5\epsilon z$ holds in case (2). Hence, with probability $\Omega(\epsilon)$, an inlier from N' can be sampled in step 8 of Algorithm 1.

Let $\mathcal{U}(\mathcal{L}_i) = \bigcup_{P_j^* \in \mathcal{L}_i} P_j^*$ be the collection of data points that belong to the bad optimal clusters in $\mathcal{L}_i$. It holds trivially that if $d(N', C_i) \leq 4OPT$, C_i induces a 4-approximation by discarding $(1 + \frac{\epsilon}{2})z$ data points in X as outliers. Let Z_i be the set of the furthest $(1+\frac{\epsilon}{2})z$ data points from X to C_i. We can get that $d(X \backslash Z_i, C_i) \leq d(X \backslash (Z^* \cup \mathcal{U}(\mathcal{L}_i)), C_i) = d(N', C_i) \leq 4OPT$, where the first inequality follows from the fact that $|Z^* \cup \mathcal{U}(\mathcal{L}_i)| \leq (1+\frac{\epsilon}{2})z$. Thus, we can always assume that $d(N', C_i) > 4OPT$ holds during the sampling process of Algorithm 1 (steps 2-8 of Algorithm 1). Let $H_i = \bigcup_{P_j^* \in \mathcal{S}_i} T_2(P_j^*) \backslash \chi(j)$ be the collection of data points that are close to the bad optimal clusters in $\mathcal{S}_i$ with boosted probabilities by l_i' and C_i smaller than 1. Let $\mathcal{U}(\mathcal{S}_i) = \bigcup_{P_j^* \in \mathcal{S}_i} P_j^*$ be the collection of data points in all bad optimal clusters that belong to $\mathcal{S}_i$. Denote E_3 as the event that the sampled point x in step 8 of Algorithm 1 belongs to H_i. Define $P_r(E_3)$ as the probability that event E_3 happens. The following lemma shows that event E_3 happens with probability $\Omega(\epsilon)$.

Lemma 3.5. *Let E_3 be the event that the sampled data point x in step 8 of Algorithm 1 belongs to H_i. Then, we have $P_r(E_3) \geq \frac{\epsilon}{35840}$.*

By Lemma 3.5, if case (2) happens, with probability $\Omega(\epsilon)$, at least one bad cluster in $\mathcal{S}_i$ becomes good in each iteration. Let $\mathcal{C}_{q+1}$ be the set of centers opened after repeating the sampling process in Algorithm 1 for $q = O(\frac{k}{\epsilon})$ rounds. The following lemma shows that $\mathcal{C}_{q+1}$ yields a 4-approximate solution with $(1 + \frac{\epsilon}{2})z$ data points discarded as outliers.

Lemma 3.6. *By repeating the sampling process of Algorithm 1 for $q = O(\frac{k}{\epsilon})$ rounds, with constant probability, we have $|\mathcal{S}_{q+1}| = 0$ and $\mathcal{L}_{q+1}^n \leq \frac{\epsilon z}{2}$.*

Lemma 3.6 shows that, if Algorithm 1 samples $O(k/\epsilon)$ points using $q = O(k/\epsilon)$ rounds, a set $\mathcal{C}_{q+1}$ of centers with $|\mathcal{S}_{q+1}| = 0$ and $\mathcal{L}_{q+1}^n \leq \frac{\epsilon z}{2}$ can be obtained. Then, by discarding the furthest $(1 + \frac{\epsilon}{2})z$ data points from X to $\mathcal{C}_{q+1}$, a 3.5-approximate solution can be obtained with $O(k/\epsilon)$ centers opened and $(1 + \frac{\epsilon}{2})z$ outliers discarded. More specifically, let Z_{q+1} be the set of the furthest $(1 + \frac{\epsilon}{2})z$ data points from X to $\mathcal{C}_{q+1}$. We can get that $d(X \backslash Z_{q+1}, \mathcal{C}_{q+1}) \leq d(X \backslash (\mathcal{U}(\mathcal{L}_{q+1}) \cup Z^*), \mathcal{C}_{q+1}) \leq 3.5OPT$, where the last inequality follows from Lemma 3.6 that $\mathcal{L}_{q+1}^n \leq \frac{\epsilon z}{2}$. However, the analysis is based on the assumption that $d(N', C_i) > 4OPT$ holds for the whole sampling process. Once $d(N', C_i) \leq 4OPT$ happens in the i-th iteration of Algorithm 1 for some $i \leq q$, a 4-approximate solution with $(1 + \frac{\epsilon}{2})z$ outliers discarded and $O(k/\epsilon)$ centers opened can be obtained. Thus, the final approximation ratio should be 4 by the standard worst case analysis. Putting all these together, Theorem 1.1 can be proved (detailed proofs in Appendix B).

Under the assumption that each optimal cluster has size at least $3z$ (Im et al., 2020), in the i-th iteration of Algorithm 1, $|\chi(j)| \geq 0.5|T_2(P_j^*)| \geq 0.25|P_j^*| \geq \frac{3}{4}z$ holds for each

optimal cluster $P_j^* \in \mathcal{L}_i$. In this case, data points from $\chi(j)$ for each optimal cluster $P_j^* \in \mathcal{L}_i$ can be picked with constant probability in each sampling iteration. Moreover, there is no need to obtain a very accurate estimation for l_{true}^i, and we can let ϵ be any fixed constant smaller than 1. Hence, after repeating Algorithm 1 for $q = O(k)$ rounds, with constant probability, we can get that $|\mathcal{L}_{q+1}| = 0$. In this case, we have $N' = N \backslash (\cup_{P_j^* \in \mathcal{L}_{q+1}} P_j^*) = N$, and the probability of picking an inlier from N' becomes a constant. By Lemma 3.5, the probability of picking a data point $x \in H_i$ becomes a constant if $\mathcal{L}_{q+1}^n = 0$. Thus, an improved 4-approximation can be obtained with $O(k)$ centers opened and exactly z outliers discarded, which proves Theorem 1.3 (detailed proofs in Appendix B).

3.2. The Center Reduction Algorithm

In section 3.1, a 4-approximate solution with $O(k/\epsilon)$ centers opened and at most $(1 + \frac{\epsilon}{6})z$ outliers discarded can be obtained by replacing ϵ with $\epsilon_1 = \frac{\epsilon}{6}$ as the input for Algorithm 1. To reduce the number of centers opened, a direct way is to apply a weighted k-means with outliers algorithm (such as the algorithm in (Grunau & Rozhoň, 2022)) on the weighted instance constructed using the $O(k/\epsilon)$ centers opened, where an $O(1)$-approximate solution can be obtained with $O(z)$ outliers discarded. To further reduce the number of outliers discarded, we propose a center reduction algorithm as described in Algorithm 3, where a $(O(\frac{1}{\epsilon}), 1+\epsilon)$-approximate solution can be obtained. Algorithm 3 uses the Fast-Sampling method to find an initial solution C_1 with size $O(\frac{k}{\epsilon})$ and $(1+\frac{\epsilon}{3})z$ outliers discarded (step 2 of Algorithm 3). Let Z be the set of the furthest $(1 + \frac{\epsilon}{3})z$ data points in X to C_1. Define $\mathcal{L}_1' = \{P_j^* : d(P_j^*, C_1) > 3.5OPT_j, |\chi(j)| \geq 0.5|T_2(P_j^*)|\}$. Let $N_1' = N \backslash \mathcal{U}(\mathcal{L}_1')$ be the set of inliers obtained by removing the data points in bad optimal clusters of $\mathcal{L}_1'$. By Lemma 3.6, we have $d(N_1', C_1) \leq 4OPT$. The goal of Algorithm 3 is to recover most of the discarded inliers in $N_1' \cap Z$.

Observe that there are at most $\frac{\epsilon z}{12}$ inliers in N_1' with distances larger than $\frac{48OPT}{\epsilon z}$ to C_1. Otherwise $d(N_1', C_1) > 4OPT$, which contradicts the fact that $d(N_1', C_1) \leq 4OPT$. Hence, it holds trivially that all the data points in $X \backslash Z$ are within distances $\frac{48OPT}{\epsilon z}$ to C_1. In each round of the for loop in step 7 of Algorithm 3, the algorithm guesses the nearest $\frac{\epsilon z}{12}$ data points from Z to C_1 as the points in $N_1' \cap Z$ with distances smaller than $\frac{48OPT}{\epsilon z}$ to C_1, and removes them from Z. Since each time we recall a number of $\frac{\epsilon z}{12}$ data points in Z, at most $\frac{\epsilon z}{12}$ inliers in N_1' with distances smaller than $\frac{48OPT}{\epsilon z}$ may not be recalled among $O(1/\epsilon)$ guesses. Note that there are also at most $\frac{\epsilon z}{12}$ inliers in N_1' with distances larger than $\frac{48OPT}{\epsilon z}$ to C_1. Then, there exists at least one guess such that at most $\frac{\epsilon z}{6}$ inliers in N_1' are discarded as outliers (with at most $\frac{\epsilon z}{12}$ data points in $N_1' \cap Z$ that are not recalled and $\frac{\epsilon z}{12}$ data points in N_1' with distances larger than $\frac{48OPT}{\epsilon z}$ to C_1) while the remaining data points in $X \backslash Z$ are all within distances $\frac{48OPT}{\epsilon z}$ to C_1. Then, in this guess, a weighted instance can be constructed by assigning the data points in $X \backslash Z$ to their closest centers such that the remaining number of outliers to be discarded can be bounded by $(1 + \frac{\epsilon}{3})z - |Z|$. Then, by calling a $(O(\frac{1}{\epsilon}), 1+\epsilon)$-approximation algorithm $\mathcal{F}$ (such as the algorithm proposed in (Grunau & Rozhoň, 2022)) on the weighted instance and setting $(1 + \frac{\epsilon}{3})z - |Z|$ as the number of outliers to be discarded with $\epsilon = \frac{\epsilon}{3}$ as input, an $O(1/\epsilon)$-approximate solution can be obtained with $(1+\epsilon)z$ outliers discarded. Putting all these together, Theorem 1.2 can be proved (detailed proof of Theorem 1.2 is given in Appendix B).

Algorithm 3 Center-Reduction($X, k, z, d, \epsilon, \eta, \mathcal{F}$)

Input: An instance (X, k, z, d) of the k-means with outliers, parameters ϵ and η, a weighted approximation algorithm $\mathcal{F}$ with k centers opened and $(1+\epsilon)z$ outliers discarded.
Output: A set $C \subset \mathbb{R}^d$.

1: $\epsilon_1 = \frac{\epsilon}{6}$, $\epsilon_2 = \frac{\epsilon}{3}$.
2: Initialize C_1 = Fast-Sampling$(X, k, z, d, \eta, \epsilon_1)$.
3: $C_1' = C_1$.
4: Let Z be the set of the furthest $(1 + \frac{\epsilon}{3})z$ data points in X to C_1, and assign each data point $x \in X \backslash Z$ to its closest center in C_1'.
5: For each $c \in C_1'$, let $w(c)$ be the number of points assigned to c, and assign a weight $w(c)$ to c.
6: Call Algorithm $\mathcal{F}$ on C_1' to obtain a solution C_f using $\epsilon = \epsilon_2$ and $z = (1 + \frac{\epsilon}{3})z - |Z|$ as inputs, and set $best = d(X \backslash Z', C_f)$, where Z' is the set of the furthest $(1+\epsilon)z$ data points from X to C_f.
7: **for** $j = 1$ to $\lceil \frac{2(1+\epsilon_2)}{\epsilon_1} \rceil$ **do**
8: Let $T_j \subseteq Z$ be the set of the nearest $\frac{\epsilon_1 z}{2}$ data points from Z to C_1, and set $Z = Z \backslash T_j$, $C_1' = C_1$.
9: Assign each $x \in X \backslash Z$ to its closest center in C_1'.
10: For each $c \in C_1'$, let $w(c)$ be the number of points assigned to c, and assign a weight $w(c)$ to c.
11: Call Algorithm $\mathcal{F}$ on C_1' to obtain a solution C_2 using $\epsilon = \epsilon_2$ and $z = (1 + \frac{\epsilon}{3})z - |Z|$ as inputs.
12: Let Z_2 be the set of the furthest $(1+\epsilon)z$ data points from X to C_2.
13: **if** $d(X \backslash Z_2, C_2) < best$ **then**
14: $best = d(X \backslash Z_2, C_2)$, $C_f = C_2$.
15: **end if**
16: **end for**
17: **return** C_f.

4. Experiments

In this section, we demonstrate the performance of our algorithms by comparing with state-of-the-art algorithms for the

Table 2. Comparison results of k-means with outliers approximation algorithms. "NA" indicates the algorithm timed out.

Dataset	Method	Cost	Recall	Time(s)	Dataset	Method	Cost	Recall	Time(s)
Skin-5 ($k = 10$)	TIKmeans	**68049.49**	0.7249	1.40	Skin-10 ($k = 10$)	TIKmeans	**72448.37**	0.9313	1.34
	IKmeans	72714.88	**0.7657**	**0.59**		IKmeans	79923.51	**0.9366**	**0.57**
	RobustKmeans++	71209.78	0.7645	0.66		RobustKmeans++	75818.17	0.9233	0.63
	NKmeans	74097.74	0.7326	10.99		NKmeans	76784.33	0.9305	10.87
	LS++	70614.17	0.7415	2374.42		LS++	75520.31	0.9329	2396.30
SUSY-5 ($k = 10$)	TIKmeans	**62009744.02**	**0.7610**	52.37	SUSY-10 ($k = 10$)	TIKmeans	**63745990.97**	0.9657	51.28
	IKmeans	64567215.89	0.6432	**31.02**		IKmeans	65767311.47	**0.9806**	**31.87**
	RobustKmeans++	64039304.52	0.7585	49.85		RobustKmeans++	65524845.52	0.9761	49.24
	NKmeans	65176198.19	0.7090	256.40		NKmeans	66202766.40	0.9581	382.34
	LS++	NA	NA	>24h		LS++	NA	NA	>24h
SIFT-5 ($k = 100$)	TIKmeans	**12033800256**	**1**	10906.2	SIFT-10 ($k = 100$)	TIKmeans	12081741573	**1**	10733.8
	IKmeans	12275654391	**1**	**9635.4**		IKmeans	12107041166	**1**	**9374.2**
	RobustKmeans++	12077778481	0.9998	92840.3		RobustKmeans++	**11983478657**	**1**	92223.8
	NKmeans	NA	NA	>72h		NKmeans	NA	NA	>72h
	LS++	NA	NA	>72h		LS++	NA	NA	>72h
Shuttle ($k = 10$)	TIKmeans	**65588.61**	0	0.80	KDDFULL ($k = 3$)	TIKmeans	32912080.32	**0.6138**	13.59
	IKmeans	71585.51	**0.2353**	**0.11**		IKmeans	32672394.99	0.6109	**6.35**
	RobustKmeans++	87238.76	0	0.23		RobustKmeans++	32907569.93	0.6109	10.49
	NKmeans	83192.53	0.1765	0.56		NKmeans	**32218082.93**	0.6135	34.79
	LS++	76149.82	0.1765	532.77		LS++	NA	NA	>4h

k-means with outliers problem.

Datasets: We test the performance of different algorithms on real-world datasets including 4 datasets used in (Im et al., 2020; Deshpande et al., 2020) and one large-scale dataset used in (Matsui et al., 2017). The datasets are Skin (245,057 $\times$ 3, $k = 10$), SUSY (5,000,000 $\times$ 18, $k = 10$), Shuttle (43,500 $\times$ 9, $k = 10$), KDDFULL (4,898,431 $\times$ 37, $k = 3$) and SIFT (100,000,000 $\times$ 128, $k = 100$). The outliers are generated by the rules in (Im et al., 2020; Ding & Huang, 2021; Deshpande et al., 2020) as follows. First, we normalize the given dataset such that the mean and standard deviations are 0 and 1 on each dimension, respectively. For the datasets SKIN, SUSY, and SIFT, we randomly add 1% outliers that lie in the range $[-\xi, \xi]^d$ for $\xi = 5$ and 10, and denote the resulting datasets as SKIN-5, SKIN-10, SUSY-5, SUSY-10, SIFT-5, and SIFT-10, respectively. For the dataset Shuttle, it contains 43,500 data points, and 99.6% of them are in the four largest classes. We take the two smallest classes containing 17 data points as outliers. For the dataset KDDFULL, it contains 4,898,431 data points, and 98.3% of them are in the three largest classes. We take 23 smaller classes containing 45,747 data points as outliers.

Algorithms and Parameters: In our experiments, for comparison between approximation algorithms, we consider 5 fast approximation algorithms: our algorithms in Algorithm 3 and Algorithm 1, LS++ in (Grunau & Rozhoň, 2022), NKmeans in (Im et al., 2020), and Robust k-means++ in (Deshpande et al., 2020). In the experiments, we use parameter ϵ to control the number of outliers discarded. We also use parameter δ to control the accuracy of oversampling factor estimation by setting the probability summation threshold in Algorithm 2 as $R = \frac{(1+\epsilon)z}{1-\delta}$. We give evaluations of the performances of our algorithms under different parameter settings in Appendix C.1. In experiments, we fix ϵ, η and δ to be 0.5. The number of sampling iterations is multiplied by a factor of β for $\beta = 1.5$. The number of centers opened and outliers discarded for each dataset follows the same settings in (Im et al., 2020; Deshpande et al., 2020). In Appendix C.2, we also test the performance with varying k ranging from 5 to 50 and varying z ranging from 1% to 10%, respectively. For the SIFT dataset, since the data size is much larger than other datasets, moderate changes in k do not significantly influence the clustering cost. Thus, we test the values of k ranging from 50 to 200 on SIFT. In Appendix C.3, we conduct experiments on the performance of our proposed algorithms with varying β.

In our experiments, we also give comparison between our algorithms and outlier detection methods (details in Appendix C.4). The outlier detection methods adapt two-stage strategies, where a filtering process is used to discard the suspect outliers followed by a clustering process on the remaining data points to obtain the final results. We choose the following algorithms for identifying outliers, which are summarized as follows: (1) The probabilistic method in (Li et al., 2022) (ECO); (2) Two fast outlier detection methods in (Liu et al., 2008) (IFOREST) and (Sugiyama & Borgwardt, 2013) (Sampling). For the clustering step, we use the following methods: k-means++ (Arthur & Vassilvitskii, 2007) (the popular linear-time algorithm) and LSDS++ (Fan et al., 2023) (the fastest constant-approximation algorithm).

In Appendix C.5, we present experiments on specific synthetic datasets to demonstrate the scenarios where our algorithm is particularly effective compared to the RobustK-means++ algorithm (the current fastest approximation algorithm in practice). The synthetic datasets are generated by Gaussian distributions, where the k clustering centers

$\{c_1, c_2, ..., c_k\}$ are randomly selected, with each cluster P_i formed as $P_i = \{x \in P_i : x \sim \mathcal{N}(c_i, \sigma)\}$ using Gaussian generators, where σ is to control the deviation. We test different parameter settings with varying k and σ, where n, ξ, d are fixed to be $n = 10,000, \xi = 0.5, d = 20$ (ξ represents that the generated outliers are in range $[-\xi, \xi]^d$).

Description of Algorithms. The approximation algorithms used in our experiments for comparison are summarized as follows. (1) **NKmeans.** This is the density-based algorithm proposed in (Im et al., 2020), where we run a sampling-based version of the NKmeans algorithm (as given in (Im et al., 2020)) on a coreset for fast implementation; (2) **RobustKmeans++.** This is the sampling-based method in (Deshpande et al., 2020), which picks $O(\frac{kn}{z})$ centers and applies the weighted k-means++ method to obtain the final results; (3) **IKmeans.** This is the algorithm which uses Algorithm 1 to execute $O(\frac{k}{\epsilon})$ iterations with 5 data points independently sampled in each iteration, followed by the weighted k-means++ method (as used in (Deshpande et al., 2020)) to obtain the final results; (4) **TIKmeans.** This is the algorithm which uses Algorithm 1 to execute $O(\frac{k}{\epsilon})$ iterations with 5 data points independently sampled in each iteration, followed by Algorithm 3 to obtain the final results, where the algorithm in (Bhaskara et al., 2019) is used as the weighted algorithm; (5) **LS++.** This is the local search algorithm given in (Grunau & Rozhoň, 2022).

Experimental Setup. We perform our experiments on a machine with 100 Intel Xeon Gold 6230 CPUs and 1TB Memory. Following the settings in (Deshpande et al., 2020; Im et al., 2020; Zhang et al., 2021), we fix the number of centers opened as k by running a weighted k-means algorithm to obtain the final clustering centers (see the description of algorithms). Also, following the settings in (Deshpande et al., 2020; Im et al., 2020; Zhang et al., 2021), we fix the outliers discarded as the furthest z data points to the set of centers obtained for fair comparison. We test the clustering cost, recall, and running time for each algorithm, where recall is defined as the number of ground truth outliers found by the algorithms divided by the number of true outliers. Following the settings in (Im et al., 2020), we run all the algorithms on each dataset 10 times and report the best clustering cost and corresponding recall.

Results. Table 2 shows the comparison results of the clustering cost, recall, and running time when k and z are fixed. On dataset Skin-5 and Shuttle, compared with RobustKmeans++ and NKmeans algorithms, our TIKmeans algorithm reduces the clustering cost by at least 4.43% and 21.16%, respectively. For a total of 8 datasets, the clustering cost is reduced by at least 5% on average compared with other algorithms. For running time, our IKmeans algorithm runs faster than other algorithms. On dataset SIFT-5 and SIFT-10, compared with RobustKmeans++ and NKmeans algorithms, our proposed IKmeans algorithm reduces the running time by at least 50%. The results highlight the increasing advantages of our algorithms as the data size grows. For a total of 8 datasets, on average, our IKmeans algorithm is at least 20% faster than the state-of-the-art algorithms.

The experimental results with different parameter settings (Appendix C.1) show that larger ϵ and η or smaller δ will lead to faster running time of our algorithms. Smaller ϵ and η achieve better clustering quality, while δ does not influence the clustering quality significantly. The experimental results on the performance with a varying number of centers opened and outliers discarded (Appendix C.2) show that our proposed TIKmeans algorithm can still achieve the best clustering quality and recall for most datasets, while our proposed IKmeans algorithm is the fastest among different algorithms. The experimental results with a varying size of centers opened during the sampling process (Appendix C.3) show that the number of centers opened does not influence the recall index much, which indicates that the outliers discarded are not significantly influenced by a relaxed number of centers opened during the sampling process.

5. Conclusion

In this work, we present fast sampling-based methods for the k-means with outliers problem, which have running time independent of the aspect ratio Δ. We show experimentally that our algorithms achieve competitive performance on different datasets compared with other state-of-the-art algorithms for the k-means with outliers problem.

Acknowledgments

This work was supported by National Natural Science Foundation of China (62172446, 62350004, 62332020, 62072476), Open Project of Xiangjiang Laboratory (22XJ02002), and Central South University Research Programme of Advanced Interdisciplinary Studies (2023QYJC023). This work was also carried out in part using computing resources at the High Performance Computing Center of Central South University.

Impact Statement

This work presents fast approximation schemes to deal with the k-means with outliers problem. Our primary purpose here is to offer algorithmic insights to help the clustering process when data noise interferes with clustering decisions. Moreover, since outliers frequently appear across diverse scenarios, our proposed methods exhibit potential for broad applications within the field of machine learning, and we believe that this contribution has minimal realistic downside.

References

Aggarwal, A., Deshpande, A., and Kannan, R. Adaptive sampling for k-means clustering. In *Proceedings of the 12th International Workshop, Approximation, Randomization, and Combinatorial Optimization*, pp. 15–28, 2009.

Arthur, D. and Vassilvitskii, S. k-means++: The advantages of careful seeding. In *Proceedings of the 18th Annual Symposium on Discrete Algorithms*, pp. 1027–1035, 2007.

Bhaskara, A., Vadgama, S., and Xu, H. Greedy sampling for approximate clustering in the presence of outliers. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, pp. 11146–11155, 2019.

Blum, M., Floyd, R. W., Pratt, V. R., Rivest, R. L., Tarjan, R. E., et al. Time bounds for selection. *Journal of Computer and System Sciences*, 7(4):448–461, 1973.

Charikar, M., Khuller, S., Mount, D. M., and Narasimhan, G. Algorithms for facility location problems with outliers. In *Proceedings of the 12th Annual Symposium on Discrete Algorithms*, pp. 642–651, 2001.

Chen, K. A constant factor approximation algorithm for k-median clustering with outliers. In *Proceedings of the 19th Annual Symposium on Discrete Algorithms*, pp. 826–835, 2008.

Cohen-Addad, V. Approximation schemes for capacitated clustering in doubling metrics. In *Proceedings of the 31st Annual Symposium on Discrete Algorithms*, pp. 2241–2259, 2020.

Cohen-Addad, V., Mirrokni, V., and Zhong, P. Massively parallel k-means clustering for perturbation resilient instances. In *Proceedings of the 39th International Conference on Machine Learning*, pp. 4180–4201, 2022.

Deshpande, A., Kacham, P., and Pratap, R. Robust k-means++. In *Proceedings of the 36th Conference on Uncertainty in Artificial Intelligence*, pp. 799–808, 2020.

Ding, H. and Huang, J. Is simple uniform sampling efficient for center-based clustering with outliers: When and why? *CoRR*, abs/2103.00558, 2021.

Fan, C., Li, P., and Li, X. LSDS++: Dual sampling for accelerated k-means++. In *Proceedings of the 40th International Conference on Machine Learning*, pp. 9640–9649, 2023.

Friggstad, Z., Khodamoradi, K., Rezapour, M., and Salavatipour, M. R. Approximation schemes for clustering with outliers. *ACM Transactions on Algorithms*, 15(2):1–26, 2019.

Grunau, C. and Rozhoň, V. Adapting k-means algorithms for outliers. In *Proceedings of the 36th International Conference on Machine Learning*, pp. 7845–7886, 2022.

Gupta, S., Kumar, R., Lu, K., Moseley, B., and Vassilvitskii, S. Local search methods for k-means with outliers. *Proceedings of the VLDB Endowment*, 10:757–768, 2017.

Im, S., Qaem, M. M., Moseley, B., Sun, X., and Zhou, R. Fast noise removal for k-means clustering. In *Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics*, pp. 456–466, 2020.

Krishnaswamy, R., Li, S., and Sandeep, S. Constant approximation for k-median and k-means with outliers via iterative rounding. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pp. 646–659, 2018.

Li, Z., Zhao, Y., Hu, X., Botta, N., Ionescu, C., and Chen, G. H. Ecod: Unsupervised outlier detection using empirical cumulative distribution functions. *IEEE Transactions on Knowledge and Data Engineering*, 35(12): 12181–12193, 2022.

Liu, F. T., Ting, K. M., and Zhou, Z. H. Isolation forest. In *Proceedings of the 8th IEEE International Conference on Data Mining*, pp. 413–422, 2008.

Matsui, Y., Ogaki, K., Yamasaki, T., and Aizawa, K. Pqk-means: Billion-scale clustering for product-quantized codes. In *Proceedings of the 25th ACM International Conference on Multimedia*, pp. 1725–1733, 2017.

Nguyen, H. L., Nguyen, T., and Jones, M. Fair range k-center. *arXiv preprint arXiv:2207.11337*, 2022.

Sugiyama, M. and Borgwardt, K. M. Rapid distance-based outlier detection via sampling. In *Proceedings of the 26th International Conference on Neural Information Processing Systems*, pp. 467–475, 2013.

Zhang, Z., Feng, Q., Huang, J., Guo, Y., Xu, J., and Wang, J. A local search algorithm for k-means with outliers. *Neurocomputing*, 450:230–241, 2021.

A. Discussions on the Running Time Lower Bounds

In the following, we give a brief discussion on the tradeoff between the number of centers opened, the number of outliers discarded, and the running time given that the approximation guarantee on clustering quality is fixed as a constant. Table 3 summarizes the existing lower bounds on the running time.

When k and z are not relaxed. In this setting, a k-means with outliers instance was constructed in (Grunau & Rozhoň, 2022) such that finding an $O(1)$-approximate solution on this instance requires $\Omega(n^2)$ running time. The example shows that there is a fundamental limit for the runtime for achieving an $O(1)$-approximation with exactly k centers opened and exactly z outliers discarded.

When z is relaxed. In this setting, Grunau and Rozhoň (Grunau & Rozhoň, 2022) provided a lower bound on the running time in the general metric space query model, where a running time of $\Omega(\frac{nk^2}{z})$ is required to obtain a constant approximation with $O(z)$ outliers discarded if $10000k\log k \leq z \leq \frac{n}{10000}$. However, as far as we know, there is no established lower bound for achieving $O(1)$-approximation with $(1+\epsilon)z$ outliers discarded. The current best result with exactly k centers opened and $(1+\epsilon)z$ outliers discarded is an $O(\frac{1}{\epsilon})$-approximation (called LS++) proposed in (Grunau & Rozhoň, 2022) and has a running time of $O(\frac{d}{\epsilon}nk\log k(\log\log k + \frac{1}{\epsilon}\log\frac{1}{\epsilon})\log(n\Delta))$.

When k is relaxed or k and z are both relaxed. In both settings, as far as we know, there is no known lower bound on the running time. The only result with $O(k)$ centers opened and exactly z outliers discarded is a $(1+\epsilon)$-approximation scheme proposed in (Friggstad et al., 2019) with running time $O(n^{d^{O(d)}}poly(k))$. If both k and z are relaxed, no known results can guarantee a constant approximate solution with $(1+\epsilon)z$ outliers discarded and running time that matches our results.

Table 3. Existing running time lower bounds for constant approximation.

	No Relaxation	$O(z)$	$(1+\epsilon)z$	$O(k)$	$O(k)$,$(1+\epsilon)z$	$O(k/\epsilon)$,$(1+\epsilon)z$
Lower Bound	$\Omega(n^2)$	$\Omega(nk^2/z)$	Unknown	Unknown	Unknown	Unknown
Current Best	$n^{O(1/\epsilon^3)}$ [(Krishnaswamy et al., 2018)]	$ndk\log\log n + poly(k)d\log(n\Delta)$ (ours)	Unknown	$n^{d^{O(d)}}poly(k)$ [(Friggstad et al., 2019)]	Unknown	$ndk\log\log n/\epsilon^2$ (ours)

B. Missing Proofs in Section 3

Lemma 3.1. *For each $l_\gamma \in \lambda(i)$, it holds that $l_\gamma \leq l^i_{true}$.*

Proof. We first assume that case (1) happens, where $Q_j \subseteq T_i$ or $Q' \subseteq T_i$. For a data point $x \in X$, let $p_x = \frac{d(x,C_i)}{\sum_{y\in X} d(y,C_i)}$ be the distance-based sampling probability. Given a non-negative integer t, we use $[t] = \{1,2,...,t\}$ to denote the set of integers within range $[1,t]$. We first consider an arbitrary $l_\gamma \in \lambda(i)$ where $l_\gamma = l^j_f$ for some $j \in [\lfloor\frac{1+\epsilon}{\epsilon}\rfloor]$. In this case, we have

$$l^j_f - l^i_{true} = \frac{(1+\epsilon)z - |Q_j|}{1-\sum_{x\in Q_j} p_x} - \frac{(1+\epsilon)z-|T_i|}{1-\sum_{x\in T_i} p_x}$$
$$= \frac{1}{1-\sum_{x\in Q_j} p_x}\left((1+\epsilon)z - |Q_j| - l^i_{true}(1-\sum_{x\in Q_j} p_x)\right).$$

Let $\Psi(j) = ((1+\epsilon)z - |Q_j| - l^i_{true}(1-\sum_{x\in Q_j} p_x))$. Observe that $l^i_{true}(1-\sum_{x\in Q_j} p_x) \geq (1+\epsilon)z - |Q_j|$, where the inequality follows from the fact that the distance-based sampling probabilities of data points in Q_j boosted by l^i_{true} and C_i are larger than 1. Hence, we have $\Psi(j) \leq 0$, which means $l^j_f \leq l^i_{true}$. Since the case for $l'_f \in \lambda(i)$ is similar to the case for l^j_f, we also have $l'_f \leq l^i_{true}$.

Now we assume that case (2) happens, where $T_i \subset Q_j$ or $T_i \subset Q'$. Similar to case (1), let $\Psi'(j) = ((1+\epsilon)z - |Q_j| - l^i_{true}(1-\sum_{x\in Q_j} p_x))$. Observe that since $T_i \subset Q_j$, we have $l^i_{true}(1-\sum_{x\in Q_j} p_x) = l^i_{true}(1-\sum_{x\in T_i} p_x - \sum_{x\in Q_j\setminus T_i} p_x) \geq (1+\epsilon)z - |T_i| - (|Q_j| - |T_i|) = (1+\epsilon)z - |Q_j|$, where the first inequality follows from the fact that data points in $Q_j \cap T_i$ contribute exactly 1 to the probability summation and data points in $Q_j \setminus T_i$ contribute smaller than 1 to the probability summation. Hence, in this case, we have $l^j_f \leq l^i_{true}$. Since the case for l'_f is similar to the case for l^j_f, we also have $l'_f \leq l^i_{true}$ in case (2). □

Lemma 3.2. *Let l_f be the estimation of the oversampling factor returned by Algorithm 2. Then, $l^i_{true} \leq \max\{\epsilon z l_f, 2l_f\}$.*

Proof. Recall that l^i_{true} is the true oversampling factor such that $\sum_{x\in X} t(l^i_{true}, x, C_i) = (1+\epsilon)z$, and $T_i = \left\{x \in X : l^i_{true} d(x, C_i)/d(X, C_i) \geq 1\right\}$ is the set of data points in X whose probabilities boosted by l^i_{true} and C_i are larger than 1. We use $\nu(T_i) = d(T_i, C_i)/d(X, C_i)$ to denote the summation of distance-based sampling probabilities of data points in T_i. Note that data points in T_i can only contribute 1 to the probability summation. In the j-th iteration of the for loop in step 3 of Algorithm 2, let Q_j be the set of the guessed data points obtained in step 5, where the boosted probabilities of data points in Q_j are regarded to be larger than 1. Let Q' be the set of the guessed data points obtained in step 8 of Algorithm 2. Define $\nu(j) = d(Q_j, C_i)/d(X, C_i)$ as the distance-based probability summation of data points in Q_j. Let $Q_0 = \emptyset$. There are two cases that may happen: (1) $T_i \subseteq Q_j$ and $T_i \nsubseteq Q_{j-1}$ for some $j \in \left[\lfloor \frac{1+\epsilon}{\epsilon} \rfloor\right]$; (2) $T_i \subseteq Q'$. If case (1) happens, we have

$$\begin{aligned} l^j_f &= \frac{R - |Q_j|}{1 - \nu(j)} \\ &= \frac{R - |T_j| - |Q_j \backslash T_i|}{1 - \nu(j)} \\ &\geq \frac{R - |T_i| - |Q_j \backslash T_i|}{1 - \nu(T_i)} \\ &\geq \frac{0.5(R - |T_i|)}{1 - \nu(T_i)} \\ &= 0.5 l^i_{true}, \end{aligned}$$

where the first inequality follows from the fact that $T_i \subseteq Q_j$ and $\nu(T_i) \leq \nu(j)$, and the second inequality follows from the fact that $|Q_j \backslash T_i| \leq \epsilon z$ and $R - |T_i| \geq \epsilon z + |Q_j \backslash T_i|$. Hence, in this case, we have $l^i_{true} \leq 2 l^j_f$.

If case (2) happens, let $\nu' = d(Q', C_i)/d(X, C_i)$. Then, it holds that

$$l'_f = \frac{1}{1 - \nu'} \geq \frac{1}{1 - \nu(T_i)} \geq \frac{\frac{1}{\epsilon z}(R - |T_i|)}{1 - \nu(T_i)} \geq \frac{l^i_{true}}{\epsilon z},$$

where the first inequality follows from the fact that $\nu(T_i) \leq \nu'$, and the second inequality follows from the fact that $R - |T_i| \leq \epsilon z$. Since l_f is taken as the maximum value among all l^j_f and l'_f, we can get that $l^i_{true} \leq \max\{2 l_f, \epsilon z l_f\}$. □

Lemma 3.3 *For any $P^*_j \in \mathcal{P}(C^*)$, it holds that $|T_\alpha(P^*_j)| \geq (1 - 1/\alpha)|P^*_j|$.*

Proof.

$$\begin{aligned} d(P^*_j, \left\{c^*_j\right\}) &\geq d(P^*_j \backslash T_\alpha(P^*_j), \{c^*_j\}) \\ &\geq |P^*_j| \left(1 - \frac{|T_\alpha(P^*_j)|}{|P^*_j|}\right) \frac{\alpha d(P^*_j, \left\{c^*_j\right\})}{|P^*_j|}, \end{aligned}$$

which implies that $|T_\alpha(P^*_j)| \geq \left(1 - \frac{1}{\alpha}\right)|P^*_j|$. □

Lemma 3.4 *Let P^*_j be an optimal cluster with $d(P^*_j, C_i) = \beta d(P^*_j, \left\{c^*_j\right\})$ for some $\beta \geq 3.5$. Then, $d(T_\alpha(P^*_j), C_i) \geq \frac{1}{200}(\beta - 1) d(P^*_j, \left\{c^*_j\right\})$ with $\alpha = 2$.*

Proof. Let s_j be the closest center in C_i to c^*_j. It holds that $d(c^*_j, s_j) \geq \frac{(\beta-1) d(P^*_j, \left\{c^*_j\right\})}{|P^*_j|}$. Otherwise $d(P^*_j, \{s_j\}) < \beta d(P^*_j, \left\{c^*_j\right\})$ holds by using Lemma 2.1, which contradicts with the assumption that $d(P^*_j, C_i) = \beta d(P^*_j, \left\{c^*_j\right\})$. Hence, for any data point $x \in T_\alpha(P^*_j)$, we have

$$\begin{aligned} \sqrt{d(x, C_i)} &\geq \sqrt{d(c^*_j, C_i)} - \sqrt{d(x, c^*_j)} \\ &\geq \sqrt{\frac{d(P^*_j, \left\{c^*_j\right\})}{|P^*_j|}} \left(\sqrt{\beta - 1} - \sqrt{\alpha}\right). \end{aligned}$$

Since $\sqrt{\beta-1} \geq \sqrt{2.5}$ and $\alpha = 2$, we have $\sqrt{\alpha} \leq \sqrt{\frac{4(\beta-1)}{5}}$. Thus,

$$\begin{aligned} d(T_2(P_j^*), C_i) &\geq |T_2(P_j^*)| \frac{d(P_j^*, \{c_j^*\})}{|P_j^*|} \left(\sqrt{\beta-1} - \sqrt{\alpha}\right)^2 \\ &\geq \frac{1}{200}(\beta-1) d(P_j^*, \{c_j^*\}), \end{aligned}$$

where the last step follows from Lemma 3.3. □

Lemma 3.5. *Let E_3 be the event that the sampled data point x in step 8 of Algorithm 1 belongs to H_i. Then, we have $P_r(E_3) \geq \frac{\epsilon}{35840}$.*

Proof. Consider an arbitrary bad optimal cluster $P_j^* \in \mathcal{S}_i$. By Lemma 3.4, if $d(P_j^*, C_i) = \beta d(P_j^*, \{c_j^*\})$ for $\beta \geq 3.5$, then $d(p, C_i) \geq \frac{1}{200}(\beta-1)\frac{d(P_j^*, \{c_j^*\})}{|P_j^*|}$ holds for each $p \in T_2(P_j^*) \backslash \chi(j)$. By the definition of the optimal clusters in $\mathcal{S}_i$, we have $|\chi(j)| < 0.5|T_2(P_j^*)|$, which implies that $|T_2(P_j^*)\backslash\chi(j)| \geq 0.5|T_2(P_j^*)| \geq 0.25|P_j^*|$. Then, we have

$$\begin{aligned} d(T_2(P_j^*)\backslash\chi(j), C_i) &\geq \frac{|P_j^*|}{800}(\beta-1)\frac{d(P_j^*, \{c_j^*\})}{|P_j^*|} \\ &\geq \frac{\beta-1}{800} d(P_j^*, \{c_j^*\}) \\ &= \frac{\beta-1}{800\beta} d(P_j^*, C_i) \\ &\geq \frac{1}{1120} d(P_j^*, C_i), \end{aligned}$$

where the last inequality follows from $\beta \geq 3.5$. Then,

$$\begin{aligned} P_r(E_3) &= P_r(E_1) P_r(E_3|E_1) \\ &\geq \frac{\epsilon}{2(1+\epsilon)} \frac{\sum_{x \in H_i} t(l_i', x, C_i)}{\sum_{x \in N'} t(l_i', x, C_i)} \\ &\geq \frac{\epsilon}{2(1+\epsilon)} \frac{\sum_{x \in H_i} l_i' d(x, C_i)}{\sum_{x \in N'} l_i' d(x, C_i)} \\ &\geq \frac{\epsilon}{2(1+\epsilon)} \frac{\sum_{P_j^* \in \mathcal{S}_i} d(T_2(P_j^*)\backslash\chi(j), C_i)}{d(N', C_i)} \\ &\geq \frac{\epsilon}{1+\epsilon} \frac{d(\mathcal{U}(\mathcal{S}_i), C_i)}{2240 d(N', C_i)}, \end{aligned}$$

where the second inequality follows from the fact that $t(l_i', x, C_i) = l_i' d(x, C_i)$ holds for each $x \in H_i$, and the last inequality follows from $d(T_2(P_j^*)\backslash\chi(j), C_i) \geq \frac{1}{1120} d(P_j^*, C_i)$ for each $P_j^* \in B_i$. By the definition of good and bad clusters, we have that each good optimal cluster $P_h^* \in G_i$ has clustering cost $d(P_h^*, C_i) \leq 3.5 OPT_j$, and each bad optimal cluster $P_h^* \in B_i$ has clustering cost $d(P_h^*, C_i) > 3.5 OPT_j$. Moreover, according to the case assumption we have $d(N', C_i) \geq 4OPT$. Then, it holds that $\frac{d(\mathcal{U}(\mathcal{S}_i), C_i)}{d(N', C_i)} \geq 1 - \frac{\sum_{P_h^* \in G_i} d(P_h^*, C_i)}{d(N', C_i)} \geq 1 - \frac{3.5}{4} = \frac{1}{8}$, which indicates that $P_r(E_3) \geq \frac{\epsilon}{35840}$ if $\epsilon \leq 1$. □

Lemma 3.6. *By repeating the sampling process of Algorithm 1 for $q = O(\frac{k}{\epsilon})$ rounds, with constant probability, we have $|\mathcal{S}_{q+1}| = 0$ and $\mathcal{L}_{q+1}^n \leq \frac{\epsilon z}{2}$.*

Proof. In the i-th iteration of our Algorithm 1, we first consider the case that $|\mathcal{L}_{i+1}| < |\mathcal{L}_i|$ happens. Define a variable $a_i = 1$ if $|\mathcal{L}_{i+1}| < |\mathcal{L}_i|$. Otherwise $a_i = 0$. It holds that a_i has value 1 with probability at least $\frac{\epsilon}{32}$ if $\mathcal{L}_i^n \geq \frac{\epsilon z}{2}$. Let $p = \frac{\epsilon}{32}$. Denote q_1 as the total number of sampling iterations required to make all the optimal clusters in $\mathcal{L}_1$ good when $\mathcal{L}_i^n \geq \frac{\epsilon z}{2}$

always holds during the sampling process. Let $q_1 = \frac{64tk}{\epsilon}$, where $t \geq 1$ is a large constant. Assume that $\mathcal{L}_i^n \geq \frac{\epsilon z}{2}$ holds for q_1 iterations. In the worst case, there are k bad optimal clusters in the beginning. Then, by Chernoff Bound, we have

$$\begin{aligned} P_r\left(\sum_{j=1}^{q_1} a_j < k\right) &< P_r\left(\sum_{j=1}^{q_1} a_j < \frac{pq_1}{2}\right) \\ &< e^{-\frac{pq_1}{8}} \\ &< e^{-\frac{k}{4}} \\ &\leq e^{-\frac{1}{4}}. \end{aligned}$$

This implies that $P_r(\sum_{j=1}^{q_1} a_j \geq k) \geq 1 - e^{-\frac{1}{4}}$. Thus, by repeating the sampling process $q_1 = O(\frac{k}{\epsilon})$ rounds, with constant probability, either all the bad optimal clusters in $\mathcal{L}_1$ become good or $\mathcal{L}_{j+1}^n < \frac{\epsilon z}{2}$ happens after the j-th iteration of the sampling process for some $j < q_1$. For both cases, we have $\mathcal{L}_{q_1+1}^n < \frac{\epsilon z}{2}$.

After executing $q_1 = O(\frac{k}{\epsilon})$ iterations of the sampling process for Algorithm 1, we can get that $\mathcal{L}_{q_1+1}^n < \frac{\epsilon z}{2}$. Now we consider the case that $|\mathcal{S}_{i+1}| < |\mathcal{S}_i|$ happens. By Lemma 3.5, this case happens with probability at least $\frac{\epsilon}{35840}$ if $\mathcal{L}_i^n \leq \frac{\epsilon z}{2}$. Similar to the case that $|\mathcal{L}_{i+1}| < |\mathcal{L}_i|$, by repeating the sampling process for another $q_2 = \frac{71680kt}{\epsilon}$ iterations for some constant $t \geq 1$, with constant probability, all the bad optimal clusters in $\mathcal{S}_1$ become good. Putting all these together, by repeating the sampling process for $q = O(\frac{k}{\epsilon})$ rounds, we can obtain a set $\mathcal{C}_{q+1}$ of centers such that $|\mathcal{S}_{q+1}| = 0$ and $\mathcal{L}_{q+1}^n \leq \frac{\epsilon z}{2}$ hold with constant probability. □

Theorem 1.1. *For the k-means with outliers problem, there exists an algorithm that runs in time $O(\frac{ndk \log\log n}{\epsilon^2})$ and outputs a 4-approximate solution with constant probability by opening $O(k/\epsilon)$ centers and discarding $(1+\epsilon)z$ outliers.*

Proof. For approximation guarantees on clustering quality, by Lemma 3.6 and the case assumptions, we can obtain a 4-approximate solution with $O(\frac{k}{\epsilon})$ centers opened and $(1+\frac{\epsilon}{2})z$ outliers discarded using Algorithm 1. For the running time, in each iteration, the algorithm requires to calculate the distances between data points to the current set of centers opened for oversampling factor estimation and sampling-based probability distribution construction. In each iteration, the distances from data points to their nearest centers can be updated in time $O(nd)$ by maintaining the nearest centers (or distances) from the data points to the current set of centers opened. On the other hand, with nearest distances, the oversampling factor finding process (steps 3-7 of Algorithm 1) can be executed in time $O(\frac{nd \log\log n}{\epsilon})$ using linear selection methods (Blum et al., 1973) to find the furthest ϵz data points to the current set of centers opened. Thus, each sampling iteration can be executed in time $O(\frac{nd \log\log n}{\epsilon})$. By repeating the sampling process for $O(k/\epsilon)$ rounds, the total running time is $O(\frac{ndk \log\log n}{\epsilon^2})$. □

Theorem 1.3. *For the k-means with outliers problem, under the assumption that each optimal cluster has size at least $3z$, there exists an algorithm that runs in time $O(ndk \log\log n)$ and outputs a 4-approximate solution with constant probability by opening $O(k)$ centers and discarding z outliers.*

Proof. If each optimal cluster has size at least $3z$, then it holds trivially that each optimal cluster in $\mathcal{L}_i$ can be sampled with constant probability. Hence, by Lemma 3.6, after repeating the sampling process in Algorithm 1 for $q_1 = O(k)$ rounds, all bad clusters in $\mathcal{L}_1$ become good with constant probability. Then, since $\mathcal{L}_{q_1+1}^n = 0$, the probability of picking an inlier from N' becomes a constant. Hence, according to Lemma 3.5, the probability of picking an inlier close to the center of some bad optimal cluster in $\mathcal{S}_{q_1+1}$ also becomes a constant. Consequently, according to Lemma 3.6, after repeating the sampling process in Algorithm 1 for another $q_2 = O(k)$ rounds, all bad optimal clusters in $\mathcal{S}_1$ become good with constant probability, which yields a 4-approximate solution with $O(k)$ centers opened and exactly z outliers discarded. □

Theorem 1.2. *For the k-means with outliers problem, there exists an algorithm that runs in time $O(\frac{ndk \log\log n}{\epsilon^2} + dpoly(k, \frac{1}{\epsilon}) \log(n\Delta))$ and outputs an $O(\frac{1}{\epsilon})$-approximate solution with constant probability by opening exactly k centers and discarding $(1+\epsilon)z$ outliers.*

Proof. Observe that there must exist a weighted instance C_1', where at most $\frac{\epsilon z}{6}$ inliers in N_1' are discarded by C_1' (i.e., the set of data points in N_1' that are not recalled back by step 8 of Algorithm 3). Denote the set of data points in N_1' discarded by C_1'

as $\mathcal{O}(N_1')$. Let Z_g be the set of data points discarded by weighted instance C_1' (i.e., the set Z obtained after executing step 8 of Algorithm 3). Let C_f be the solution obtained by calling a $(O(\frac{1}{\epsilon}), 1+\epsilon)$-approximation algorithm $\mathcal{F}$ on C_1' with $\epsilon = \frac{\epsilon}{3}$ and $z = (1+\frac{\epsilon}{3})z - |Z_g|$ as inputs. Let Z_f^w be the set of data points in C_1' discarded by the weighted algorithm $\mathcal{F}$, where each data point $z \in Z_f^w$ is associated with a weight $w'(z)$ to denote the weight units of z discarded by algorithm $\mathcal{F}$. Based on Z_f^w, we can obtain a set Z_f of data points with size $\sum_{x\in Z_f^w} w'(x)$ as the set of outliers in X discarded by $\mathcal{F}$. More specifically, for each data point $c \in C_1'$, let $\phi(c)$ be the set of data points in $X\backslash Z_g$ assigned to c. Then, for each $c \in Z_f^w$, we can discard an arbitrary subset $\phi'(c) \subseteq \phi(c)$ with size $w'(c)$ as the data points discarded by the weighted Algorithm $\mathcal{F}$, where $Z_f = \cup_{c\in Z_f^w}\phi'(c)$. According to the definition of Z_f, we can get that $|Z_f| = (1+\frac{\epsilon}{3})((1+\frac{\epsilon}{3})z - |Z_g|)$.

Let $\mathcal{H} = X\backslash(Z_f \cup Z_g)$. By assigning the data points in $\mathcal{H}$ to their closest centers in C_1 (C_1 is the initial set of centers obtained in step 2 of Algorithm 1), for each $c \in C_1$, we use $\sigma(c)$ to denote the number of data points in $\mathcal{H}$ that are assigned to c. For each point $x \in X$, let y_x denote the closest point in C_1 to x. We use C_w^* to denote the optimal solution of the weighted k-center with outliers instance C_1', where each $c \in C_1'$ is associated with a weight $w''(c)$ to denote the weight units of c that are not discarded by the optimal solution C_w^*. Define $\mathcal{V} = (X\backslash Z_g)\backslash(Z^* \cup \mathcal{U}(\mathcal{L}_1'))$. By assigning the data points in $\mathcal{V}$ to their closest center in C_1', for each $c \in C_1'$, we use $\sigma'(c)$ to denote the number of data points in $\mathcal{V}$ assigned to c. Denote Z_f' as the set of the furthest $(1+\epsilon)z$ data points from X to C_f. Then, we have that $d(X\backslash Z_f', C_f) \le d(X\backslash(Z_f \cup Z_g), C_f)$ according to the definition of Z_f' and the fact that $|Z_f \cup Z_g| = |Z_g| + (1+\frac{\epsilon}{3})((1+\frac{\epsilon}{3})z - |Z_g|) \le (1+\frac{\epsilon}{3})^2 z \le (1+\epsilon)z$, where the last inequality follows from $0 < \epsilon \le 1$. Next, we show how to bound the cost of $d(X\backslash(Z_f \cup Z_g), C_f)$. Based on the definitions of Z_f and Z_g, we can get that

$$
\begin{aligned}
d(X\backslash(Z_f \cup Z_g), C_f) &= \sum_{x\in X\backslash(Z_f\cup Z_g)} d(x, C_f) \\
&\le \sum_{x\in X\backslash(Z_f\cup Z_g)} 2d(x, y_x) + \sum_{x\in X\backslash(Z_f\cup Z_g)} 2d(y_x, C_f) \\
&\le \sum_{x\in X\backslash Z_g} 2d(x, y_x) + 2\sum_{c\in C_1'} \sigma(c) d(c, C_f) \\
&\le \sum_{x\in (X\backslash Z_g)\backslash(Z^*\cup\mathcal{U}(\mathcal{L}_1'))} 2d(x, y_x) + \sum_{x\in (X\backslash Z_g)\cap(Z^*\cup\mathcal{U}(\mathcal{L}_1'))} 2d(x, y_x) + O(\frac{1}{\epsilon})\sum_{c\in C_1'} w''(c) d(c, C_w^*) \\
&\le \sum_{x\in (X\backslash Z_g)\backslash(Z^*\cup\mathcal{U}(\mathcal{L}_1'))} 2d(x, y_x) + \sum_{x\in (X\backslash Z_g)\cap(Z^*\cup\mathcal{U}(\mathcal{L}_1'))} 2d(x, y_x) + O(\frac{1}{\epsilon})\sum_{c\in C_1'} \sigma'(c) d(c, C^*) \\
&\le O(1)OPT + O(\frac{OPT}{\epsilon}) + O(\frac{1}{\epsilon})\left(\sum_{x\in (X\backslash Z_g)\backslash(Z^*\cup\mathcal{U}(\mathcal{L}_1'))} 2d(x, y_x) + 2d(x, C^*)\right) \\
&\le O(\frac{1}{\epsilon})OPT + O(\frac{1}{\epsilon})OPT \\
&= O(\frac{1}{\epsilon})OPT,
\end{aligned}
$$

where the second inequality follows from the definition of $\sigma(x)$, the third inequality follows from the fact that $\mathcal{F}$ is a $(O(\frac{1}{\epsilon}), 1+\epsilon)$-approximation weighted algorithm for the k-center with outliers problem, the fourth inequality follows from the fact that C_w^* is the optimal solution for weighted instance C_1' and $|Z^* \cup Z_g \cup \mathcal{U}(\mathcal{L}_1')| \le |Z^*| + |\mathcal{U}(\mathcal{L}_1')| + |\mathcal{O}(N_1')| \le z + \frac{\epsilon z}{6} + \frac{\epsilon z}{6} = (1+\frac{\epsilon}{3})z$, the fifth inequality follows from the fact that $d(N_1', C_1) \le 4OPT$ and all the data points in $X\backslash Z_g$ are within distances $\frac{48OPT}{\epsilon z}$ to C_1, and the last inequality follows from the fact that $d(N_1', C_1) \le 4OPT$. Moreover, the number of outliers discarded can be bounded by $|Z_f \cup Z_g| \le (1+\frac{\epsilon}{3})(1+\frac{\epsilon}{3})z \le (1+\epsilon)z$ for $0 < \epsilon \le 1$. The running time analysis is similar to that of Algorithm 1 using linear selection methods (Blum et al., 1973) to recall the data points. □

C. Complementary Experimental Results

C.1. Experimental Performances with Different Parameter Settings

In this subsection, we test the performances of our algorithms under different parameter settings on datasets Skin-5 and Skin-10. Figure 1 and Figure 2 show the clustering results for TIKmeans using different parameters on datasets Skin-5 and

Skin-10, respectively. In each figure, "epsilon" represents the parameter ϵ, "eta" represents the parameter η, and "delta" represents the parameter δ.

It can be seen that, in general, larger ϵ and η or smaller δ lead to a faster implementation of TIKmeans algorithm. Smaller ϵ and η achieve better clustering quality while δ does not influence the clustering quality much.

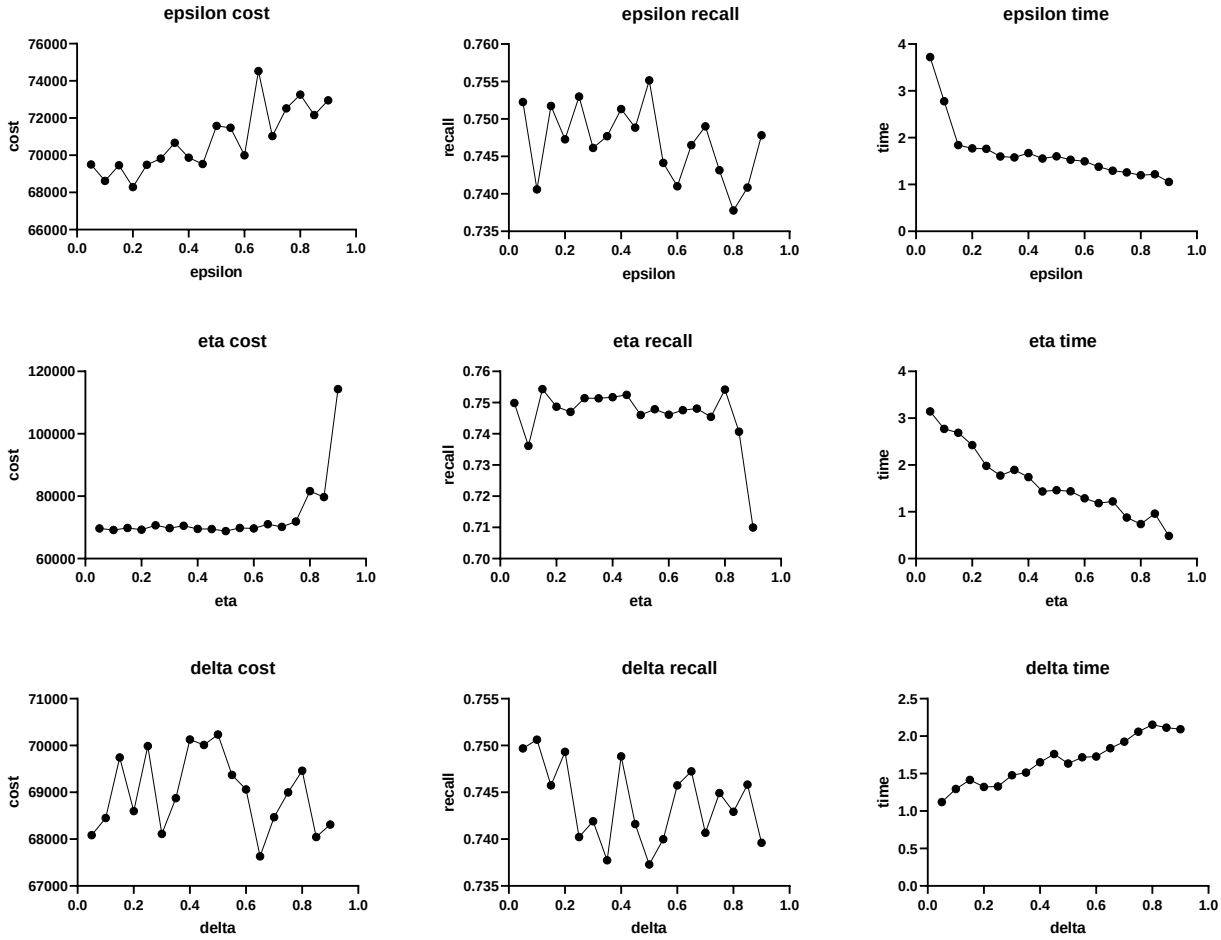

Figure 1. Clustering results for TIKmeans using different parameters on dataset Skin-5

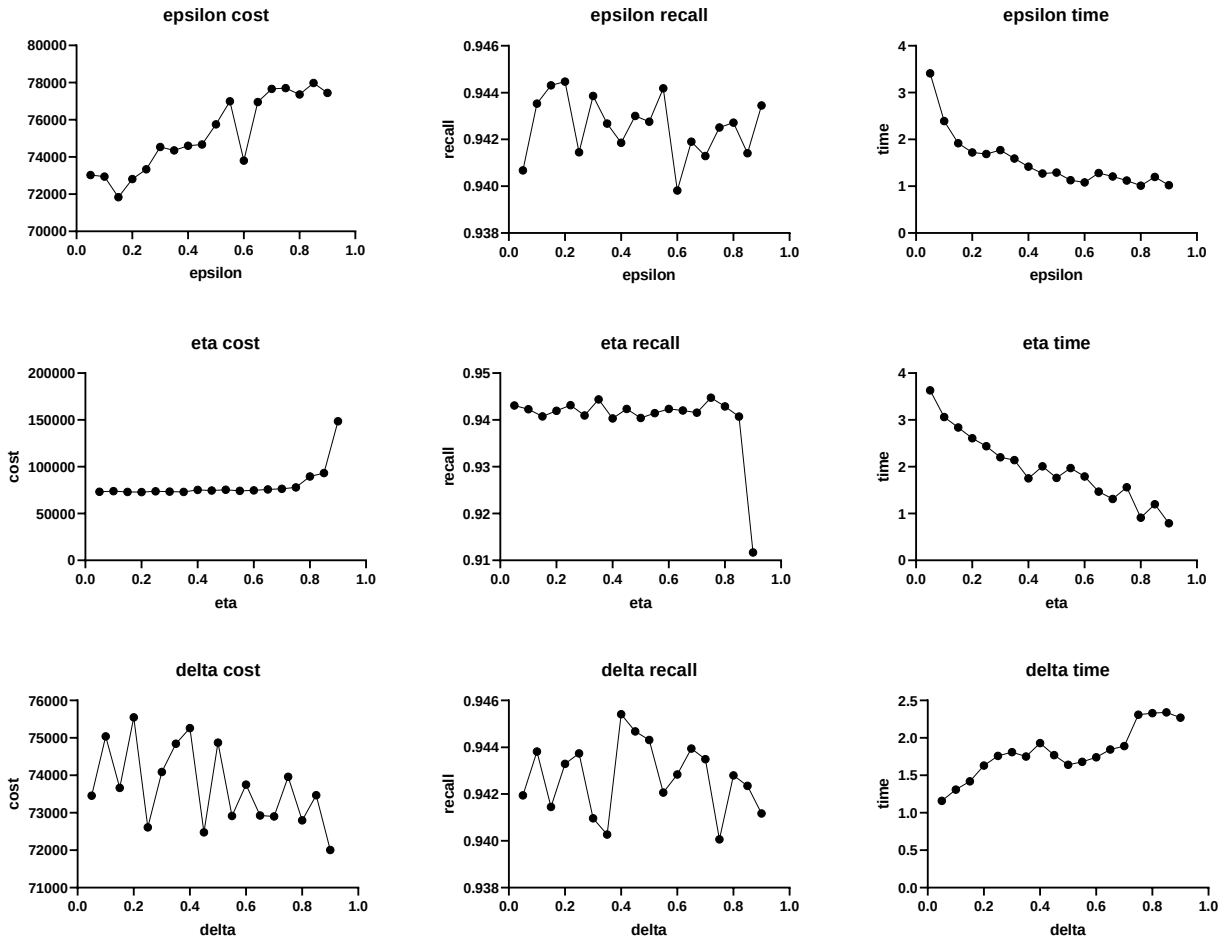

Figure 2. Clustering results for TIKmeans using different parameters on dataset Skin-10

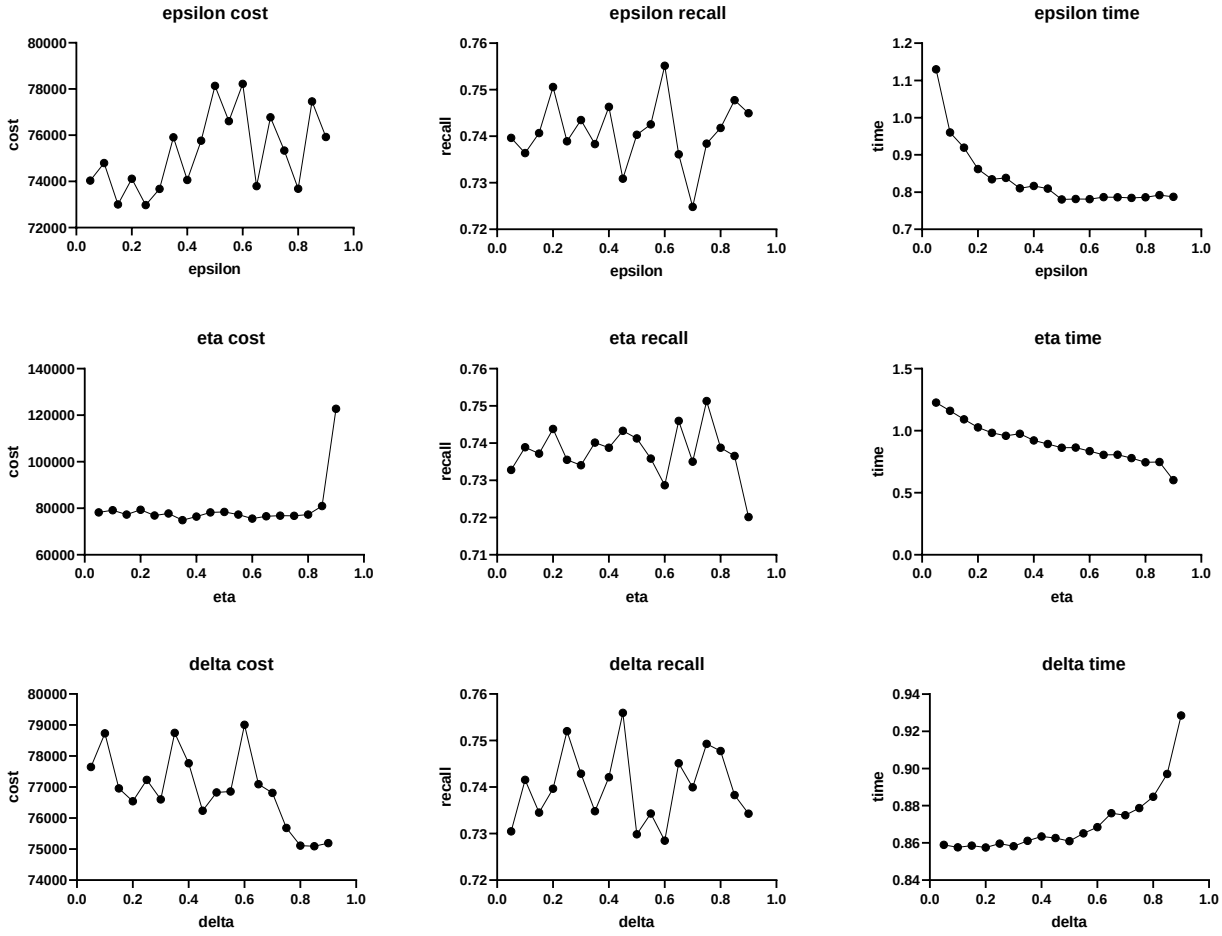

Figure 3. Clustering results for IKmeans using different parameters on dataset Skin-5

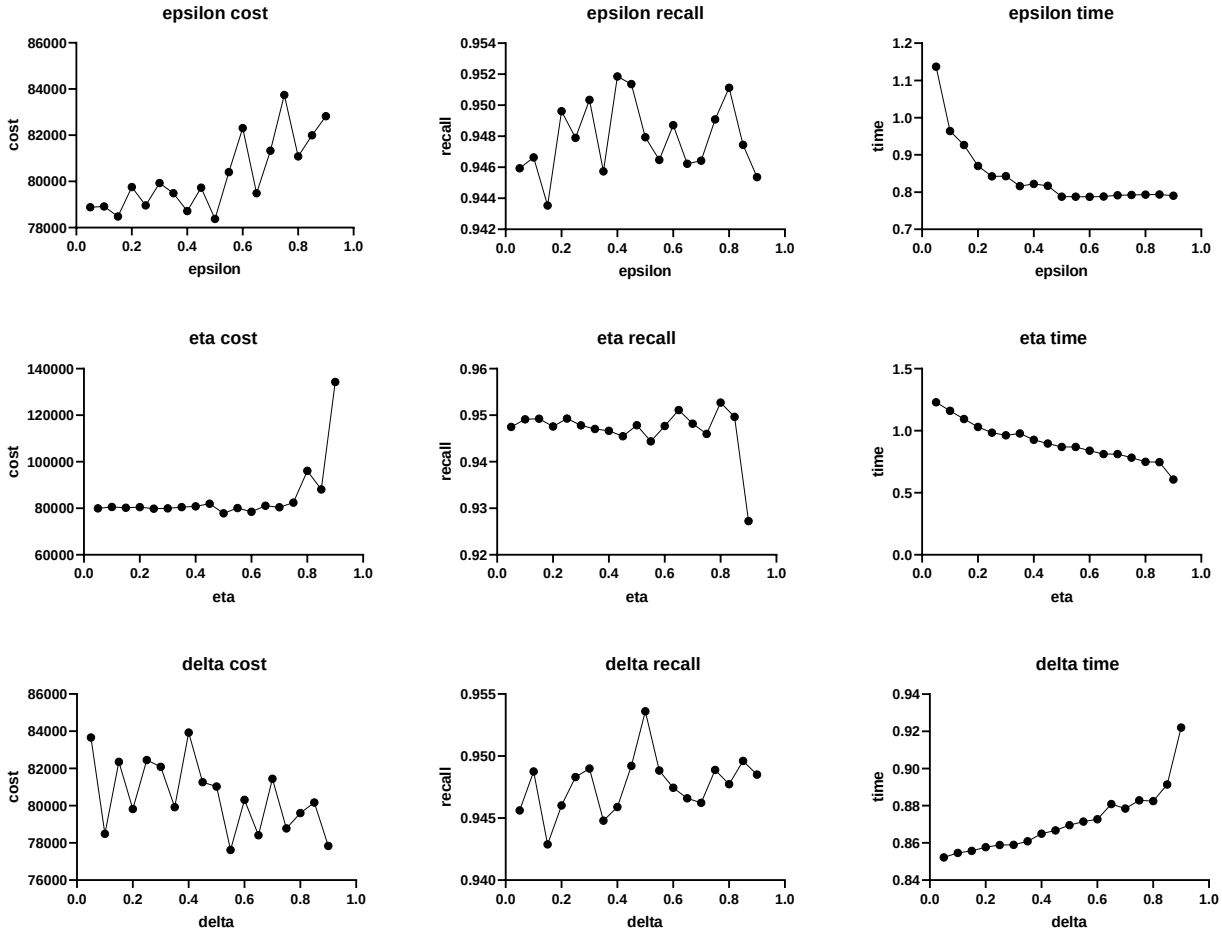

Figure 4. Clustering results for IKmeans using different parameters on dataset Skin-10

Figure 3 and Figure 4 show the clustering results for IKmeans using different parameters on datasets Skin-5 and Skin-10, respectively. It can be seen that, in general, larger ϵ and η or smaller δ lead to a faster implementation of IKmeans. Smaller ϵ and η achieve better clustering quality while δ does not influence the clustering quality much.

C.2. Experimental Performances with Varying Number of Centers Opened and Outliers Discarded

In this section, we present complementary experiments on the performance of our proposed sampling-based algorithms with different choices of k and z to show the parameter robustness of different algorithms. In general, for a particular dataset, the number of clusters k and the number of outliers z do not influence the performance of our proposed algorithms. In the following, we present results for more choices of k and z on different datasets, where k ranges from 10 to 50 and z ranges from $1\%n$ to $10\%n$, to demonstrate the performance of our proposed methods compared with other algorithms. For the LS++ algorithm, the running time increases significantly as the data size and the number of centers k grow. Since for most datasets, LS++ fails to return a solution within 24 hours, we only provide comparison results for other algorithms. For the SIFT dataset, since the data size is much larger than other datasets, moderate changes in k do not significantly influence the clustering cost. Thus, we test the values of k ranging from 50 to 200 on SIFT.

It can be seen from the tables that, for all algorithms on a particular dataset, a larger k leads to a lower clustering cost with higher recall and running time, whereas a larger z leads to a lower clustering cost and higher recall. Different settings of z do not influence the running time of the proposed algorithms. Experimental results show that our proposed TIKmeans algorithm achieves the best clustering quality and recall for most cases, while our proposed IKmeans algorithm is the fastest compared with other algorithms.

Table 4. Clustering performance on dataset Skin-5 with varying k from 5 to 25 and fixed $z = 1\%n$.

Cost	$k = 5$	$k = 10$	$k = 15$	$k = 20$	$k = 25$
TIKmeans	133001.69	**69166.76**	**48687.18**	**37333.04**	**31424.09**
IKmeans	137438.69	69184.39	51028.29	37507.20	31650.40
RobustKmeans++	135760.39	70574.93	50079.34	38774.81	32833.94
NKmeans	**131279.93**	72237.72	51885.67	41251.32	33929.32
Recall	$k = 5$	$k = 10$	$k = 15$	$k = 20$	$k = 25$
TIKmeans	0.7669	0.7413	0.7310	0.7895	0.7774
IKmeans	0.7645	**0.7625**	0.7331	**0.8291**	**0.8190**
RobustKmeans++	0.7661	0.7609	**0.7613**	0.7314	0.7948
NKmeans	**0.7689**	0.7197	0.7310	0.7633	0.7883
Time	$k = 5$	$k = 10$	$k = 15$	$k = 20$	$k = 25$
TIKmeans	1.39	2.37	3.43	5.03	5.78
IKmeans	**0.42**	**0.77**	**1.13**	**1.46**	**1.85**
RobustKmeans++	0.47	0.81	1.38	1.47	1.88
NKmeans	3.29	13.92	31.46	57.48	93.74

Table 5. Clustering performance on dataset Skin-5 with varying k from 30 to 50 and fixed $z = 1\%n$.

Cost	$k = 30$	$k = 35$	$k = 40$	$k = 45$	$k = 50$
TIKmeans	28214.01	**23124.98**	**20217.43**	**18980.59**	17282.84
IKmeans	**26712.49**	23594.32	22403.47	19352.91	**16983.94**
RobustKmeans++	27460.27	24615.45	22473.25	20754.49	19098.23
NKmeans	28309.55	25418.44	22343.45	21518.82	19582.24
Recall	$k = 30$	$k = 35$	$k = 40$	$k = 45$	$k = 50$
TIKmeans	0.7859	0.7920	0.8287	0.8162	0.8376
IKmeans	0.8045	0.7992	**0.8323**	**0.8465**	**0.8441**
RobustKmeans++	0.7903	0.8307	0.7996	0.8255	0.8194
NKmeans	**0.8610**	**0.8319**	0.7843	0.8089	0.7580
Time	$k = 30$	$k = 35$	$k = 40$	$k = 45$	$k = 50$
TIKmeans	6.96	7.87	9.70	10.16	11.99
IKmeans	**2.13**	**2.51**	**2.89**	**3.32**	**3.78**
RobustKmeans++	2.31	2.54	2.97	3.52	3.88
NKmeans	140.62	194.56	258.30	328.05	424.09

Table 6. Clustering performance on dataset Skin-10 with varying k from 5 to 25 and fixed $z = 1\%n$.

Cost	$k = 5$	$k = 10$	$k = 15$	$k = 20$	$k = 25$
TIKmeans	**132588.34**	74153.30	49329.10	**35996.06**	**31346.05**
IKmeans	138698.15	**72951.08**	50072.42	37135.89	32602.12
RobustKmeans++	137933.29	77644.30	50830.38	47606.80	35945.20
NKmeans	137561.83	78409.65	**48636.44**	40860.58	34571.45
Recall	$k = 5$	$k = 10$	$k = 15$	$k = 20$	$k = 25$
TIKmeans	0.9510	0.9347	**0.9408**	0.9449	**0.9702**
IKmeans	**0.9551**	0.9404	**0.9408**	0.9689	0.9408
RobustKmeans++	0.9543	**0.9640**	0.9326	0.9485	0.9624
NKmeans	0.9514	0.9587	0.9400	**0.9730**	0.9534
Time	$k = 5$	$k = 10$	$k = 15$	$k = 20$	$k = 25$
TIKmeans	1.45	2.48	3.87	4.77	6.00
IKmeans	**0.43**	**0.60**	**0.77**	**0.95**	**1.13**
RobustKmeans++	0.51	0.83	1.16	1.46	1.81
NKmeans	3.45	13.3	31.56	57.79	93.77

Table 7. Clustering performance on dataset Skin-10 with varying k from 30 to 50 and fixed $z = 1\%n$.

Cost	$k = 30$	$k = 35$	$k = 40$	$k = 45$	$k = 50$
TIKmeans	**28279.90**	**24872.02**	**21010.90**	20562.72	**18187.60**
IKmeans	28885.94	26696.04	22985.64	**20217.50**	18720.93
RobustKmeans++	36015.10	30775.52	28880.69	28194.32	25787.91
NKmeans	32783.28	35675.38	33097.31	27866.86	28763.89
Recall	$k = 30$	$k = 35$	$k = 40$	$k = 45$	$k = 50$
TIKmeans	0.9681	**0.9657**	**0.9685**	**0.9555**	0.9481
IKmeans	**0.9685**	0.9367	0.9632	0.9538	**0.9601**
RobustKmeans++	0.9379	0.9469	0.9469	0.9371	0.9245
NKmeans	0.9571	0.9220	0.9020	0.9404	0.9175
Time	$k = 30$	$k = 35$	$k = 40$	$k = 45$	$k = 50$
TIKmeans	6.83	8.13	9.28	10.19	12.21
IKmeans	**1.30**	**1.45**	**1.62**	**1.81**	**2.01**
RobustKmeans++	2.18	2.55	2.94	3.38	3.87
NKmeans	139.88	192.47	258.66	332.46	418.80

Table 8. Clustering performance on dataset SUSY-5 with varying k from 5 to 25 and fixed $z = 1\%n$.

Cost	$k = 5$	$k = 10$	$k = 15$	$k = 20$	$k = 25$
TIKmeans	76506571.19	**63303490.50**	**58241195.75**	**53150916.80**	**50012984.14**
IKmeans	**75161216.13**	64384761.92	59031750.53	54601183.21	52000973.94
RobustKmeans++	75363995.40	64744175.91	58799082.14	53998470.07	50930055.07
NKmeans	78044742.48	68129835.21	60729749.41	56443827.61	52409638.62
Recall	$k = 5$	$k = 10$	$k = 15$	$k = 20$	$k = 25$
TIKmeans	0.6683	**0.6985**	**0.7254**	**0.8252**	0.7407
IKmeans	0.6334	0.6069	0.7157	0.8001	**0.8345**
RobustKmeans++	**0.7242**	0.6802	0.7155	0.7353	0.7840
NKmeans	0.6932	0.6845	0.7144	0.6959	0.7881
Time	$k = 5$	$k = 10$	$k = 15$	$k = 20$	$k = 25$
TIKmeans	47.38	60.12	74.01	87.85	102.24
IKmeans	**20.31**	**29.37**	**39.05**	**48.79**	**58.1**
RobustKmeans++	26.45	50.57	71.14	97.59	117.41
NKmeans	81.93	254.23	675.83	1391.35	2582.76

Table 9. Clustering performance on dataset SUSY-5 with varying k from 30 to 50 and fixed $z = 1\%n$.

Cost	$k = 30$	$k = 35$	$k = 40$	$k = 45$	$k = 50$
TIKmeans	**47359631**	**44867910**	**43604086**	**43269023**	**41477222**
IKmeans	49054613	46938023	44634763	44289220	42737899
RobustKmeans++	48363286	46465226	45437418	43819938	43273159
NKmeans	49116640	47631174	45828267	44637656	43793074
Recall	$k = 30$	$k = 35$	$k = 40$	$k = 45$	$k = 50$
TIKmeans	0.8207	0.7990	0.7788	0.7910	0.8016
IKmeans	**0.8226**	0.8301	0.8237	**0.8603**	0.8555
RobustKmeans++	0.8194	0.8106	**0.8518**	0.8292	**0.8651**
NKmeans	0.7826	**0.8522**	0.7880	0.8261	0.8069
Time	$k = 30$	$k = 35$	$k = 40$	$k = 45$	$k = 50$
TIKmeans	92.61	105.76	119.01	127.67	141.07
IKmeans	**65.67**	**77.93**	**88.63**	**95.03**	**109.72**
RobustKmeans++	137.39	169.31	188.05	207.05	226.71
NKmeans	4292.79	6514.81	9355.65	12626.07	17441.47

Table 10. Clustering performance on dataset SUSY-10 with varying k from 5 to 25 and fixed $z = 1\%n$.

Cost	$k = 5$	$k = 10$	$k = 15$	$k = 20$	$k = 25$
TIKmeans	**76931998.91**	**65170740.90**	61115030.52	**54374376.30**	**51543466.66**
IKmeans	79990751.06	65825639.96	**58880753.99**	55494181.80	52662368.24
RobustKmeans++	80757039.16	65803549.89	59247264.81	55862554.74	51548007.32
NKmeans	78600699.75	67033340.54	60664573.60	55148867.68	53220072.49
Recall	$k = 5$	$k = 10$	$k = 15$	$k = 20$	$k = 25$
TIKmeans	0.9717	0.9652	0.9743	0.9808	**0.9880**
IKmeans	**0.9770**	0.9694	0.9776	0.9857	0.9809
RobustKmeans++	0.9725	**0.9799**	0.9780	**0.9880**	0.9860
NKmeans	0.9594	0.9640	**0.9907**	0.9843	0.9832
Time	$k = 5$	$k = 10$	$k = 15$	$k = 20$	$k = 25$
TIKmeans	47.45	60.72	73.93	88.07	101.84
IKmeans	**19.89**	**28.77**	**38.17**	**46.99**	**56.28**
RobustKmeans++	26.24	49.75	70.42	96.90	116.11
NKmeans	82.46	257.29	684.65	1353.63	2567.51

Table 11. Clustering performance on SUSY-10 with varying k from 30 to 50 and fixed $z = 1\%n$.

Cost	$k = 30$	$k = 35$	$k = 40$	$k = 45$	$k = 50$
TIKmeans	49090571	**45972711**	**44823068**	**44021010**	**42376519**
IKmeans	**48756523**	47809234	45925463	44255460	42688771
RobustKmeans++	49964920	46431096	45739430	44035489	43635342
NKmeans	49586439	47269972	46004081	44936911	43416153
Recall	$k = 30$	$k = 35$	$k = 40$	$k = 45$	$k = 50$
TIKmeans	**0.9873**	0.9880	0.9875	0.9874	0.9826
IKmeans	0.9849	0.9790	0.9837	**0.9881**	**0.9853**
RobustKmeans++	0.9852	**0.9894**	**0.9888**	0.9850	0.9847
NKmeans	0.9829	0.9851	0.9840	0.9814	0.9744
Time	$k = 30$	$k = 35$	$k = 40$	$k = 45$	$k = 50$
TIKmeans	91.85	98.04	112.08	124.92	137.07
IKmeans	**66.54**	**72.09**	**83.32**	**97.16**	**105.72**
RobustKmeans++	136.71	165.12	185.23	204.27	227.32
NKmeans	4329.1	6565.01	9306.71	13138.28	17744.28

Table 12. Clustering performance on dataset SIFT-5 with varying k from 50 to 200 and fixed $z = 1\%n$.

Cost	$k = 50$	$k = 100$	$k = 150$	$k = 200$
TIKmeans	**13005055711**	**12033800256**	**11446843957**	**11217336739**
IKmeans	13172627781	12275654391	11642167619	11224453041
RobustKmeans++	13095090316	12077778481	11493088419	NA
NKmeans	NA	NA	NA	NA
Recall	$k = 50$	$k = 100$	$k = 150$	$k = 200$
TIKmeans	**1**	**1**	**1**	**1**
IKmeans	**1**	**1**	0.9999	**1**
RobustKmeans++	**1**	0.9998	**1**	NA
NKmeans	NA	NA	NA	NA
Time	$k = 50$	$k = 100$	$k = 150$	$k = 200$
TIKmeans	8020.93	10906.2	17308.9	25570.99
IKmeans	**5243.94**	**9635.4**	**10122.39**	**13091.47**
RobustKmeans++	47169.19	92840.3	151129.76	$>$ 72h
NKmeans	$>$ 72h	$>$ 72h	$>$ 72h	$>$ 72h

Table 13. Clustering performance on SIFT-10 with varying k from 50 to 200 and fixed $z = 1\%n$.

Cost	$k = 50$	$k = 100$	$k = 150$	$k = 200$
TIKmeans	13009053091	12081741573	**11470728229**	**11212034557**
IKmeans	13192544237	12107041166	11582239578	11395892613
RobustKmeans++	**12924333298**	**11983478657**	11510477529	NA
NKmeans	NA	NA	NA	NA
Recall	$k = 50$	$k = 100$	$k = 150$	$k = 200$
TIKmeans	**1**	**1**	**1**	**1**
IKmeans	**1**	**1**	**1**	**1**
RobustKmeans++	**1**	**1**	**1**	NA
NKmeans	NA	NA	NA	NA
Time	$k = 50$	$k = 100$	$k = 150$	$k = 200$
TIKmeans	8775.9	10733.8	16845.2	25682.38
IKmeans	**5292.4**	**9374.2**	**10828.9**	**13857.4**
RobustKmeans++	48360.4	92223.8	159366.4	$>$ 72h
NKmeans	$>$ 72h	$>$ 72h	$>$ 72h	$>$ 72h

Table 14. Clustering performance on Skin-5 with varying z from $1\%n$ to $5\%n$ and fixed $k = 10$.

Cost	$z = 1\%n$	$z = 2\%n$	$z = 3\%n$	$z = 4\%n$	$z = 5\%n$
TIKmeans	**64786.31**	**65100.22**	**65720.50**	68458.42	**60736.34**
IKmeans	71871.21	68508.17	66279.40	**65510.93**	65366.71
RobustKmeans++	72139.69	69895.30	73838.02	66695.92	66386.79
NKmeans	75964.88	70560.98	73342.50	66461.91	74162.59
Recall	$z = 1\%n$	$z = 2\%n$	$z = 3\%n$	$z = 4\%n$	$z = 5\%n$
TIKmeans	0.7392	0.7843	0.7830	0.8137	0.8390
IKmeans	0.7613	**0.7894**	0.7988	0.8173	**0.8605**
RobustKmeans++	**0.7792**	0.7823	0.7797	0.8240	0.8435
NKmeans	0.7078	0.7651	**0.8060**	**0.8292**	0.8394
Time	$z = 1\%n$	$z = 2\%n$	$z = 3\%n$	$z = 4\%n$	$z = 5\%n$
TIKmeans	2.27	2.32	2.28	2.38	2.31
IKmeans	**0.61**	**0.61**	**0.61**	**0.62**	**0.63**
RobustKmeans++	0.83	0.84	0.84	0.85	0.86
NKmeans	13.46	7.4	5.48	4.49	3.9

Table 15. Clustering performance on Skin-5 with varying z from $6\%n$ to $10\%n$ and fixed $k = 10$.

Cost	$z = 6\%n$	$z = 7\%n$	$z = 8\%n$	$z = 9\%n$	$z = 10\%n$
TIKmeans	63506.86	**62947.58**	**61230.81**	**63100.28**	**59344.19**
IKmeans	**61824.42**	66127.58	62570.48	65745.74	61660.30
RobustKmeans++	67136.44	69719.64	67068.58	65594.05	68895.94
NKmeans	78933.56	80379.68	73653.64	77361.83	73040.27
Recall	$z = 6\%n$	$z = 7\%n$	$z = 8\%n$	$z = 9\%n$	$z = 10\%n$
TIKmeans	0.8584	**0.8667**	**0.8853**	**0.8905**	0.8857
IKmeans	0.8514	0.8647	0.8798	0.8830	**0.8915**
RobustKmeans++	**0.8590**	0.8606	0.8454	0.8810	0.8818
NKmeans	0.8526	0.8541	0.8756	0.8625	0.8888
Time	$z = 6\%n$	$z = 7\%n$	$z = 8\%n$	$z = 9\%n$	$z = 10\%n$
TIKmeans	2.39	2.59	2.46	2.57	2.53
IKmeans	**0.73**	**0.82**	**0.74**	**0.75**	**0.76**
RobustKmeans++	0.78	0.84	0.78	0.81	0.81
NKmeans	3.53	3.21	3.02	2.93	2.77

Table 16. Clustering performance on Skin-10 with varying z from $1\%n$ to $5\%n$ and fixed $k = 10$.

Cost	$z = 1\%n$	$z = 2\%n$	$z = 3\%n$	$z = 4\%n$	$z = 5\%n$
TIKmeans	74015.60	74219.05	**68796.43**	**68124.14**	**75247.27**
IKmeans	76897.91	**70525.45**	76010.99	76636.38	89996.15
RobustKmeans++	77700.52	79382.63	79933.22	81708.13	84753.06
NKmeans	**73716.44**	75465.23	75226.64	76592.18	75577.78
Recall	$z = 1\%n$	$z = 2\%n$	$z = 3\%n$	$z = 4\%n$	$z = 5\%n$
TIKmeans	0.9498	0.9443	0.9528	**0.9563**	**0.9542**
IKmeans	0.9465	0.9502	**0.9548**	0.9463	0.9462
RobustKmeans++	**0.9526**	0.9563	0.9488	0.9446	0.9529
NKmeans	0.9424	**0.9620**	0.9537	0.9553	0.9533
Time	$z = 1\%n$	$z = 2\%n$	$z = 3\%n$	$z = 4\%n$	$z = 5\%n$
TIKmeans	2.60	2.50	2.73	2.60	2.44
IKmeans	**0.78**	**0.76**	**0.79**	**0.76**	**0.75**
RobustKmeans++	0.83	0.83	0.84	0.85	0.86
NKmeans	13.44	7.48	5.52	4.56	4.02

Table 17. Clustering performance on Skin-10 with varying z from $6\%n$ to $10\%n$ and fixed $k = 10$.

Cost	$z = 6\%n$	$z = 7\%n$	$z = 8\%n$	$z = 9\%n$	$z = 10\%n$
TIKmeans	**69275.11**	**65275.5**	70124.64	**69053.54**	**66495.53**
IKmeans	77158.66	74924.31	**69899.49**	76236.49	72217.37
RobustKmeans++	82217.23	82915.68	79387.25	76790.36	87849.24
NKmeans	80888.75	78856.59	80999.44	76502.57	78045.86
Recall	$z = 6\%n$	$z = 7\%n$	$z = 8\%n$	$z = 9\%n$	$z = 10\%n$
TIKmeans	0.961	0.9658	0.9694	0.9637	0.972
IKmeans	0.9617	0.9647	0.9677	**0.9689**	0.9702
RobustKmeans++	0.9604	0.958	0.9602	0.962	0.9616
NKmeans	**0.9651**	**0.9665**	**0.9702**	0.9629	**0.9751**
Time	$z = 6\%n$	$z = 7\%n$	$z = 8\%n$	$z = 9\%n$	$z = 10\%n$
TIKmeans	2.37	2.39	2.40	2.32	2.36
IKmeans	**0.51**	**0.59**	**0.64**	**0.69**	**0.67**
RobustKmeans++	0.87	0.88	0.89	0.9	0.9
NKmeans	3.62	3.37	3.14	2.97	2.89

Table 18. Clustering performance on SUSY-5 with varying z from $1\%n$ to $5\%n$ and fixed $k = 10$.

Cost	$z = 1\%n$	$z = 2\%n$	$z = 3\%n$	$z = 4\%n$	$z = 5\%n$
TIKmeans	**62758784.43**	**63910448.61**	**64388795.91**	62495299.04	**62697556.27**
IKmeans	65783225.59	64912207.36	65246163.32	64642186.03	63671705.04
RobustKmeans++	64453941.70	65069784.16	65287913.29	**61526407.40**	64922196.82
NKmeans	64749757.12	64832980.44	65402357.48	64499515.06	64709049.91
Recall	$z = 1\%n$	$z = 2\%n$	$z = 3\%n$	$z = 4\%n$	$z = 5\%n$
TIKmeans	0.7363	0.8068	0.8444	0.8830	0.8917
IKmeans	0.7215	0.8015	0.8552	0.8902	0.9008
RobustKmeans++	**0.7690**	**0.8157**	**0.8801**	**0.9167**	**0.9315**
NKmeans	0.6890	0.7925	0.8657	0.8949	0.9172
Time	$z = 1\%n$	$z = 2\%n$	$z = 3\%n$	$z = 4\%n$	$z = 5\%n$
TIKmeans	60.84	59.9	60.35	60.88	61.26
IKmeans	**29.64**	**29.3**	**29.16**	**29.31**	**29.73**
RobustKmeans++	50.48	50.63	51.1	51.48	51.95
NKmeans	256.25	171.86	144.82	128.76	121.89

Table 19. Clustering performance on SUSY-5 with varying z from $6\%n$ to $10\%n$ and fixed $k = 10$.

Cost	$z = 6\%n$	$z = 7\%n$	$z = 8\%n$	$z = 9\%n$	$z = 10\%n$
TIKmeans	**62979798.60**	**64022621.01**	63048475.88	63637522.62	65435146.73
IKmeans	64829133.51	65376733.61	66384399.56	66896282.84	65847357.87
RobustKmeans++	64333574.10	64053981.19	**62868433.36**	**63604037.88**	**63829121.16**
NKmeans	64752980.43	65808864.54	63748304.73	65423397.85	64790008.58
Recall	$z = 6\%n$	$z = 7\%n$	$z = 8\%n$	$z = 9\%n$	$z = 10\%n$
TIKmeans	0.9202	0.9219	0.9220	0.9284	0.9411
IKmeans	0.8997	0.9130	0.9364	0.9154	0.9451
RobustKmeans++	**0.9217**	**0.9388**	0.9206	**0.9369**	**0.9483**
NKmeans	0.9192	0.9043	**0.9482**	0.9209	0.9475
Time	$z = 6\%n$	$z = 7\%n$	$z = 8\%n$	$z = 9\%n$	$z = 10\%n$
TIKmeans	62.95	63.23	63.55	64.2	64.18
IKmeans	**30.42**	**30.53**	**30.78**	**31.16**	**31.32**
RobustKmeans++	52.97	52.89	53.72	54.03	54.84
NKmeans	116.46	112.68	110.36	108.16	105.3

Table 20. Clustering performance on SUSY-10 with varying z from $1\%n$ to $5\%n$ and fixed $k = 10$.

Cost	$z = 1\%n$	$z = 2\%n$	$z = 3\%n$	$z = 4\%n$	$z = 5\%n$
TIKmeans	65444363.61	**66093452.48**	68542989.95	66803669.72	**63516819.42**
IKmeans	66435461.94	66635419.50	**64357041.09**	67485941.66	68153278.76
RobustKmeans++	66317811.28	66572096.45	65874645.76	65812489.59	63962275.52
NKmeans	**64590541.36**	66405391.38	65267405.61	**65728455.77**	67356351.80
Recall	$z = 1\%n$	$z = 2\%n$	$z = 3\%n$	$z = 4\%n$	$z = 5\%n$
TIKmeans	0.9695	0.9811	**0.9912**	0.9899	0.9921
IKmeans	0.9734	0.9855	0.989	**0.9948**	0.9915
RobustKmeans++	**0.9746**	0.9811	0.9866	0.9898	**0.9946**
NKmeans	0.9674	**0.9907**	0.9875	0.9917	0.9891
Time	$z = 1\%n$	$z = 2\%n$	$z = 3\%n$	$z = 4\%n$	$z = 5\%n$
TIKmeans	59.78	59.7	60.15	60.43	61.03
IKmeans	**29.34**	**28.66**	**28.54**	**28.46**	**28.69**
RobustKmeans++	50.78	49.99	50.6	50.92	51.54
NKmeans	255.55	173.77	146.58	131.46	123.89

Table 21. Clustering performance on SUSY-10 with varying z from $6\%n$ to $10\%n$ and fixed $k = 10$.

Cost	$z = 6\%n$	$z = 7\%n$	$z = 8\%n$	$z = 9\%n$	$z = 10\%n$
TIKmeans	**64994013.81**	**64502595.20**	**64081521.71**	**66273986.19**	**65503933.45**
IKmeans	67337311.62	67766754.31	69849715.71	68923763.38	67159027.39
RobustKmeans++	66885305.76	67675644.04	65113556.81	66385763.36	66425920.69
NKmeans	65804048.90	66175936.65	64753806.68	66404345.02	65523697.38
Recall	$z = 6\%n$	$z = 7\%n$	$z = 8\%n$	$z = 9\%n$	$z = 10\%n$
TIKmeans	0.9918	0.9922	**0.9951**	0.9949	0.9936
IKmeans	**0.9937**	0.9934	0.9950	**0.9954**	**0.9965**
RobustKmeans++	0.9907	0.9926	0.9946	0.9945	0.9947
NKmeans	0.9933	**0.9950**	0.9936	0.9938	0.9963
Time	$z = 6\%n$	$z = 7\%n$	$z = 8\%n$	$z = 9\%n$	$z = 10\%n$
TIKmeans	62.44	63.12	62.92	63.47	63.96
IKmeans	**29.25**	**29.32**	**29.54**	**29.61**	**30.55**
RobustKmeans++	52.1	52.5	52.99	53.6	54.51
NKmeans	116.5	113.34	111.21	107.86	106.51

Table 22. Clustering performance on SIFT-5 with varying z from $1\%n$ to $5\%n$ and fixed $k = 100$.

Cost	$z = 1\%n$	$z = 2\%n$	$z = 3\%n$	$z = 4\%n$	$z = 5\%n$
TIKmeans	**12033800256**	**1222361151**	**12109199438**	**11984006794**	**11979987277**
IKmeans	12275654391	1230394902	12125780107	12070117144	12068091628
RobustKmeans++	12077778481	1223089808	12143632947	12070268466	12059086576
NKmeans	NA	NA	NA	NA	NA
Recall	$z = 1\%n$	$z = 2\%n$	$z = 3\%n$	$z = 4\%n$	$z = 5\%n$
TIKmeans	**1**	**1**	**1**	**1**	**1**
IKmeans	**1**	**1**	**1**	**1**	**1**
RobustKmeans++	0.9998	**1**	**1**	**1**	**1**
NKmeans	NA	NA	NA	NA	NA
Time	$z = 1\%n$	$z = 2\%n$	$z = 3\%n$	$z = 4\%n$	$z = 5\%n$
TIKmeans	10870.6	11055.2	11859.4	11424.1	11965.7
IKmeans	**9635.4**	**9372.0**	**9837.5**	**9695.3**	**9320.7**
RobustKmeans++	92840	95168	93781	94487	96596
NKmeans	$>$ 72h	$>$ 72h	$>$ 72h	$>$ 72h	$>$ 72h

Table 23. Clustering performance on SIFT-5 with varying z from $6\%n$ to $10\%n$ and fixed $k = 100$.

Cost	$z = 6\%n$	$z = 7\%n$	$z = 8\%n$	$z = 9\%n$	$z = 10\%n$
TIKmeans	**11970682447**	**11981855317**	**11984096251**	12011016446	**11986722152**
IKmeans	12049816240	12002497596	12010444072	12039251997	12014269554
RobustKmeans++	12022383906	12000769418	11990243429	**12008302455**	11989202199
NKmeans	NA	NA	NA	NA	NA
Recall	$z = 6\%n$	$z = 7\%n$	$z = 8\%n$	$z = 9\%n$	$z = 10\%n$
TIKmeans	**1**	**1**	**1**	**1**	**1**
IKmeans	**1**	**1**	**1**	**1**	**1**
RobustKmeans++	**1**	**1**	**1**	**1**	**1**
NKmeans	NA	NA	NA	NA	NA
Time	$z = 6\%n$	$z = 7\%n$	$z = 8\%n$	$z = 9\%n$	$z = 10\%n$
TIKmeans	11413.0	10664.8	11637.3	11757.1	11696.7
IKmeans	**9705.1**	**9240.8**	**9929.6**	**9772.9**	**9567.2**
RobustKmeans++	95260	97268	99922	94212	96506
NKmeans	$>$ 72h	$>$ 72h	$>$ 72h	$>$ 72h	$>$ 72h

Table 24. Clustering performance on SIFT-10 with varying z from $1\%n$ to $5\%n$ and fixed $k = 100$.

Cost	$z = 1\%n$	$z = 2\%n$	$z = 3\%n$	$z = 4\%n$	$z = 5\%n$
TIKmeans	12081741573	**12111370981**	12145140135	**12090385992**	**12088781032**
IKmeans	12107041166	12135208822	12163379429	12198221417	12153404554
RobustKmeans++	**11983478657**	12222488921	**12131881316**	12109811037	12136491955
NKmeans	NA	NA	NA	NA	NA
Recall	$z = 1\%n$	$z = 2\%n$	$z = 3\%n$	$z = 4\%n$	$z = 5\%n$
TIKmeans	**1**	**1**	**1**	**1**	**1**
IKmeans	**1**	**1**	**1**	**1**	**1**
RobustKmeans++	**1**	**1**	**1**	**1**	**1**
NKmeans	NA	NA	NA	NA	NA
Time	$z = 1\%n$	$z = 2\%n$	$z = 3\%n$	$z = 4\%n$	$z = 5\%n$
TIKmeans	10733.8	11656.6	12101.4	12323.1	10959.4
IKmeans	**9374.2**	**9110.8**	**9817.9**	**9501.3**	**9240.4**
RobustKmeans++	92223	93572	94779	95051	93979
NKmeans	$>$ 72h	$>$ 72h	$>$ 72h	$>$ 72h	$>$ 72h

Table 25. Clustering performance on SIFT-10 with varying z from $6\%n$ to $10\%n$ and fixed $k = 100$.

Cost	$z = 6\%n$	$z = 7\%n$	$z = 8\%n$	$z = 9\%n$	$z = 10\%n$
TIKmeans	**12091294505**	**12046590461**	**12070090465**	**12023308618**	**12021583648**
IKmeans	12107895648	12088364369	12092477387	12042054652	12041631539
RobustKmeans++	12119472576	12094581099	12095366699	12063952102	12037320094
NKmeans	NA	NA	NA	NA	NA
Recall	$z = 6\%n$	$z = 7\%n$	$z = 8\%n$	$z = 9\%n$	$z = 10\%n$
TIKmeans	**1**	**1**	**1**	**1**	**1**
IKmeans	**1**	**1**	**1**	**1**	**1**
RobustKmeans++	**1**	**1**	**1**	**1**	**1**
NKmeans	NA	NA	NA	NA	NA
Time	$z = 6\%n$	$z = 7\%n$	$z = 8\%n$	$z = 9\%n$	$z = 10\%n$
TIKmeans	10867.1	10805.4	11058.9	10921.1	11770.4
IKmeans	**9480.8**	**9575.4**	**9135.7**	**9694.2**	**9907.9**
RobustKmeans++	97072	94623	100606	95736	99554
NKmeans	$>$ 72h	$>$ 72h	$>$ 72h	$>$ 72h	$>$ 72h

C.3. Experimental Performances on the Relaxed Number of Centers Opened

In this section, we provide more results on the relation between different sizes of the centers selected by our sampling methods and the final recall in the following table. We set the number of centers selected during the sampling process as βk for β ranging from 1 to 8 to test the performance of our proposed methods. It can be seen that the number of centers opened does not significantly influence the recall index if the noise added is far away from the inliers (Skin-10 and SUSY-10). In this case, opening more centers does not significantly change the designation of outliers. However, when the outliers are not far away from the true clusters, opening more centers leads to misclassification of outliers.

Table 26. Change of recall on different datasets by setting the number of centers sampled as βk.

Skin-5	$\beta = 1$	$\beta = 2$	$\beta = 3$	$\beta = 4$	$\beta = 5$	$\beta = 6$	$\beta = 7$	$\beta = 8$
TIKmeans	0.7654	0.7115	0.7580	0.7201	0.7294	0.7376	0.7299	0.7384
IKmeans	0.7662	0.7005	0.7503	0.7613	0.7588	0.7543	0.7621	0.7658
Skin-10	$\beta = 1$	$\beta = 2$	$\beta = 3$	$\beta = 4$	$\beta = 5$	$\beta = 6$	$\beta = 7$	$\beta = 8$
TIKmeans	0.9575	0.9543	0.9543	0.9375	0.9432	0.9424	0.9408	0.9404
IKmeans	0.9388	0.9506	0.9449	0.9408	0.9404	0.9400	0.9428	0.9375
SUSY-5	$\beta = 1$	$\beta = 2$	$\beta = 3$	$\beta = 4$	$\beta = 5$	$\beta = 6$	$\beta = 7$	$\beta = 8$
TIKmeans	0.7882	0.6764	0.7283	0.6080	0.6445	0.7979	0.7578	0.7407
IKmeans	0.6541	0.6099	0.7237	0.6907	0.5912	0.7184	0.7262	0.8124
SUSY-10	$\beta = 1$	$\beta = 2$	$\beta = 3$	$\beta = 4$	$\beta = 5$	$\beta = 6$	$\beta = 7$	$\beta = 8$
TIKmeans	0.9753	0.9753	0.9743	0.9730	0.9713	0.9690	0.9631	0.9649
IKmeans	0.9743	0.9648	0.9738	0.9687	0.9748	0.9646	0.9701	0.9706
Shuttle	$\beta = 1$	$\beta = 2$	$\beta = 3$	$\beta = 4$	$\beta = 5$	$\beta = 6$	$\beta = 7$	$\beta = 8$
TIKmeans	0.1176	0	0.3529	0	0.0588	0	0	0.3529
IKmeans	0.2941	0	0.2941	0.0588	0.1176	0.2941	0.2941	0.3529

C.4. Comparisons with Outlier Detection Methods

In this subsection, we give comparison results with other outlier detection methods. The outlier detection methods adopt two-stage strategies, where a filtering process is used to discard the suspect outliers followed by a clustering process on the remaining data points to obtain the final results. We choose the following algorithms to serve as the procedure to identify outliers, which are summarized as follows: (1) The probabilistic method in (Li et al., 2022) (ECO); (2) Two fast outlier detection methods in (Liu et al., 2008) (IFOREST) and (Sugiyama & Borgwardt, 2013) (Sampling). For the clustering process, we use the following methods: k-means++ (Arthur & Vassilvitskii, 2007) (the popular linear-time algorithm denoted as KM in the experiments, where we repeat for 10 times in our experiments to enhance the performances on k-means++) and LSDS++ (Fan et al., 2023) (the fastest constant-approximation algorithm).

We test the performances on 8 datasets, including Skin-5 (245,047 $\times$ 3), Skin-10 (245,047 $\times$ 3), SUSY-5 (5,000,000 $\times$ 18), SUSY-10 (5,000,000 $\times$ 18), HIGGS-5 (11,000,000 $\times$ 28), HIGGS-10 (11,000,000 $\times$ 28) [2], Shuttle (43,500 $\times$ 9) and KDDFULL (4,898,431 $\times$ 37) to demonstrate the scalability of the algorithms.

The results show that our IKmeans algorithm always achieves the best running time compared with other algorithms. Compared to "IFOREST+KM" and "Sampling+KM" methods, on average, our TIKmeans algorithm achieves at least 8% improvement in clustering cost. Although the clustering costs of our algorithms are not better than those of the "Sampling+LSDS++" method on some of the datasets, on average, IKmeans algorithm is at least 3 times faster than the "Sampling+LSDS++" method with better recall. The results reveal that the outlier detection step potentially compromises the accuracy of outlier identification. Moreover, our sampling-based algorithms achieve better accuracy in outlier identification with much faster running time compared to other two-step procedures based on local search methods.

Table 27. Comparison results with outlier detection methods.

Dataset	Method	Cost	Recall	Time(s)	Dataset	Method	Cost	Recall	Time(s)
Skin-5($k=10$)	TIKmeans	68049.49	0.7249	1.40	Skin-10($k=10$)	TIKmeans	**72448.37**	0.9313	1.34
	IKmeans	72714.88	0.7657	**0.59**		IKmeans	79923.51	0.9366	**0.57**
	ECOD+LSDS++	77295.27	0.7591	4.94		ECOD+LSDS+	77185.24	0.9410	4.54
	ECOD+KM	99187.17	0.7565	1.73		ECOD+KM	110350.79	0.9398	1.68
	IFOREST+LSDS++	71604.48	0.7645	15.95		IFOREST+LSDS++	77754.72	0.9418	21.37
	IFOREST+KM	87163.20	0.7560	14.21		IFOREST+KM	107361.52	0.9434	17.85
	Sampling+LSDS++	**64338.60**	0.7213	6.69		Sampling+LSDS++	73986.63	0.9394	8.98
	Sampling+KM	103730.83	**0.7669**	1.45		Sampling+KM	110881.26	**0.9466**	2.71
SUSY-5($k=10$)	TIKmeans	**62009744.02**	**0.7610**	52.37	SUSY-10($k=10$)	TIKmeans	63745990.97	0.9657	51.28
	IKmeans	64567215.89	0.6432	**31.02**		IKmeans	65767311.47	**0.9806**	**31.87**
	ECOD+LSDS++	62411224.21	0.6414	411.18		ECOD+LSDS++	64317119.99	0.9581	382.33
	ECOD+KM	69434356.01	0.6220	226.31		ECOD+KM	71406548.44	0.9656	227.24
	IFOREST+LSDS++	62390485.55	0.6069	762.71		IFOREST+LSDS++	65396216.44	0.9595	762.77
	IFOREST+KM	71284866.87	0.5612	645.37		IFOREST+KM	73307321.39	0.9563	655.94
	Sampling+LSDS++	62086548.45	0.5915	234.15		Sampling+LSDS++	**62861629.25**	0.9626	231.42
	Sampling+KM	70274277.80	0.5799	52.85		Sampling+KM	70336337.93	0.9620	52.45
Shuttle($k=10$)	TIKmeans	**65588.61**	0	0.80	KDDFULL($k=3$)	TIKmeans	32912080.32	0.6138	13.59
	IKmeans	71585.51	0.2353	**0.11**		IKmeans	**32672394.99**	0.6109	**6.35**
	ECOD+LSDS++	108848.04	0.1765	0.95		ECOD+LSDS++	39850069.04	0.6441	175.44
	ECOD+KM	117886.61	0.1765	0.38		ECOD+KM	45734240.61	0.5766	104.78
	IFOREST+LSDS++	93967.88	**0.2941**	3.22		IFOREST+LSDS++	40136898.91	0.6353	562.34
	IFOREST+KM	123861.95	0.1765	2.72		IFOREST+KM	43900960.86	**0.6451**	506.38
	Sampling+LSDS++	83140.46	0.1765	0.78		Sampling+LSDS++	32794029.79	0.6109	88.45
	Sampling+KM	117886.32	0.1765	0.22		Sampling+KM	57880009.63	0.6432	18.16
HIGGS-5($k=10$)	TIKmeans	318253744.73	0.7822	110.25	HIGGS-10($k=10$)	TIKmeans	321224528.63	**0.9803**	124.36
	IKmeans	320724229.60	**0.7959**	**59.72**		IKmeans	332663391.07	0.9801	**69.56**
	ECOD+LSDS++	**310575469.59**	0.7441	1180.81		ECOD+LSDS++	321175721.32	0.9693	1187.3
	ECOD+KM	342806052.39	0.6670	735.05		ECOD+KM	342496070.35	0.9761	757.11
	IFOREST+LSDS++	316878302.36	0.6195	1833.113		IFOREST+LSDS++	**317136850.88**	0.9704	1908.44
	IFOREST+KM	349755375.93	0.6730	1477.80		IFOREST+KM	342123536.51	0.9731	1514.77
	Sampling+LSDS++	322668394.78	0.6059	541.79		Sampling+LSDS++	317558363.82	0.9702	506.34
	Sampling+KM	341049644.85	0.6071	136.92		Sampling+KM	332381122.44	0.9717	137.19

C.5. Experiments on Synthetic Data with Large Number of Clusters and Significant Deviations

In this subsection, we give specific scenarios where our algorithm demonstrates particular effectiveness over RobustK-means++ algorithm. The synthetic datasets are generated by Gaussian distributions, where the k clustering centers

[2] https://archive.ics.uci.edu/dataset/280/higgs

$\{c_1, c_2, ..., c_k\}$ are randomly selected, with each cluster P_i formed as $P_i = \{x \in P_i : x \sim \mathcal{N}(c_i, \sigma)\}$ using Gaussian distribution, and σ is to control the deviation. Moreover, each cluster P_i roughly has $\frac{n}{k}$ data points.

We test different parameter settings with varying k and σ, where n, ξ, d are fixed to be $n = 10,000, \xi = 0.5, d = 20$ (ξ represents that the randomly generated outliers are in range $[-\xi, \xi]^d$). The results show that RobustKmeans++'s accuracy in identifying outliers is almost zero when $k = 50$ and $\sigma = 1$ while TIKmeans algorithm achieves an accuracy of 0.99 for outlier identification. Moreover, RobustKmeans++'s accuracy in identifying outliers is 0 when $k = 30$ and $\sigma = 2$ while our algorithms achieve the results of 0.94 and 0.48, respectively. These results indicate that RobustKmeans++ can hardly handle the scenarios when the number of clusters is large and the datasets have significant deviations.

Table 28. Comparisons results on synthetic datasets.

Datat Settings	Method	Cost	Recall	Time(s)
$k = 30, \sigma = 1.0$	TIKmeans	**9723.76**	**0.9901**	3.86
	IKmeans	10526.04	**0.9901**	**0.37**
	RobustKmeans++	10380.49	**0.9901**	0.67
$k = 40, \sigma = 1.0$	TIKmeans	**10387.45**	**0.9901**	5.19
	IKmeans	11194.94	**0.9901**	**0.59**
	RobustKmeans++	13991.13	**0.9901**	1.12
$k = 50, \sigma = 1.0$	TIKmeans	**10584.55**	**0.9901**	6.98
	IKmeans	12455.90	**0.9901**	**0.88**
	RobustKmeans++	13759.12	0	1.76
$k = 30, \sigma = 1.0$	TIKmeans	**9723.76**	**0.9901**	3.86
	IKmeans	10526.04	**0.9901**	**0.37**
	RobustKmeans++	10380.49	**0.9901**	0.67
$k = 30, \sigma = 1.5$	TIKmeans	**20565.79**	**0.9901**	3.64
	IKmeans	22246.86	**0.9901**	**0.39**
	RobustKmeans++	23899.62	**0.9901**	0.71
$k = 30, \sigma = 2.0$	TIKmeans	**36246.04**	**0.9405**	3.53
	IKmeans	45139.84	0.4812	**0.38**
	RobustKmeans++	45550.98	0	0.73