Coresets for Deletion-Robust k-Center Clustering

Ruien Li
reli@stu.ecnu.edu.cn
East China Normal University
Shanghai, China

Yanhao Wang
yhwang@dase.ecnu.edu.cn
East China Normal University
Shanghai, China

Michael Mathioudakis
michael.mathioudakis@helsinki.fi
University of Helsinki
Helsinki, Finland

Abstract

The k-center clustering problem is of fundamental importance for a broad range of machine learning and data science applications. In this paper, we study the deletion-robust version of the problem. Specifically, we aim to extract a small subset of a given data set, referred to as a *coreset*, that contains a provably good set of k centers even after an adversary deletes up to z arbitrarily chosen points from the data set. We propose a 4-approximation algorithm that provides a coreset of size $O(kz)$. To our knowledge, this is the first algorithm for deletion-robust k-center clustering with a theoretical guarantee. Moreover, we accompany our theoretical results with extensive experiments, demonstrating that our algorithm achieves significantly better robustness than non-trivial baselines against three heuristic gray-box and white-box adversarial deletion attacks.

CCS Concepts

• **Theory of computation → Approximation algorithms analysis**; • **Information systems → Clustering**.

Keywords

K-center Clustering, Deletion Robustness, Coreset

ACM Reference Format:
Ruien Li, Yanhao Wang, and Michael Mathioudakis. 2024. Coresets for Deletion-Robust k-Center Clustering. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management (CIKM '24), October 21–25, 2024, Boise, ID, USA.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3627673.3679890

1 Introduction

The k-center problem is a fundamental and well-studied formulation of metric clustering [13], with broad applications in machine learning and data science tasks [21, 22, 31]. In this problem, for a set X of n points in a metric space and a parameter $k \in \mathbb{Z}_+$, the goal is to find a subset $S \subseteq X$ of size k such that the maximum distance between any point in X and its nearest neighbor in S is minimized. This maximum distance is called the loss (or cost) of the set S of centers w.r.t. X. The k-center problem is known to be NP-hard [13] to approximate within a factor less than 2; that is, no polynomial algorithm can guarantee to find a set of centers whose loss is better than two times the optimal loss unless $P = NP$.

CIKM '24, October 21–25, 2024, Boise, ID, USA

ACM ISBN 979-8-4007-0436-9/24/10
https://doi.org/10.1145/3627673.3679890

In this paper, we investigate the deletion-robust version of k-center clustering. One primary motivation for considering data deletions is *privacy* [26]: At any time, users can exercise their "right to be forgotten" and thus prevent some data points from being used for analysis or picked as centers. Another motivation is *fault tolerance* [20]: When some centers become unavailable, how can we find their alternatives efficiently?

Following existing approaches [26], we formulate the robustness to deletion as a two-phase game against an adversary. In the first phase, the algorithm receives a data set X, the number k of centers to select, and a robustness parameter $z \in \mathbb{Z}_+$ to indicate the number of points to be deleted, and extracts a small subset $C \subseteq X$ (i.e., *coreset*). Moreover, during the first phase, an adversary chooses a set $D \subseteq X$ with $|D| = z$ to delete from X. In making this choice, the adversary is assumed to know the algorithm that produces the coreset and may have direct access to the resulting coreset instance C ("white-box setting") or not ("gray-box setting"). In the second phase, the adversary reveals the deletion set D; and as a response, the algorithm selects a set S of k centers from $C \setminus D$. In this problem formulation, it is desirable to design an algorithm to efficiently provide a concise coreset (sublinear to or independent of n) and a center selection with approximation guarantees.

Related Work. There is a comprehensive body of literature on the k-center clustering problem. In a seminal work, Gonzalez [13] proposed a greedy $O(nk)$-time algorithm, typically referred to as GMM, that produces the best possible 2-approximate solution. Recently, approximation algorithms were designed for k-center clustering in streaming [5, 8, 15], distributed [5, 24, 29], fully-dynamic [6, 14], and fairness-constrained [1, 8, 17, 18, 21] settings. Robustness to outliers was also considered in the design of k-center clustering algorithms [5, 10, 24, 25]. However, none of the above algorithms can be used for the deletion-robust setting with theoretical guarantees.

The closest formulation to ours is the fault-tolerant k-center problem [7, 12, 20], which selects k "robust" centers such that when $z < k$ centers fail, the remaining $k - z$ centers still serve as an approximately good solution. The fault-tolerant scheme differs from our deletion-robust scheme since it follows a more restricted setting. Specifically, in the fault-tolerant scheme, the algorithm can select only k points, knowing that z of them may be deleted, whereas, in our deletion-robust scheme, the coreset size can be greater than k. Moreover, for the fault-tolerant scheme, the deleted z points are limited to the selected centers, whereas, for our deletion-robust scheme, all points are candidates for deletion.

Deletion-robust submodular optimization [2, 4, 9, 11, 19, 26, 30] has also attracted much attention recently. We adopt the same scheme of deletion robustness as in these studies. However, since submodular optimization differs from k-center clustering, their algorithms cannot be used directly for our problem.

Our Contributions. We present DeletionRobustKC, a coreset-based approximation algorithm for the problem of deletion-robust k-center clustering. DeletionRobustKC consists of two subroutines, namely (1) DRCoreset, which extends GMM to compute a coreset C of size $O(kz)$ on the data set X, including enough center alternatives to anticipate any set of z deletions, and (2) DRSolution, which selects a set S of k centers closest to the original centers by GMM from the remaining points in C after the deletion of z points. Moreover, we show that DeletionRobustKC achieves an approximation factor of 4. To our knowledge, this is the first algorithm with a known theoretical guarantee for this problem. Finally, we provide an extensive empirical study on the effectiveness of DeletionRobustKC. Specifically, we devise several strategies for an adversary to decide the points to delete, including a gray-box strategy and two white-box strategies. The results show that DeletionRobustKC achieves significantly better robustness against all three deletion adversaries than non-trivial baselines on five public real-world and synthetic data sets.

2 Preliminaries

Let (X, d) be a finite metric space where X is a set of n points and $d : X \times X \mapsto \mathbb{R}_{\geq 0}$ is a distance function that measures the dissimilarity between any pair of points in X and satisfies the axioms of (i) *identity of indiscernibles*, (ii) *symmetry*, and (iii) *triangle inequality*. By extension, the distance function between a point x and a set S of points is defined as $d(x, S) = \min_{y \in S} d(x, y)$. Moreover, given a point $x \in X$ and a positive number $r \in \mathbb{R}_+$, a (closed) ball of radius r centered at point x is denoted by $B(x, r) = \{y \in X : d(x, y) \leq r\}$.

Based on the definitions above, the *k-center problem* asks for a set $S \subseteq X$ of k points (called "centers") to minimize the maximum distance between any point $x \in X$ and its closest center in S, i.e., $d(x, S)$. As such, the loss (or cost) of a set S w.r.t. X is defined as $c(S) = \max_{x \in X} d(x, S)$ and the optimal k-center is as follows:

$$S^* = \underset{S \subseteq X, |S| \leq k}{\arg\min} \; c(S). \tag{1}$$

An equivalent form of the k-center problem is to find k congruent balls of the smallest radius $r^* = c(S^*)$ such that all points in X are contained by at least one of these balls. We say that r^* is the optimal loss OPT of the problem. GMM [13] provides the best possible 2-approximate solution for Problem 1 in $O(nk)$ time. Specifically, it first picks an arbitrary point $x_1 \in X$ as the initial center set S_1. In the i-th iteration ($i = 2, \ldots, k$), it adds the point x_i farthest from S_{i-1}, i.e., $x_i = \arg\max_{x \in X} d(x, S_{i-1})$, to S_{i-1} as S_i. Finally, it returns the set $S = S_k$ after the k-th iteration for k-center clustering.

The deletion-robust k-center problem is posed in two phases. The first phase takes as input X, k, and a robustness parameter $z \in \mathbb{Z}_+$, and outputs a small subset of points — the so-called coreset C. The second phase takes as input a (possibly adversarial) deletion set D of size z and the output of the first phase and outputs the final k-center selection. Intuitively, the scheme asks for maintaining a coreset C from which, in the event of deletion, the set of k centers can be computed without access to the full data set. A high-quality coreset should be small in size (typically sublinear to or independent of n) while having a tight approximation factor. To measure how the coreset C can approximate the data set X, we define the notion of *(α, z)-robust coreset* as follows:

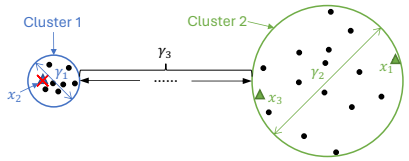
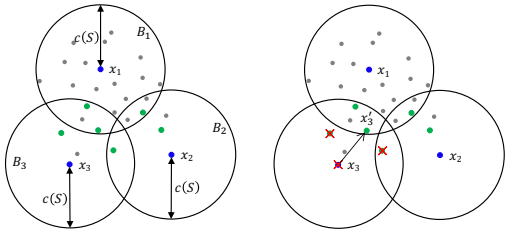


Figure 1: Illustration of the deletion vulnerability of GMM.



(a) Phase 1: DRCoreset (b) Phase 2: DRSolution

Figure 2: DeletionRobustKC for $k = 3$ and $z = 3$. (a) The coreset consisting of the points returned by GMM (Lines 1–3 of Algorithm 1) in blue and their alternative points (Lines 7–12 of Algorithm 1) in green. (b) The process for Algorithm 2 to find an alternative x_3' to the center x_3 when x_3 is included in the deletion set D, indicated by red cross-marks.

Definition 2.1. [(α, z)-Robust Coreset] A set $C \subseteq X$ is an (α, z)-robust coreset for k-center clustering on X if there is a subset $S' \subseteq C \setminus D$ with $|S'| \leq k$ such that $c(S') = \max_{x \in X \setminus D} d(x, S') \leq \alpha \cdot \mathtt{OPT}$ for an approximation factor $\alpha \geq 1$ and every $D \subseteq X$ with $|D| = z$.

Accordingly, a center set S' is α-approximate if $c(S') \leq \alpha \cdot \mathtt{OPT}$.

A straightforward adaptation of GMM to the deletion-robust setting is to run it for $k + z$ iterations to obtain a coreset $C = S_{k+z}$, ensuring that at least k points remain in the set after any z deletions. However, such an approach does not provide any approximation guarantee, as the remaining points may be arbitrarily bad for k-center clustering. We provide an example in Figure 1 to illustrate that GMM is not robust to adversarial deletion. Suppose that we have a point set X with two clusters, where the intercluster distance γ_3 is much greater than the intracluster distances γ_1, γ_2. For $k = 2$ and $z = 1$, GMM returns a solution $S_3 = (x_1, x_2, x_3)$, which is vulnerable to adversarial deletion: Once x_2 is deleted, $S' = (x_1, x_3)$ does not contain any point in the first cluster and becomes arbitrarily bad for k-center clustering.

3 Algorithm

This section introduces a coreset-based approximation algorithm, DeletionRobustKC, for deletion-robust k-center clustering, which implements the two-phase scheme described in Section 2. In the first phase, without knowledge of D, it extracts from the data set X a coreset C of size $O(kz)$ that is robust to any deletion set D of size z. In the second phase, after D is revealed, it removes D from C and, from the remaining points C', finds a final set of k centers. The overall process of DeletionRobustKC is illustrated in Figure 2.

Phase 1 (DRCoreset): Algorithm 1 contains DRCoreset, the procedure for building the coreset during the first phase. First, it invokes GMM to find a set S of size k (Lines 1–3). Then, it calculates the clustering cost $c(S)$ of the set S with respect to the data set X (Line 4). Based on the center set S and the cost $c(S)$, it draws k congruent balls $B_1, \ldots, B_k$, each of which is centered at a point

Algorithm 1: DRCoreset

Input : Data set X, size parameter k, and robustness parameter z
Output : Set $C \subseteq X$

1 Pick an arbitrary point $x_1 \in X$ and initialize $S_1 = \{x_1\}$;
2 **for** $i = 2$ **to** k **do**
3 $\quad x_i \leftarrow \arg\max_{x \in X} d(x, S_{i-1})$ and $S_i \leftarrow S_{i-1} \cup \{x_i\}$;
4 $S \leftarrow S_k$ and $c(S) \leftarrow \max_{x \in X} d(x, S)$;
5 **for** $i = 1$ **to** k **do**
6 $\quad$ Compute a ball B_i centered at point x_i of radius $c(S)$, i.e., $B_i = \{x \in X : d(x, x_i) \leq c(S)\}$;
7 **for** $i = 1$ **to** k **do**
8 $\quad$ **if** $|B_i| \leq z$ **then**
9 $\quad\quad C_i \leftarrow B_i$;
10 $\quad$ **else**
11 $\quad\quad C_i \leftarrow \{x_i\}$;
12 $\quad\quad$ Add z arbitrary points in B_i (excluding x_i) to C_i;
13 **return** $C \leftarrow \bigcup_{i=1}^{k} C_i$;

$x_i \in S$ (for $i \in [k]$) and has a radius equal to $c(S)$ (Lines 5–6). By the definition of $c(S)$, it is guaranteed that every point $x \in X$ must be contained in at least one ball (and possibly more than one). Next, it finds a set C_i of *alternatives* for each point $x_i \in S$ in response to its potential deletion. Intuitively, all points in B_i are considered possible alternatives to x_i – and, to anticipate the possibility that all z deletions occur within the same ball, z of them are selected for the coreset for each x_i. Specifically, for each $i \in [k]$, if B_i has at most z points, all of them are selected as C_i; otherwise, in addition to x_i, z points are arbitrarily selected from B_i to form C_i (Lines 7–12). Finally, the union of all C_i's is returned as the coreset C (Line 13).

Phase 2 (DRSolution): Algorithm 2 contains DRSolution, the procedure for computing the k centers during the second phase. First, it removes any deleted points D from the coreset C to obtain the updated coreset C' (Line 1). Then, it considers three cases for each $i \in [k]$: (1) if x_i is not deleted, x_i is still used as a center in the robust solution S' (Line 4); (2) if x_i is deleted but some of its alternatives in C_i remain after deletion, the alternative point closest to x_i in $C_i \cap C'$ is selected as the new center x'_i to replace x_i in the solution (Line 6); (3) if x_i is deleted along with all its alternatives in C_i, then an arbitrary point in C' is selected to replace x_i (Line 8). The above three cases guarantee that the returned solution contains at least one point from each non-empty $C_i \setminus D$ for $i \in [k]$.

Theoretical Analysis. Next, we analyze the approximation factor and complexity of DeletionRobustKC.

Theorem 3.1. *Algorithm 1 provides a $(4, z)$-robust coreset C of size $O(kz)$ in $O(nk)$ time.*

Proof Sketch. First, as analyzed in [13], we have $c(S) \leq 2 \cdot \mathtt{OPT}$. For each $x \in X$, there exists $x_i \in S$ with $d(x, x_i) \leq c(S)$, and thus $x \in B_i$. Then, according to the triangle inequality, for any $x, y \in B_i$, we have $d(x, y) \leq d(x, x_i) + d(y, x_i) \leq 2 \cdot c(S) \leq 4 \cdot \mathtt{OPT}$. For any deletion set D with $|D| \leq z$, each $C_i \subseteq B_i$ must satisfy one of the following two cases: (i) $C_i \setminus D \neq \emptyset$ and there is some $x \in C_i$ such that $d(x, y) \leq 4 \cdot \mathtt{OPT}$ for all $y \in B_i$; (ii) $C_i \setminus D = \emptyset$, and it must hold that $|C_i| \leq z$, $B_i = C_i$, and $B_i \setminus D = \emptyset$: in this case, no points remain from B_i and it does not contribute to the cost after deletion. Therefore, for a set T of centers, where each point is selected from a (nonempty) set $C_i \setminus D$, we have $|T| \leq k$ and $d(x, T) \leq 4 \cdot \mathtt{OPT}$ for every $x \in X \setminus D$, as $\bigcup_{i=1}^{k} B_i = X$, and thus $c(T) \leq 4 \cdot \mathtt{OPT}$. Finally, since $|C_i| \leq z + 1$ for each $i \in [k]$, we have $|C| \leq k(z + 1) = O(kz)$. Therefore, C is a $(4, z)$-robust coreset of size $O(kz)$.

The time complexity of GMM is $O(nk)$. Then, obtaining the k balls $B_1, \ldots, B_k$ also takes $O(nk)$ time. Finally, building all C_i's and C requires $O(kz)$ time. Since $n > k, z$, the time complexity of Algorithm 1 is $O(nk)$. □

Algorithm 2: DRSolution

Input : Coreset $C = \bigcup_{i=1}^{k} C_i$, deletion set $D \subseteq X$ with $|D| = z$, and size parameter k
Output : Set $S' \subseteq C \setminus D$ with $|S'| = k$

1 $C' \leftarrow C \setminus D$ and $S' \leftarrow \emptyset$;
2 **for** $i = 1$ **to** k **do**
3 $\quad$ **if** $x_i \in C'$ **then**
4 $\quad\quad S' \leftarrow S' \cup \{x_i\}$;
5 $\quad$ **else if** $C_i \cap C' \neq \emptyset$ **then**
6 $\quad\quad x'_i \leftarrow \arg\min_{x \in C_i \cap C'} d(x, x_i)$ and $S' \leftarrow S' \cup \{x'_i\}$;
7 **if** $|S'| < k$ **then**
8 $\quad$ Pick $k - |S'|$ points arbitrarily from $C' \setminus S'$ as a set S'';
9 $\quad S' \leftarrow S' \cup S''$;
10 **return** S';

Theorem 3.2. *Algorithm 2 returns a 4-approximate set S' of centers with $|S'| = k$ in $O(kz)$ time.*

Proof Sketch. According to the proof of Theorem 3.1, a set T is 4-approximate if $T \cap (C_i \setminus D) \neq \emptyset$ for every nonempty $C_i \setminus D$, which always holds by the set S' by Algorithm 2. Moreover, adding any new point to S' does not increase its clustering cost w.r.t. $X \setminus D$. Therefore, S' is a 4-approximate solution. Since finding x'_i for x_i takes $O(z)$ time, the time complexity of Algorithm 2 is $O(kz)$. □

4 Experiments

In this section, we present the setup and results of the experiments.

Implementation. All experiments were carried out on a computer with an Intel Core i5-10200H CPU @2.40GHz and 16.0GB RAM running Windows 10. The algorithms were implemented in Python 3. Our code and data are published at https://github.com/HonokaKousaka/DRkC.

Data Sets. In the experiments, we used four public real-world data sets and one synthetic data set, with the description, size, and dimensionality for each of them shown in Table 1. We randomly sampled 1,000 points from each data set for evaluation.

Baselines. We compare DeletionRobustKC with three baselines, which differ in how they build a coreset in the first phase: (i) GMM greedily picks $k + z$ points; (ii) FaultTolerantKC [20] provides

Table 1: Statistics of data sets in the experiments, where n is the number of data points and dim is the dimensionality.

Dataset	Description	n	dim
Adult [3]	Numeric attributes for income prediction	45,222	6
CelebA [23]	Features for celebrity images by VGG16	202,599	25,088
GloVe [28]	Global vectors for word representation	400,000	100
MovieLens [16]	User vectors derived from rating matrix	162,541	50
Synthetic	Gaussian blobs made by scikit-learn [27]	1,000,000	20

Table 2: Robustness of GMM, FaultTolerantKC, and DeletionRobustKC against different adversarial strategies (GB-GMM, WB-NN, and WB-Greedy) for the parameters $(k, z) = (10, 10)$, $(5, 20)$, and $(20, 5)$. The best result in each setting appears in bold.

Adversary		GB-GMM			WB-NN			WB-Greedy		
(k, z)		$(10, 10)$	$(5, 20)$	$(20, 5)$	$(10, 10)$	$(5, 20)$	$(20, 5)$	$(10, 10)$	$(5, 20)$	$(20, 5)$
Adult	GMM	1.3329	1.9476	1.4093	3.1066	2.6464	1.9664	2.9282	2.2984	1.8752
	FaultTolerantKC	1.3888	**1.0230**	1.2025	1.9513	2.3494	1.7595	2.5177	2.3546	1.8963
	DeletionRobustKC	**1.1950**	1.3925	**1.1002**	**1.0388**	**1.0467**	**1.0674**	**1.0823**	**1.0496**	**1.0856**
CelebA	GMM	1.2413	1.2844	1.2196	1.2862	1.2683	1.2724	1.2797	1.2904	1.2926
	FaultTolerantKC	1.2364	1.2820	1.2261	1.3263	1.3128	1.2596	1.3653	1.3412	1.3872
	DeletionRobustKC	**1.0355**	**1.0380**	**1.0298**	**1.0536**	**1.0486**	**1.0551**	**1.0541**	**1.0561**	**1.0606**
GloVe	GMM	1.1622	1.1881	1.0975	1.2309	1.2375	1.1456	1.2500	1.2718	1.2222
	FaultTolerantKC	1.1757	1.1309	1.1038	1.2295	1.1902	1.0749	1.2469	1.2901	1.1884
	DeletionRobustKC	**1.0595**	**1.0486**	**1.0508**	**1.0566**	**1.0628**	**1.0576**	**1.0705**	**1.0535**	**1.0469**
MovieLens	GMM	1.2136	1.2097	1.1875	1.2611	1.3466	1.1739	1.3632	1.4330	1.2805
	FaultTolerantKC	1.1981	1.2528	1.1916	1.2946	1.2502	1.1972	1.3734	1.4632	1.2869
	DeletionRobustKC	**1.0663**	**1.0567**	**1.0658**	**1.1216**	**1.1096**	**1.0635**	**1.1286**	**1.1569**	**1.0802**
Synthetic	GMM	1.0869	1.0762	1.1171	1.1921	1.1697	1.1725	1.2212	1.2275	1.1791
	FaultTolerantKC	1.0590	1.0307	1.0266	1.2233	1.2069	1.1082	1.1091	1.1915	1.1592
	DeletionRobustKC	**1.0431**	**1.0304**	**1.0176**	**1.0449**	**1.0409**	**1.0133**	**1.0457**	**1.0476**	**1.0285**

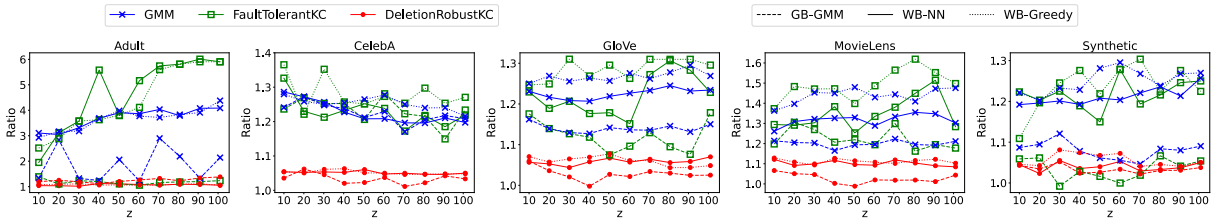


Figure 3: Robustness of each algorithm against different adversarial strategies when $k = 10$ and $z = 10, 20, \ldots, 100$.

$k+z$ centers robust to z failures; (iii) FullCoreset retains all points in the data set. In the second phase, with the deletion set D revealed, each baseline applies GMM on the remaining points of the coreset to obtain the k centers.

Adversaries. We devise the following three strategies for the adversary to decide the points to delete: (i) GB-GMM is a gray-box strategy without access to the coreset that aims to remove "good" center candidates from the data: to select the deletion set, it runs GMM for z iterations. (ii) WB-NN is a white-box strategy that knows the coreset before deletion. Since the loss of centers is bounded by the maximum loss, WB-NN aims to increase the loss by producing some points that any point in the coreset cannot represent. To achieve this, WB-NN randomly selects a point in the coreset and deletes the point along with its $(z-1)$-nearest neighbors. (iii) WB-Greedy is a white-box strategy that proceeds greedily in z iterations, each choosing a point that maximizes the increase in the loss if deleted.

Measures. To evaluate the robustness of each algorithm, we use the *ratio* of the loss of its solution w.r.t. a deletion set picked by an adversary over the loss achieved by FullCoreset w.r.t. the same deletion set. Note that FullCoreset is equivalent to GMM picking the k centers post-deletion, and its solution is 2-approximate for the post-deletion data set. Moreover, we ran each algorithm ten times and reported the average of the ratios. The smaller the (average) ratio is, the more robust the algorithm is to deletion.

Robustness Results. Table 2 presents the robustness results. We observe the following: DeletionRobustKC almost always shows significantly better robustness than GMM and FaultTolerantKC. The fact that FaultTolerantKC appears much less robust than DeletionRobustKC and exhibits close robustness to GMM is mainly attributed to the misalignment between the fault-tolerant and deletion-robust schemes, as explained in Section 1. As expected, the two white-box adversaries produce more effective deletion attacks (higher ratios) than the gray-box adversary in most cases.

Effect of Parameter z on Robustness. We further evaluate the effect of z on the robustness of each algorithm. The results for $k = 10$ and $z = 10, 20, \ldots, 100$ as shown in Figure 3. DeletionRobustKC consistently shows better robustness than the two baselines across all data sets and z's. In addition, its ratios are mostly steady for different z's and adversaries. In contrast, GMM and FaultTolerantKC often become less robust with increasing z, particularly when attacked by white-box adversaries. In terms of time efficiency, taking CelebA as an example, DeletionRobustKC runs in around 0.03 seconds when $k = 10$ and $z = 10$, while GMM and FaultTolerantKC run in about 0.07 and 20 seconds in the same setting.

5 Conclusion

In this paper, we formulated a novel deletion-robust k-center problem and proposed a simple yet effective coreset-based approximation algorithm, DeletionRobustKC, for this problem. We performed extensive theoretical and empirical studies to demonstrate the superior robustness of DeletionRobustKC against adversarial deletions compared to competing baselines. As a preliminary study, this paper leaves some open questions for future work. For example, as the coreset size $O(kz)$ and the approximation factor 4 of DeletionRobustKC are not tight compared to their lower bounds (resp., $O(k+z)$ and 2), how can they be further improved?

Acknowledgments. This work was supported by the National Natural Science Foundation of China (Grant No. 62202169) and the Academy of Finland (Decision no. 347747).

References

[1] Haris Angelidakis, Adam Kurpisz, Leon Sering, and Rico Zenklusen. 2022. Fair and Fast k-Center Clustering for Data Summarization. In *Proceedings of the 39th International Conference on Machine Learning (ICML '22)*. PMLR, 669–702.

[2] Dmitrii Avdiukhin, Slobodan Mitrović, Grigory Yaroslavtsev, and Samson Zhou. 2019. Adversarially Robust Submodular Maximization under Knapsack Constraints. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '19)*. Association for Computing Machinery, New York, NY, USA, 148–156.

[3] Barry Becker and Ronny Kohavi. 1996. Adult. UCI Machine Learning Repository. https://doi.org/10.24432/C5XW20

[4] Ilija Bogunovic, Slobodan Mitrovic, Jonathan Scarlett, and Volkan Cevher. 2017. Robust Submodular Maximization: A Non-Uniform Partitioning Approach. In *Proceedings of the 34th International Conference on Machine Learning (ICML '17)*. PMLR, 508–516.

[5] Matteo Ceccarello, Andrea Pietracaprina, and Geppino Pucci. 2019. Solving k-center Clustering (with Outliers) in MapReduce and Streaming, almost as Accurately as Sequentially. *Proc. VLDB Endow.* 12, 7 (2019), 766–778.

[6] T-H. Hubert Chan, Arnaud Guerquin, and Mauro Sozio. 2018. Fully Dynamic k-Center Clustering. In *Proceedings of the 2018 World Wide Web Conference (WWW '18)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 579–587.

[7] Shiri Chechik and David Peleg. 2015. The fault-tolerant capacitated K-center problem. *Theor. Comput. Sci.* 566 (2015), 12–25.

[8] Ashish Chiplunkar, Sagar Kale, and Sivaramakrishnan Natarajan Ramamoorthy. 2020. How to Solve Fair k-Center in Massive Data Models. In *Proceedings of the 37th International Conference on Machine Learning (ICML '20)*. PMLR, 1877–1886.

[9] Shuang Cui, Kai Han, and He Huang. 2024. Deletion-Robust Submodular Maximization with Knapsack Constraints. *Proceedings of the AAAI Conference on Artificial Intelligence* 38, 10 (2024), 11695–11703.

[10] Hu Ding, Haikuo Yu, and Zixiu Wang. 2019. Greedy Strategy Works for k-Center Clustering with Outliers and Coreset Construction. In *27th Annual European Symposium on Algorithms (ESA 2019)*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 40:1–40:16.

[11] Paul Duetting, Federico Fusco, Silvio Lattanzi, Ashkan Norouzi-Fard, and Morteza Zadimoghaddam. 2022. Deletion Robust Submodular Maximization over Matroids. In *Proceedings of the 39th International Conference on Machine Learning (ICML '22)*. PMLR, 5671–5693.

[12] Cristina G. Fernandes, Samuel P. de Paula, and Lehilton L. C. Pedrosa. 2018. Improved Approximation Algorithms for Capacitated Fault-Tolerant k-Center. *Algorithmica* 80, 3 (2018), 1041–1072.

[13] Teofilo F. Gonzalez. 1985. Clustering to Minimize the Maximum Intercluster Distance. *Theor. Comput. Sci.* 38 (1985), 293–306.

[14] Gramoz Goranci, Monika Henzinger, Dariusz Leniowski, Christian Schulz, and Alexander Svozil. 2021. Fully Dynamic k-Center Clustering in Low Dimensional Metrics. In *2021 Proceedings of the Workshop on Algorithm Engineering and Experiments (ALENEX)*. SIAM, 143–153.

[15] Sudipto Guha. 2009. Tight Results for Clustering and Summarizing Data Streams. In *Proceedings of the 12th International Conference on Database Theory (ICDT '09)*. Association for Computing Machinery, New York, NY, USA, 268–275.

[16] F. Maxwell Harper and Joseph A. Konstan. 2016. The MovieLens Datasets: History and Context. *ACM Trans. Interact. Intell. Syst.* 5, 4, Article 19 (2016), 19 pages.

[17] Sèdjro Salomon Hotegni, Sepideh Mahabadi, and Ali Vakilian. 2023. Approximation Algorithms for Fair Range Clustering. In *Proceedings of the 40th International Conference on Machine Learning (ICML '23)*. PMLR, 13270–13284.

[18] Matthew Jones, Huy L. Nguyen, and Thy D. Nguyen. 2020. Fair k-Centers via Maximum Matching. In *Proceedings of the 37th International Conference on Machine Learning (ICML '20)*. PMLR, 4940–4949.

[19] Ehsan Kazemi, Morteza Zadimoghaddam, and Amin Karbasi. 2018. Scalable Deletion-Robust Submodular Maximization: Data Summarization with Privacy and Fairness Constraints. In *Proceedings of the 35th International Conference on Machine Learning (ICML '18)*. PMLR, 2549–2558.

[20] Samir Khuller, Robert Pless, and Yoram J. Sussmann. 2000. Fault tolerant K-center problems. *Theor. Comput. Sci.* 242, 1-2 (2000), 237–245.

[21] Matthäus Kleindessner, Pranjal Awasthi, and Jamie Morgenstern. 2019. Fair k-Center Clustering for Data Summarization. In *Proceedings of the 36th International Conference on Machine Learning (ICML '19)*. PMLR, 3448–3457.

[22] Andrew Lim, Brian Rodrigues, Fan Wang, and Zhou Xu. 2005. k-Center problems with minimum coverage. *Theor. Comput. Sci.* 332, 1-3 (2005), 1–17.

[23] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. 2015. Deep Learning Face Attributes in the Wild. In *2015 IEEE International Conference on Computer Vision (ICCV)*. IEEE, 3730–3738.

[24] Gustavo Malkomes, Matt J. Kusner, Wenlin Chen, Kilian Q. Weinberger, and Benjamin Moseley. 2015. Fast Distributed K-Center Clustering with Outliers on Massive Data. *Advances in Neural Information Processing Systems* 28 (2015), 1063–1071.

[25] Richard Matthew McCutchen and Samir Khuller. 2008. Streaming Algorithms for k-Center Clustering with Outliers and with Anonymity. In *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques, 11th International Workshop, APPROX 2008, and 12th International Workshop, RANDOM 2008, Boston, MA, USA, August 25-27, 2008. Proceedings*. Springer, Berlin, Heidelberg, 165–178.

[26] Baharan Mirzasoleiman, Amin Karbasi, and Andreas Krause. 2017. Deletion-Robust Submodular Maximization: Data Summarization with "the Right to be Forgotten". In *Proceedings of the 34th International Conference on Machine Learning (ICML '17)*. PMLR, 2449–2458.

[27] Fabian Pedregosa, Gaël Varoquaux, et al. 2011. Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.* 12 (2011), 2825–2830.

[28] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 1532–1543.

[29] Xiaoliang Wu, Qilong Feng, Ziyun Huang, Jinhui Xu, and Jianxin Wang. 2024. New Algorithms for Distributed Fair k-Center Clustering: Almost Accurate as Sequential Algorithms. In *Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS '24)*. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, USA, 1938–1946.

[30] Guangyi Zhang, Nikolaj Tatti, and Aristides Gionis. 2022. Coresets remembered and items forgotten: submodular maximization with deletions. In *2022 IEEE International Conference on Data Mining (ICDM)*. IEEE, 676–685.

[31] Tinghao Zhang, Kwok-Yan Lam, and Jun Zhao. 2024. Device Scheduling and Assignment in Hierarchical Federated Learning for Internet of Things. *IEEE Internet Things J.* 11, 10 (2024), 18449–18462.