TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214 09/14 pp 1072–1084
DOI: 10.26599/TST.2023.9010013
Volume 28, Number 6, December 2023

Fair k-Center Problem with Outliers on Massive Data

Fan Yuan, Luhong Diao, Donglei Du, and Lei Liu*

Abstract: The clustering problem of big data in the era of artificial intelligence has been widely studied. Because of the huge amount of data, distributed algorithms are often used to deal with big data problems. The distributed computing model has an attractive feature: it can handle massive datasets that cannot be put into the main memory. On the other hand, since many decisions are made automatically by machines in today's society, algorithm fairness is also an important research area of machine learning. In this paper, we study two fair clustering problems: the centralized fair k-center problem with outliers and the distributed fair k-center problem with outliers. For these two problems, we have designed corresponding constant approximation ratio algorithms. The theoretical proof and analysis of the approximation ratio, and the running space of the algorithm are given.

Key words: machine learning; distributed algorithm; fairness constraints; outlier constraints; k-center problem

1 Introduction

When we need to process a large amount of data, we hope to find some representative points so that we can make an estimate of the overall data through them. The problem of finding representative points from a large number of data points is called a clustering problem. The clustering problem is a very widely used and important problem in machine learning. For example, when we are faced with a situation where the dataset to be processed is particularly large, we do not want to run our machine learning algorithm on the entire input dataset but instead on a small set that retains the statistical properties of the original dataset. Finding such a small set is very important. In addition, fairness is also an important field in machine learning. If the input data is biased in the process of machine learning, the machine learning algorithm trained on these data will show the same bias. Therefore, using the algorithm rationally to eliminate bias is also a very important research topic. Algorithms currently used for clustering problems have been shown to be biased on attributes, such as gender, race, and age[1]. This motivates scientists to propose fair clustering algorithms. Recently, the fair k-center problem has been shown to be useful in computationally fair data aggregation[2].

Suppose the input data is a set of real vectors with a gender property. Consider a scenario in which one wishes to construct a summary of k data points so that both genders are equally represented. The k points represent the distribution of the original dataset, and we call these k points the center point set. Suppose now we are given a center point set S. The Euclidean distance of a point from S is the price we pay for not including it in S, and the cost of set S is the highest cost of all other points that are not in S. We want to compute a center point set S with minimum cost that is also fair; e.g., a set S contains $k/2$ women and $k/2$ men. In one sentence, we want to compute a fair center point set S wherein each gender has the same number of centers, and the data point that is farthest from this set is not too far. The fair k-center problem models this task.

Now we assume that the input to the problem is a set of vectors with attributes, and we need to find k data

• Fan Yuan, Luhong Diao, and Lei Liu are with the Beijing Institute for Scientific and Engineering Computing, Faculty of Science, Beijing University of Technology, Beijing 100124, China. E-mail: yuanfan@amss.ac.cn; diaoluhong@bjut.edu.cn; liuliu_leilei@bjut.edu.cn.
• Donglei Du is with the Faculty of Management, University of New Brunswick, Fredericton, E3B 5A3, Canada. E-mail: ddu@unb.ca.
∗ To whom correspondence should be addressed.
Manuscript received: 2022-10-04; revised: 2022-12-02; accepted: 2023-03-13

points to represent the entire set of data points. We hope that the attribute distribution and the number of these k data points meet our prerequisites. We call these k points the center point set and use S to represent them. These k points will divide the original data set into k groups. For each group, we take the maximum distance from the point in the group to the center point as the cost of this grouping. Our goal is to find a center point set that has the lowest cost and remains fair. That is, we want to find a fair center point set and find the data point farthest from this center point set such that the distance between the data point and the center point set is not too great. In the fair k-center problem, we use Q to represent the input data set of the problem. The number of data points in Q is n, the data points have m different attributes, the size of the center point set is k, and k_j is the capacity constraint of each attribute. For fairness, we have $k = \sum_{j=1}^{m} k_j$. Our goal is to find a center point set S, and the center points in S need to satisfy the condition that the number of center points that belong to attribute j is k_j. In this way, we maintain a balance of the central points, so we consider it to be fair. Furthermore, we want $\max_{i\in Q} \delta(i, S)$, where δ represents the distance function, to be minimized. Here each data point belongs to only one attribute. Under this definition of fairness, some relevant research on the fair k-center problem has been done[3]. Here we give the definition of the approximation ratio of the algorithm.

Definition 1 For any optimization problem $\mathcal{P}$ and any instance $\mathcal{I}$ of this problem, we use $\text{OPT}(\mathcal{I})$ to represent the value of the optimal solution of instance $\mathcal{I}$ and $\mathcal{A}(\mathcal{I})$ to represent the objective function value when we use algorithm $\mathcal{A}$ for instance $\mathcal{I}$. For the maximization problem, the approximate ratio of algorithm $\mathcal{A}$ is defined as

$$r(\mathcal{A}) := \inf_{\mathcal{I}} \frac{\mathcal{A}(\mathcal{I})}{\text{OPT}(\mathcal{I})}.$$

For the minimization problem, the approximate ratio of algorithm $\mathcal{A}$ is defined as

$$r(\mathcal{A}) := \sup_{\mathcal{I}} \frac{\mathcal{A}(\mathcal{I})}{\text{OPT}(\mathcal{I})}.$$

In the work of Ref. [3], for the fair k-center problem, the authors proposed an algorithm with an approximate ratio of 3, and for the distributed fair k-center problem, they proposed an algorithm with an approximate ratio of 17.

For the general k-center problem, a simple greedy algorithm can achieve an approximate ratio of 2[4, 5], and attaining an approximation ratio better than 2 is NP-hard[6]. Therefore, the general fair k-center problem is also NP-hard. The best result for the general fair k-center problem is a 3-approximation algorithm whose running time is $O(n^2 \cdot \log n)$[7]. Later, Ref. [2] gave a linear time algorithm for this problem with an approximate ratio of $O(2^c)$, where c is a constant.

The k-center clustering works well for low-noise data, but it is not effective for clustering datasets with noisy data. This is because the objective function of the k-center problem is the maximum distance from the data point to the nearest center point, and a point that is very far away can produce particularly poor results for the k-center problem. Therefore, it is necessary to study the k-center problem with noise, such as in Refs. [8–10]. Reference [11] studied the k-center problem with outliers. In this problem, the objective function is the same as the general k-center problem, except that the problem allows removing at most z data points from the dataset without clustering them. These points are called outliers, which are not taken into account in the objective function. The current best result for this problem is the 3-approximation algorithm in Ref. [11]. There are also many studies that focus on the distributed k-center problem with outliers[12–14]. The current best results for this problem are in Ref. [13], which gave a 4-round information exchange bicriteria algorithm with an approximate ratio of $(24(1+\epsilon), 1+\epsilon)$, and the communication cost of the algorithm is independent of the number of outliers.

Once the dataset becomes particularly large, the approximation algorithm for general clustering problems will become inefficient. In this case, we need distributed clustering algorithms and streaming algorithms to deal with the corresponding problems. Because of the ever-increasing number of datasets and the emergence of many modern parallel computing frameworks, the problem of clustering in distributed environments has attracted much attention in recent years. Several works studied streaming algorithms[15–18], others focused on distributed computing[19–22].

In addition to the methods above, there are sliding window methods and core methods for addressing big data clustering problems. References [23–25] examined the sliding window method , and Refs. [26, 27] have examined the core method.

In a distributed setting, the data is divided into T parts and stored on T different machines, and the machines can exchange information with each other to obtain better clustering results. The earliest study of the k-

center problem in the distributed computing setting was presented in Ref. [21], which gave a MapReduce algorithm that requires constant rounds of information exchange, the approximation ratio is constant, and the algorithm is a sampling-based MapReduce algorithm, which is simple and easy to implement, and can be used to solve various clustering problems. In the conference version of this article[28], we considered the distributed fair k-center clustering problem with outliers.

For the aforementioned reasons, in this work, we study the centralized fair k-center problem with outliers and the distributed fair k-center problem with outliers. Both models maintain fairness and are suitable for handling big data and noisy data. For the centralized fair k-center problem with outliers, we give an algorithm with an approximate ratio of 4, and for the distributed fair k-center problem with outliers, we give an algorithm with an approximate ratio of 18. In our article, we assume that the data points are stored on T data storage machines and that we can run our algorithm on these T machines. All approximation ratios of the algorithm, running time, space usage, and the amount of information exchange are proved.

2 Related Work

In addition to fairness constraints, there are also many papers that focus on the k-center problem with matroid constraints. For this problem, Ref. [7] gave the first polynomial time 3-approximation algorithm, Ref. [29] gave a one-pass algorithm with an approximation ratio of $(17+\epsilon)$ and a two-pass algorithm with an approximation ratio of 3. The space complexity of both algorithms is $O(k^2)$.

There are other ways to define fairness, such as using diversity-determined standards to measure fairness[30]. There are also many different types of fair clustering. A series of works[31–36] has studied the fair clustering problem with capacity constraints: the number of data points in each cluster has an upper or lower bound limit, or the number of data points within each cluster must be similar. There are also fair clustering problems under other definitions, such as those in Refs. [37, 38].

The k-median problem with fairness constraints was first proposed and studied by Ref. [39]. Later, Ref. [40] studied the k-median problem with matroid constraints. The works in Refs. [7, 29] are also applicable to this problem. For more work on fair clustering, please refer to Refs. [2, 30] for more detailed information.

The remainder of the paper is organized as follows: Section 3 gives preliminary definitions of the paper, Section 4 presents the algorithms and all the analysis, and Section 5 concludes this paper.

3 Preliminary

First, we introduce some symbols and marks that are used later. In our work, we set the input of the fair k-center problem to contain a data point set Q, and it has n points, a distance function $\delta : Q \times Q \rightarrow \mathbf{R}_{\geqslant 0}$, and the number of desired center points k. All points in the dataset have their own unique attribute, and we use $\{1, 2, \ldots, m\}$ to represent m attributes. We use $I : Q \rightarrow \{1, 2, \ldots, m\}$ to denote the attributes assignment function. In addition, for each attribute j, we have a fairness limit k_j that corresponds to the attribute. For all attributes, we have $k = \sum_{j=1}^{m} k_j$. This means that in the final selected center point set, the number of center points of each attribute should satisfy its corresponding fairness constraints. For the distance function δ, for any two points $q_1, q_2 \in Q$, we require that $\delta(q_1, q_1) = 0$ and $\delta(q_1, q_2) = \delta(q_2, q_1)$. We also require that the distance function δ obeys the triangle inequality: for each triple $q_1, q_2, q_3 \in Q$, we have $\delta(q_1, q_2) + \delta(q_2, q_3) \geqslant \delta(q_1, q_3)$. For any point $q_1 \in Q$ and any set $S \subseteq Q$, we use $\delta(q_1, S) = \min_{j \in S} \delta(q_1, j)$ to denote the distance between a point and a set. At the same time, we set the symbol $B(v, d, Q) = \{u : u \in Q, \delta(u, v) \leqslant d\}$ to denote a sphere of radius d centered at point v in dataset Q. In this problem, our goal is to find a set of center points that contains k points and minimize $\max_{i \in Q} \delta(i, S)$, and the attribute distribution of the center point set satisfies the fairness constraint given by our problem.

Here we need to emphasize that the number of center points selected in the algorithm process may be less than k, but it does not matter. We only need to make up the center points to k without violating the attribute constraints. The increase of the center point in all clustering problems will only lead to a decrease in the cost. In what follows, the center point selected by our algorithm just need not violate the attribute constraints. It does not matter if the number of center points is less than k.

For the fair k-center problem with outliers, the problem setting is the same as before. The only difference is that we need to find a set of outliers Z. In this problem we want to find a center point set $S \subseteq Q$, with S containing k points, and an outliers point set Z, with Z containing z points. Moreover, we hope that $\max_{i \in Q \setminus Z} \delta(i, S)$ is minimized. At the same time, we require the center point set to meet the attribute constraint, that is, $k = \sum_{j=1}^{m} k_j$. In the problem with outliers, we do not need to select outliers immediately. Instead, we can wait for the final k-center points to be selected and select z points farthest from the current center point in the data point set as outliers.

4 Problem and Algorithm

4.1 Centralized fair *k*-center problem with outliers

In order to solve the fair k-center problem with outliers, we first start from the k-center problem with outliers, and then solve the fairness constraint on the basis of this problem. Thus, we first introduce a common algorithm for solving the k-center problem with outliers. We know that problems with outliers are generally harder than problems without outliers. This is because, in this setting, we need to determine which points will be deleted, which has a great impact on the selection of the center point set. Therefore, the solution of the k-center problem with outliers may be very different from the solution of the k-center problem without outliers. In general, the correct algorithm strategy is to choose a point surrounded by many data points as the center point. Since such points are surrounded by many points, these points are unlikely to be outlier points. This idea was proposed in the algorithm of Ref. [11], which is famous for solving k-center problem with outliers.

Theorem 1[11] For the k-center problem with outliers, Algorithm 1 is an algorithm with an approximate ratio of 3, and the algorithm finds no more than z outliers.

Algorithm 1 was proposed by Charikar et al.[11]

Algorithm 1 Outliers ($Q, k, d, 3d$)

```
Input: dataset $Q$, integer $k$, and radius $d$
Step 1: Initialize $Q' = Q$ and $P = \varnothing$
Step 2: while $| P | < k$ do
Step 3:     For all points $v \in Q'$ calculate the corresponding ball $B(v, d, Q')$;
Step 4:     Find point $v_{\max} = \arg\max_{v \in Q'} | B(v, d, Q') |$;
Step 5:     $P \leftarrow P \cup \{v_{\max}\}$;
Step 6:     Calculate the corresponding ball $B(v_{\max}, 3d, Q')$;
Step 7:     $Q' \leftarrow Q' \setminus B(v_{\max}, 3d, Q')$;
Step 8: end while
Output: $P$
```

The input of the algorithm is the data point set Q, the number of center points k, and a radius d. The radius d is actually an estimate of the value of the optimal solution to our problem. We use OPT to represent the value of the optimal solution to the k-center problem with outliers. When $d = OPT$, the performance of Algorithm 1 can achieve the best. Since the value of the solution to the k-center problem is the distance between two points, the value of the optimal solution can be obtained by searching the possible values of the minimum and maximum distances between the input points. We can find a good estimate of the value of the optimal solution in polynomial time using a binary search.

The main idea of our algorithm to solve the fair k-center problem with outliers comes from Algorithm 1. Next, we give a detailed introduction to Algorithm 1, so that we can have a deeper understanding of the algorithm. For all data points $v \in Q$, the ball $B(v, d, Q)$ contains all data points in Q whose distance to point v is less than or equal to d. The algorithm first calculates the corresponding ball $B(v, d, Q')$ for all data points, and then selects the center point $v_{\max}$ of the ball that covers the most data points to add to the current solution. The idea of Algorithm 1 is consistent with what we mentioned earlier: finding the point surrounded by many data points as the center point. Then, Algorithm 1 deletes all points whose distance from the point $v_{\max}$ is less than or equal to $3d$ from the current data point set Q', to ensure that the points selected by Algorithm 1 are representative rather than selecting all the center points together. Finally, repeat the operation above k times. In this way, the final set of center points P is selected. In addition, we know that when d is equal to OPT and the final set of center points P is selected, no more than z points will remain in Q', which are the outliers finally selected by Algorithm 1. The steps of Algorithm 1 are simple, but the inner thoughts of Algorithm 1 are profound. In addition, we know that the running time of Algorithm 1 is $O(k|Q|)$.

After knowing how to deal with outliers, we next introduce how to deal with fairness constraints. We use the algorithms FindNeighbor ( ) and FindMatching ( ) to solve the fairness constraints of the problem. These two algorithms were proposed by Ref. [3] to deal with the fairness constraints of the problem. The input of the FindNeighbor ( ) (Algorithm 2) contains a data point set Q, a number of center points k, a distance function δ, a radius d, an attribute assignment function I, and a

Algorithm 2 FindNeighbor (Q, I, P, d)

```
Input: Set $Q$, $P$, number of center points $k$, attribute assignment function $I$, radius $d$, and distance function $\delta$.
Step 1: for all points $p \in P$
Step 2:     $N_p \leftarrow p$;
Step 3: end for.
Step 4: for all points $p \in P$
Step 5:     for all points $q \in Q$
Step 6:         if there is no point in $N_p$ with the same attribute as point $q$, we have $\delta(p,q) \leqslant d$;
Step 7:             $N_p \leftarrow N_p \cup \{q\}$;
Step 8:         end if
Step 9:     end for
Step 10: end for
Output: $\{N_p : p \in P\}$
```

subset $P \subseteq Q$. First, Algorithm 2 initializes $N_p = p$ for all points $p \in P$, and then we make N_p contain one point from each attribute. The premise is that such data points exist in a ball with p as the center and d as the radius. In the end, we get a neighborhood of each point $p \in P$, which contains as many data points of various attributes as possible. Obviously, we can know that the running time of FindNeighbor( ) is $O(|P||Q|)$. In short, if P is an infeasible center point set (points lacking some attributes), then FindNeighbor( ) finds as many data points of various attributes as possible in the neighborhood of each $p \in P$, so as to construct a set N_p of points that contain as many attributes as possible. In this way, from N_p, we can find a solution that satisfies the fairness constraint.

The main purpose of the algorithm FindMatching( ) (Algorithm 3) is to find a feasible solution to our problem from a set of disjoint sets. The algorithm FindMatching( ) takes the pairwise disjoint sets $N = (N_1, N_2, \ldots, N_J)$, the attribute assignment function I, and the attribute capacity constraint $\overline{k} = (k_1, k_2, \ldots, k_m)$ of the problem as inputs. It returns a set of feasible solutions S that intersect as many N_p as possible. This is obtained primarily by constructing a suitable bipartite graph and then finding the maximum cardinality matching in the bipartite graph. The running time of the algorithm is $O(J^2 \max_i |N_i|)$, where J is the number of the input sets,

Algorithm FindMatching( ) constructs the bipartite graph we need according to the following steps. We set J vertices on the left side of the bipartite graph, which means that each vertex on the left side of the graph corresponds to a set N_i. There are $|A| = |\bigcup_{j=1}^{m} A_j|$ points on the right vertex set of the bipartite graph, where

Algorithm 3 FindMatching ($N, I, \overline{k}$)

```
Input: Set $N = (N_1, N_2, \ldots, N_J)$, attribute assignment function $I$, and vector $\overline{k} = (k_1, k_2, \ldots, k_m)$ of fairness constraints
Step 1: Construct bipartite graph $G = (N, A, E)$ as follows:
Step 2: Set $J$ vertices on the left side of the bipartite graph;
Step 3: Set $|A| = |\cup_{j=1}^{m} A_j|$ vertices on the right side of the bipartite graph, $A_j$ contains $k_j$ vertices;
Step 4: for all $N_i$ and all attribute $j$
Step 5:     if $\exists p \in N_i$ such that $I(p) = j$;
Step 6:         Connect $N_i$ to all vertices in $A_j$;
Step 7:     end if
Step 8: end for
Step 9: Find the maximum cardinality matching $H$ of $G$;
Step 10: $S \leftarrow \varnothing$;
Step 11: for all edges $(N_i, a)$ of $H$
Step 12:     Let $s$ be a point in $N_i$ from attribute $j$, where $a \in A_j$;
Step 13:     $S \leftarrow S \cup \{s\}$;
Step 14: end for
Output: $S$
```

A_j contains k_j vertices of each attribute j. For the edges of the graph, if N_i contains a point from attribute j, then we connect an edge among N_i and all points in A_j, that is, to k_j vertices.

Next, we find a feasible solution set S by finding a maximum cardinality matching H of the bipartite graph. First, for each edge $e = (N_i, a)$ in the matching H, here a is a vertex in A_j. We add the points that belong to attribute j from N_i to S. At this time, we know the number of points in set A_j is k_j. Also, we know that H is a maximum matching, so we know that S contains at most k_j points from attribute j. Therefore, S satisfies the attribute constraint of the problem. Then, because $|S| = |H|$, we know that the solution S that corresponds to the maximum cardinality matching in the bipartite graph intersects as many N_i as possible. Finally, as mentioned earlier, $|S| = |H|$ may be less than k, but which has no impact on the final result of the problem.

Lemma 1 The distance from the point in the solution output by Algorithm 2 to the point in P is less than d.

When we have the aforemetioned algorithms, we next formally introduce the main algorithm of this section. Algorithm 4 is a combination of the three algorithms described above.

First, we use the algorithm Outliers $(Q, k, d, 3d)$ to find a candidate center points set, and we denote this set by P. Second, for each point $p \in P$, we use the algorithm FindNeighbor (Q, I, P, d) to find a neighborhood N_p with radius d around it. In these

Algorithm 4 Fair k-center with outliers

```
Input: Set $Q$, attribute assignment function $I$, vector $\overline{k} = (k_1, k_2, \ldots, k_m)$ of fairness constraints, radius $d$, and number of center points $k$
Step 1: $P \leftarrow$ Outliers $(Q, k, d, 3d)$;
Step 2: $\{N(p) : p \in P\} \leftarrow$ FindNeighbor$(Q, I, P, d)$;
Step 3: $S \leftarrow$ FindMatching$(\{N(p) : p \in P\}, I, \overline{k})$;
Output: $S$
```

neighborhoods, there should be as many data points containing various attributes as possible. Finally, we use the algorithm FindMatching($\{N(p) : p \in P\}, I, \overline{k}$) to obtain a final solution S to our problem, and through the algorithm FindMatching($\{N(p) : p \in P\}, I, \overline{k}$), we know that S is a feasible solution.

From the information above, we can see that the first step of Algorithm 4 is to find a suitable candidate center points set, and the second and third steps of Algorithm 4 are to satisfy the fairness constraint. In order to ensure that Algorithm 4 can maintain the fairness constraint, our outliers can be deleted at the end of the algorithm. That is, we only select k candidate center points in the first step, and select outliers after k final center points are selected in the third step. Those z data points furthest from the final center points are selected as outliers. In this way, Algorithm 4 satisfies both the fairness constraint and the outlier constraint.

In the next theorem, we prove the approximation ratio of Algorithm 4.

Theorem 2 For the fair k-center problem with outliers, Algorithm 4 is an algorithm with an approximate ratio of 4.

Proof First, we prove the approximation ratio of Algorithm 4. Suppose S is the output set of Algorithm 4.

From Algorithm 3 , we know that the points in S are all selected from $\{N(q) : q \in P\}$. And from Lemma 1, we know that for every point $s \in S$, there must exist a point $p \in P$ whose distance is less than d. This means we have $\delta(s, p) \leqslant d$.

From Algorithm 1 and Theorem 1, we know that the solution of Algorithm 1 is a 3-approximation solution, which means that for the solution P output by Algorithm 1, the distances of all data points to their center point do not exceed $3OPT$. Specifically, for any point v in Q, there must exist a point in $p \in P$ whose distance from v is less than $3d$. This means we have that $\delta(v, p) \leqslant 3d$. Adding up the two distances, we know that the distances from any point $v \in Q$ to the nearest center point in S are less than $4d$.

In the proof above, we also need to prove that S has interaction with all $N(p)$. Suppose O^* is the optimal solution of the fair k-center problem with outliers. For each $p \in P$, let $c_p \in O^*$ be the center point of p in the optimal solution. Thus we have $\delta(p, c_p) \leqslant d$. At the same time, because the distances between any two points in P are greater than $3d$, all c_q are different. According to the previous construction method, we have that $N(p)$ contains one point from every attribute, and the distanced between these points and p are less than or equal to d. Therefore, $N(p)$ must contain a point b_p that has the same attribute as c_p, and we have $\delta(p, b_p) \leqslant d$. For some $p \in P$, it is possible that $c_p = b_p$. Next, we consider set $B = \{b_p : p \in P\}$. For each point p, set B intersects all $N(p)$. B contains as many attribute points as $\{c_p : p \in P\} \subseteq O^*$, so B is also a feasible solution and satisfies the fairness constraint. Therefore, we know that there is a feasible solution B, which intersects $N(p)$ of each $p \in P$. We also know that the output S of Algorithm 4 is a feasible solution, intersecting as many $N(p)$ as possible. Therefore, S can also intersect all the $N(p)$. ■

Next, we give the time complexity analysis and memory analysis of Algorithm 4. Here we can assume that the distance between two points is calculated in $O(1)$ time.

Lemma 2 The working space of Algorithm 4 is $O(km)$, and the running time of Algorithm 4 is $O(mk^2 + kn)$.

Proof The memory space required by Algorithm 4 is used to store the set of center points and their corresponding $N(p)$. From Algorithm 1, the number of points in P is k. Since in Algorithm 2, each $N(p)$ contains at most one point from any attribute, it has at most m points. Therefore, we have that the storage space of Algorithm 4 is $O(km)$.

Carefully observing the inputs and outputs of Algorithms 1–3 in Algorithm 4, we know that Algorithms 1 and 2 both take time $O(|k||Q|) = O(kn)$, because in Algorithm 4, the size of the input P of Algorithm 2 is k. Algorithm 3 takes time $O(J^2 \cdot \max_i |N_i|) = O(|P|^2 \cdot \max_{p \in P} |N_p|) = O(k^2 m)$ because the input set number of Algorithm 3 is k, and each $N(p)$ contains at most m points. ■

4.2 Distributed fair k-center problem with outliers

Algorithm 4 we introduced in the previous section for dealing with fair k-centers may not be very effective when the dataset is particularly large, so in this section, we consider a distributed algorithm for the k-center problem. First, we introduce our problem in a distributed setting. In the distributed fair k-center problem with outliers, because of the large amount of data, all data points Q are evenly stored on T data storage machines. Our target is the same as the goal of the previous question. In this problem, we want to find a center point set $S \subseteq Q$, with S containing k points, and an outliers point set Z, with Z containing z points. Moreover, we hope that $\max_{i \in Q \setminus Z} \delta(i, S)$ is minimized. At the same time, we require that the final selected center point set S still satisfies the fairness constraint.

Here, for the convenience of proof and reading, we designate those machines that averagely store the data points Q as the distributed machines, and we call the machine that collects the processing results of all distributed machines, runs the algorithm, and outputs the final result as the central machine. Additionally we denote the set of data points stored on each distributed machine i by Q_i.

Algorithm 5 is a classic greedy algorithm for k-center problems, which will appear as a subalgorithm of Algorithm 6. We first introduce the overall idea and architecture of Algorithm 6, and then provide a detailed introduction to Algorithm 5. It contains two subalgorithms, FindNeighbor$(Q_i, I, P_i, 2d_i)$ and Greedy$(Q, k+z+1)$. The function of the algorithm Greedy$(Q, k+z+1)$ is mainly to find a suitable set of candidate center points, and the function of the algorithm FindNeighbor$(Q_i, I, P_i, 2d_i)$ is mainly to find a suitable set of feasible points that satisfy the fairness constraint. When Algorithm 6 on the distributed machine is finished, we send the result to the central machine.

The algorithm we run on the central machine is Algorithm 7. It contains three subalgorithms: the algorithm DistributeOutliers $(P', k, 5d, 11d)$, the algorithm FindNeighbor $(L', I, P, 5d)$, and the algorithm FindMatching $(\{N(p) : p \in P\}, I, \overline{k})$. Here DistributeOutliers $(P', k, 5d, 11d)$ (Algorithm 8) is used to select a suitable set of candidate center points, while the latter two algorithms, the algorithm FindNeighbor $(L', I, P, 5d)$ and the algorithm FindMatching $(\{N(p) : p \in P\}, I, \overline{k})$, are used to find the center point set that satisfies the fairness constraint.

Theorem 3[41] For the k-center problem, Algorithm 5 is an algorithm with an approximate ratio of 2.

Here we first introduce a classical algorithm proposed by Dyer et al.[41] to deal with the k-center problem, which is a simple greedy algorithm, the point farthest

Algorithm 5 Greedy (Q, k)

```
Input: Point set $Q$, number of center points $k$, and distance function $\delta$
Step 1: Initialize $S = \varnothing$;
Step 2: Pick an arbitrary point $i \in Q$;
Step 3: $S \leftarrow i$;
Step 4: while $| S | < k$
Step 5:     Choose $j$ from $Q$ with the biggest $\delta(j, S)$;
Step 6:     $S \leftarrow S \cup j$, $Q - \{j\}$;
Step 7: end while
Output: $S$
```

Algorithm 6 Distributed fair k-center with ourliers in each distributed machine i

```
Input: Point set $Q_i (1 \leqslant i \leqslant T)$, number of center points $k$, and distance function $\delta$
Step 1: Run algorithm Greedy$(Q, k+z+1)$ on each distributed machine $i$ and output a set $P_i \leftarrow \{p_1, p_2, \ldots, p_{k+z}\}$, $P_i$ has $k+z$ data points;
Step 2: $d_i \leftarrow \min_{j' : 1 \leqslant j' \leqslant k+z} \delta(p_j, p_{k+z+1})/2$;
Step 3: $\{L(p) : p \in P_i\} \leftarrow$ FindNeighbor$(Q_i, I, P_i, 2d_i)$;
Step 4: $L_i \leftarrow \cup_{p \in P_i} L(p)$;
Step 5: for all points $p \in P_i$
Step 6:     Machine $i$ record $w_p = | \{v : v \in Q_i, \delta(p, v) = \delta(P_i, v)\} | + 1$;
Step 7: end for
Output: $(P_i, L_i, w_p)$
```

Algorithm 7 Distributed fair k-center with ourliers in central machine i

```
Input: Set $P' = \cup_{i=1}^{T} P_i$, $L' = \cup_{i=1}^{T} L_i$, number of center points $k$, distance function $\delta$, and parameter $w_p$
Step 1: $P \leftarrow$ Distribute Outliers $(P', k, 5d, 11d)$;
Step 2: $\{N(p) : p \in P\} \leftarrow$ FindNeighbor$(L', I, P, 5d)$;
Step 3: $S \leftarrow$ FindMatching$(\{N(p) : p \in P\}, I, \overline{k})$;
Output: $S$
```

Algorithm 8 DistributeOutliers $(Q, k, 5d, 11d)$

```
Input: Set $Q$, number of center points $k$, radius $d$, and parameter $w_p$
Step 1: Initialize $Q' = Q$ and $P = \varnothing$;
Step 2: while $| P | < k$
Step 3:     for all points $v$ in $Q$
Step 4:         Calculate the corresponding ball $B(v, 5d, Q')$;
Step 5:     end for
Step 6:     Find a point $v_{max}$ in $Q$, the definition of point $v_{\max}$ is as follows: $v_{\max} = \text{argmax}_{v \in Q} \sum_{v' \in B(v, 5d, Q')} w_{v'}$;
Step 7:     $P \leftarrow P \bigcup \{v_{\max}\}$;
Step 8:     Calculate the corresponding ball $B\,(v_{\max}, 11d, Q')$;
Step 9:     $Q' \leftarrow Q' \backslash B(v_{\max}, 11d, Q')$;
Step 10: end while
Output:: $P$
```

from the current center point set is selected and added to the current solution, and this step is repeated k times to find the final k center points.

We run Algorithm 6 on each distributed machine. Algorithm 6 has three main steps. In the first step, each distributed machine i runs Algorithm 5 to find $(k+z+1)$ points. Then we use the first $(k + z)$ points to form a new set P_i. At this time, the last point p_{k+z+1} is the farthest point from the new dataset P_i, and its distance from P_i is $2d_i$. Therefore, the distance between each point in Q_i and the set P_i is less than $2d_i$. We use the point p_{k+z+1} to construct a suitable distance parameter $2d_i$ for the following algorithm to use.

In the second step, we use the algorithm FindNeighbor$(Q_i, I, P_i, 2d_i)$ to calculate the set $L(p)$ for all points in P_i. $L(p)$ contains as many data points of various attributes as possible, and the distances between these points and point p are less than $2d_i$. Then, in machine i, we record the parameter w_p of each point, which is the number of data points centered on p.

Finally, we send the set P_i, set $L_i \leftarrow \cup_{p \in P_i} L(p)$, and parameter w_p to the central machine. Because the set L_i contains at most one point of each attribute, we know that there are only be at most m points in L_i. And because $|P_i| = k + z$, we know that all distributed machines only deliver at most $m(k + z)$ points to the central machine.

We run Algorithm 7 in the central machine. Algorithm 7 also has three main steps. In the first step, the central machine receives information (P_i, L_i, w_p) from all distributed machines. Then, we use the algorithm DistributeOutliers $(P', k, 5d, 11d)$ to find a candidate center point set P.

In the second step, we use the algorithm FindNeighbor$(L', I, P, 5d)$ to find a multi-attribute point set $N(p)$ for each $p \in P$. According to the parameters $5d$, we know that the distances from all points in $N(p)$ to point p are less than or equal to $5d$. Since the distances among points in P are greater than $11d$, we know that for different point p, set $N(p)$ are disjoint.

Finally, we use the algorithm FindMatching$(\{N(p) : p \in P\}, I, \overline{k})$ to find and return a feasible solution S that intersects as many $N(p)$ as possible.

Definition 2 Suppose q is stored by a distributed machine i. This means point q is in the data point set Q_i. We use $cen(q)$ to denote a point in P_i whose distance from point q is less than or equal to $2d_i$.

Since the data storage machine only sends some candidate center points to the central machine during Algorithm 6, it is very likely that some points of the optimal center point set O^* of the problem are discarded during the operation of Algorithm 6.

In the following Lemma 3, we show that even if some points in the optimal solution O^* are deleted, the performance of the points selected by Algorithm 6 and the optimal solution is not very different.

Lemma 3 The set $L' = \cup_{i=1}^{l} L_i$ of data points we send to the central machine contains set B. We have that the maximum distance between the points in set $P' = \cup_{i=1}^{l} P_i$ and set B is at most $5d$.

Proof For any point $c \in O^*$ that is processed by distributed machine i, by Definition 2, we have that $\delta(c, cen(c)) \leqslant 2d_i$. Note that the output set $L(cen(c))$ of the algorithm FindNeighbor( ) in Algorithm 6 contains one point from each attribute, and the distances between these points and $cen(c)$ are less than or equal to $2d_i$. Therefore, we know that $L(cen(c)) \subseteq L_i$ must contain a point c', and c' has the same attribute as point c, so we have that $\delta(c', cen(c)) \leqslant 2d_i$. Next, according to the triangle inequality, we can obtain $\delta(c, c') \leqslant 4d_i \leqslant 4d$.

We define a new set $B = \{c' : c \in O^*\}$, we have that $B \subseteq L' = \cup_{i=1}^{l} L_i$. Since B contains exactly as many points from any attribute as O^*, we know that B also satisfies the fairness constraint as well as O^*. Also, we know that the distance between any two points in sets B and O^* will not be greater than $4d$. The distance between the point in O^* and the point in $P' = \cup_{i=1}^{l} P_i$ is less than or equal to d because $\cup_{i=1}^{l} P_i \subseteq X$. Therefore, we have that the maximum distance from the point in set $P' = \cup_{i=1}^{l} P_i$ to the point in set B doese not exceed $5d$. ■

Next, we give present the main theorem of this section and the corresponding proof. Our final output of Algorithm 7 for the distributed fair k-center problem with outliers has an approximate ratio of 18.

Theorem 4 Algorithm 7 is an 18-approximation algorithm for the distributed fair k-center problem with outliers.

Proof According to Lemma 3, L' contains a feasible solution B, whose distance from P' is no more than $5d$. For each $p \in P \subseteq P'$, we use b_p to denote a point in B that is within a distance of $5d$ from point p. Due to the distance between any two points in set P being greater than 11d, it can be known that all bp points do not intersect. According to the properties of the algorithm FindNeighbor( ), set $N(p)$ it returns contains a point

b'_p that has the same properties as point b_p. Let $B' = \{b'_p : p \in P()\}$; for each point $p \in P$, we know that set B intersects all $N(p)$ since b_p and b'_p have the same properties and b_p are disjoint. B' also contains as many points of arbitrary properties as B. Since B is a feasible solution, B' is also a feasible solution. Therefore, there is a feasible solution whereby B' intersects all $N(p)$. At the same time, we have that the output S of the algorithm FindMatching ( ) is a feasible solution set that intersects as many $N(p)$ as possible. It can thus be seen that S also intersects all $N(p)$.

Now we know that the distance between any point in set P and any point in set S will not exceed $5d$. For all $p \in P$, S intersects $N(p)$, the solution P returned by the first step of Algorithm 7 is at most $11d$ from the set P'. Also, the distance between set Q and set P' is at most $2d$, because the distance between Q_i and each P_i is at most $2d$. Adding these three distances together, we have that the distance between sets Q and S is at most $18d$. ■

We have proved above that the approximate ratio of Algorithm 7 is 18. Next, we need to prove that Algorithm 7 selects no more than z outliers during the operation of the algorithm.

First, we use $O_1, O_2, \ldots, O_k$ to denote those clusters in the optimal solution. All the clusters in the optimal solution are a subset of Q and do not include the outliers that correspond to the optimal solution. We want to show that the set of data points removed from Q' in the algorithm DistributeOutliers ( ) can be mapped one-by-one to some clusters in the optimal solution when Algorithm 7 selects each center. Also, at the end of the algorithm DistributeOutliers ( ) there should be at most z points left in Q', which are the outliers of Algorithm 7.

For all points $v \in Q_i$, we use $c(v)$ to denote the point in P_i that is closest to v. That is, the point $c(v)$ is the closest center point to v found by the algorithm Greedy ( ). For the output of the algorithm DistributeOutliers ( ), $P = \{p_1, p_2, \ldots, p_k\}$, we sort them in the order in which p is added to set P.

If for all $u \in O_i$, we have that $c(u) \in Q'$ before p_j is added to P, then we call that the optimal cluster O_i is good in the j-th iteration of the algorithm DistributeOutliers ( ). That is, not all $c(u)$ have been deleted. If a cluster is not good, it means that we cannot estimate its cost using the path from u to $c(u)$ to p_j, which makes it more difficult to estimate the cost of these points. Consequently, we first analyze the cost of those good clusters. Lemma 4 in the following shows that when p_j is added to P, we have that the sum of the weights of all points removed from Q' is at least as large as the number of data points contained in any one good cluster in the optimal solution.

Lemma 4 When a point p_j is added to P, then for any good cluster O_i we have the following formula $\sum_{v' \in B(p_j, 5d, Q')} w_{v'} \geqslant |O_i|$ established.

Proof First, consider any cluster O_i that is good at time j. Since the cluster is good, we have $c(v) \in Q'$ for all points $v \in O_i$ before p_j is added to P. Let $u = c(v^*)$, where v^* is the center point of the optimal solution in cluster O_i. We need to show that for all points $v \in O_i$, the distance between point u and point $c(v)$ will not be greater than $5OPT$.

To bound the distance between point u and point $c(v)$, it suffices to bound the sum of the distance of u to any point v in O_i and v to $c(v)$ by the triangle inequality. According to the definition of the optimal solution, we can know that the distance from any point $v \in O^*$ to v^* will not be greater than OPT. Because Algorithm 5 is a 2-approximation algorithm, in Algorithm 6, after Step 1, we have that each point in Q_i is at most a distance of $2OPT$ from its center point in set P_i. In other words, the distance between any point $v \in Q_i$ to center point $c(v) \in P_i$ is $2OPT$.

Thus, we have that u is at most a distance of $2OPT$ from v^*. Next, we have that $\delta(u, v) \leqslant \delta(v, v^*) + \delta(v^*, u) \leqslant OPT + 2OPT = 3OPT$. This means that u is at most a distance of $3OPT$ from any point v in O_i. Furthermore, we know that every point $v \in O_i$ is at most a distance of $2OPT$ from $c(v)$. Thus, u is at most a distance of $5OPT$ from any point $c(v)$ for $v \in O_i$. We have that $\delta(u, c(v)) \leqslant \delta(u, v) + \delta(v, c(v)) \leqslant 3OPT + 2OPT = 5OPT$.

Now we show that $\sum_{v' \in B(u, 5d, Q')} w_{v'} \geqslant |O_i|$. This is because every point $c(v)$ must be in $B(u, 5d, Q')$ according to the definition of $B(u, 5d, Q')$ and the fact that $\delta(u, c(v)) \leqslant 5OPT$. Furthermore, every point in $v \in O_i$ contributes to the weight of $c(v)$. Knowing that our algorithm always chooses the point p_j such that $\sum_{v' \in B(p_j, 5d, Q')} w_{v'}$ is maximized, this completes the proof. ■

Lemma 5 in the following says that for a point v in a cluster O_i, if we have $c(v)$ in $B(p_j, 5d, Q')$ when the point p_j is to be added to P. For this case, we can use $B(p_j, 11d, Q')$ to enclose all points in cluster O_i. It can be shown that, in this case, after p_j is added to P, for all points $v \in O_i$, all points $c(v)$ are not in Q'; that is, they are deleted.

Lemma 5 Suppose there is now a point p_j to be added to P. For some i and for some point $v \in O_i$, if we have $c(v) \in B(p_j, 5d, Q')$, then for all points $u \in O_i$, either we have $c(u) \in B(p_j, 11d, Q')$ established or $c(u)$ has already been removed from Q'.

Proof Consider a cluster O_i in the optimal solution and some point $v \in O_i$ where $c(v) \in B(p_j, 5d, Q')$. Note that it suffices to prove that $\delta(p_j, c(u)) \leqslant 11OPT$ by definition of $B(p_j, 11d, Q')$ for any point $u \in O_i$. To prove this, we will use several applications of the triangle inequality. In particular, we will construct a path from p_j, $c(v)$, v, u, and $c(u)$. According to the triangle inequality, if $\delta(p_j, c(v))+\delta(c(v), v)+\delta(v, u)+\delta(u, c(u)) \leqslant 11OPT$, the proof is complete.

Let us consider $\delta(p_j, c(v))$. This is at most $5OPT$ according to the definition of $B(p_j, 5d, Q')$ and the assumption that $c(v) \in B(p_j, 5d, Q')$. We know that $\delta(c(v), v) \leqslant 2OPT$ and $\delta(u, c(u)) \leqslant 2OPT$ according to the definitions of $c(v)$ and $c(u)$. Finally, we know that $\delta(u, v) \leqslant 2OPT$. This is because both u and v are assigned to the same center in the optimal solution; therefore, both of them are at a distance of OPT from some point. The triangle inequality dictates that they must be at most $2OPT$ from each other. Putting this all together completes the proof. Thus, we have that $\delta(p_j, c(v)) + \delta(c(v), v) + \delta(v, u) + \delta(u, c(u)) \leqslant 5OPT + 2OPT + 2OPT + 2OPT = 11OPT$. ■

At last, in the following Lemma 6, we want to show that the weights of the points in $\bigcup_{p_i:1\leqslant i\leqslant k} B(p_i, 11d, Q')$ are at least as large as the number of points in $\bigcup_{1\leqslant i\leqslant k} O_i$. Because the optimal solution to this problem contains z outliers, we have that $|\bigcup_{1\leqslant i\leqslant k} O_i| = n-z$. Also, we know that the weight contained in ball $B(p_i, 11d, Q')$ represents the number of points in the dataset. Lemma 6 proves that the number of points covered in all balls $B(p_i, 11d, Q')$ is bigger than $n - z$, so we know that the number of points uncovered by our algorithm is not bigger than z.

Lemma 6 $\sum_{i=1}^{k} \sum_{u\in B(p_i,11d,Q')} w_u \geqslant n - z$.

Proof To prove Lemma 6, we will show a one-to-one mapping of each point in $\bigcup_{1\leqslant i\leqslant k} O_i$ to a unique unit of weight in $\sum_{i=1}^{k} \sum_{u\in B(p_i,11d,Q')} w_u$. Our proof proceeds by induction. We will show that for any $0 \leqslant j \leqslant k$, each unique point in $\bigcup_{1\leqslant i\leqslant j} O_i$ can be mapped to a unique unit of weight of the points in $\bigcup_{i=1}^{j} B(p_i, 11d, Q')$ where $O_1, O_2, \ldots, O_j$ are some clusters in ordering in the optimal solution that we fix inductively.

Assume we have mapped each point in $\bigcup_{1\leqslant i\leqslant j} O_i$, to a unique unit of weight of the points in $\bigcup_{i=1}^{j} B(p_i, 11d, Q')$. Now consider the weight of the points in the set $B(p_{j+1}, 11d, Q')$. We divide the analysis into two cases. For the first case, say that for some $i' \notin \{1, 2, \ldots, j\}$ and $u \in O_{i'}$, it is the case that $c(u)$ is in $B(p_i, 5d, Q')$ for some $i \leqslant j + 1$. Then, according to Lemma 5, it is the case that just after p_{j+1} is added to P, all of the points $c(u)$ are no longer in Q' for all $u \in O_{i'}$. Thus, in this case, we map each point in $u \in O_{i'}$ to a unit of the weight of $c(u)$. Intuitively, this is the unit of weight that u contributes to $c(u)$.

Otherwise, say that the first case does not hold. Then, we know that for all $u \in O_{i'}$ and for any $i' \notin \{1, 2, \ldots, j\}$, it is the case that $c(u)$ is in Q' after p_{j+1} is added to P. According to Lemma 4, it must be that $\sum_{u\in B(p_{j+1},5d,Q')} w_u \geqslant |O_{i'}|$ for any $i' \notin \{1, 2, \ldots, j\}$. In this case, take any cluster in the optimal solution that is not $O_{i'}$ for $1 \leqslant i \leqslant j$ and fix this cluster to be O_{j+1}. Map each of the points in O_{j+1} to a unique unit of weight of the points in $B(p_{j+1}, 11d, Q')$. This completely defines the mapping.

Notice that each point must be assigned to a unique unit of weight. This is because, in the first case, each point is charged to the weight it contributes to, and then it is removed from Q'. By removing the points from Q', we can never charge them later in the second case. Furthermore, since we charge the points to the weight they contribute to, no two points are charged to the same unit of weight in the first case. In the second case, we charge each point $u \in O_{i'}$ to a unit of weight in $B(p_{j+1}, 11d, Q')$. Knowing that $B(p_{j+1}, 11d, Q')$ is removed from Q', the second case will not be charged to this weight again. Furthermore, since $B(p_{j+1}, 11d, Q')$ does not contain a point $c(u)$ where $u \in O_{i''}$ for any $i'' \notin \{1, 2, \ldots, j\}$, no weight in any point in $B(p_{j+1}, 11d, Q')$ comes from a point in $O_{i''}$ for any $i'' \notin \{1, 2, \ldots, j\}$, so no weight of the points in $B(p_{j+1}, 11d, Q')$ can be charged to the first case later. Thus, we finish the proof. ■

Finally, we present the running time and working memory analysis of the algorithm.

Lemma 7 The running time of Algorithm 6 is $O((k+z+1)(n/T)^2 + kn/T)$ and the running time of Algorithm 7 is $O(kT(k+z)+(k+z)mTk+k^2m)$. The working memory in each distributed machine is $O(n/T)$, and in the central machine, it is $O((k + z)mT)$.

Proof The main time spent in Algorithm 6 is Greedy $(Q, k+z+1)$ and FindNeighbor $(Q_i, I, P_i, 2d_i)$. In Greedy $(Q, k+z+1)$, the upper bound of the number of calculations for the distance between points should be $O((k+z+1)n^2)$. This is because the algorithm will run $(k+z+1)$ times, and each time the distance from each point to the center point set is calculated, we do not know whether k and z are of the same order as n, so the maximum number of calculations may be n^2. Also, because the number of points in each distributed machine is n/T, the running time is $O((k+z+1)(n/T)^2)$. We know from the previous conclusion that the running time of FindNeighbor $(Q_i, I, P_i, 2d_i)$ is $O(kn/T)$, so the running time of Algorithm 6 is $O((k+z+1)(n/T)^2 + kn/T)$.

The running time analysis of Algorithm 7 is similar to Algorithm 4. Let us focus on the input of Algorithm 7. The size of P' is $(k+z)T$, and the size of L' is $(k+z)mT$. Therefore, the running time of the algorithm DistributeOutliers $(P', k, 5d, 11d)$ is $O(kT(k+z))$, and the running time of the algorithm FindNeighbor $(L', I, P, 5d)$ is $O(|L'||P|) = O((k+z)mTk)$. The running time of FindMatching($\{N(p) : p \in P\}, I, \overline{k}$) is still $O(k^2 m)$, so we know that the running time of Algorithm 7 is $O(kT(k+z) + (k+z)mTk + k^2 m)$.

The working memory in each distributed machine is $O(n/T)$ because n points are evenly placed in these T machines. The working memory in the central machine is $O((k+z)mT)$ because these T machines all send $m(k+z)$ points to the central machine. ■

Considering Algorithm 4 above, we find that the calculation time has nothing to do with n, which reveals the benefits of distributed computing. The selection of T requires a trade-off. The larger T is, the less time the distributed machine uses and the more time the central machine uses, and vice versa.

5 Conclusion

In this paper, we consider two fair clustering problems with outliers, namely, the fair k-center problem with outliers and the distributed fair k-center problem with outliers. For these two problems, we give the corresponding constant approximation ratio algorithm. For the fair k-center problem with outliers, we give an algorithm with an approximate ratio of 4. For the distributed fair k-center problem with outliers, we give an algorithm with an approximate ratio of 18. In addition, the theoretical proof and analysis of the approximation ratio, running time, and running space of the algorithm are given.

Acknowledgment

This work was supported by the National Natural Science Foundation of China (Nos. 12131003, 11771386, and 11728104), the Beijing Natural Science Foundadtion Project (No. Z200002), the General Research Projects of Beijing Educations Committee in China (No. KM201910005013), the Natural Sciences and Engineering Research Council of Canada (NSERC) (No. 06446), and the General Program of Science and Technology Development Project of Beijing Municipal Education Commission (No. KM201810005005).

References

[1] M. Kay, C. Matuszek, and S. A. Munson, Unequal representation and gender stereotypes in image search results for occupations, in *Proc. 33rd Annu. ACM Conf. Human Factors in Computing Systems*, Seoul, Republic of Korea, 2015, pp. 3819–3828.

[2] M. Kleindessner, P. Awasthi, and J. Morgenstern, Fair k-center clustering for data summarization, in *Proc. 36th Int. Conf. Machine Learning*, Long Beach, CA, USA, 2019, pp. 3448–3457.

[3] A. Chiplunkar, S. Kale, and S. N. Ramamoorthy, How to solve fair k-center in massive data models, in *Proc. 37th Int. Conf. Machine Learning*, Virtual Event, 2020, pp. 1877–1886.

[4] T. F. Gonzalez, Clustering to minimize the maximum intercluster distance, *Theor. Comput. Sci.*, vol. 38, pp. 293–306, 1985.

[5] D. S. Hochbaum and D. B. Shmoys, A best possible heuristic for the k-center problem, *Math. Operat. Res.*, vol. 10, no. 2, pp. 180–184, 1985.

[6] W. L. Hsu and G. L. Nemhauser, Easy and hard bottleneck location problems, *Discrete Appl. Math.*, vol. 1, no. 3, pp. 209–215, 1979.

[7] D. Z. Chen, J. Li, H. Liang, and H. Wang, Matroid and knapsack center problems, *Algorithmica*, vol. 75, no. 1, pp. 27–52, 2016.

[8] P. K. Agarwal and J. M. Phillips, An efficient algorithm for 2D Euclidean 2-center with outliers, in *Proc. 16th Annu. European Symp. Algorithms*, Karlsruhe, Germany, 2008, pp. 64–75.

[9] S. Guha, R. Rastogi, and K. Shim, Techniques for clustering massive data sets, in *Clustering and Information Retrieval*, W. Wu, H. Xiong, and S. Shekhar, eds. New York, NY, USA: Springer, 2004, pp. 35–82.

[10] M. Hassani, E. Müller, and T. Seidl, EDISKCO: Energy efficient distributed in-sensor-network k-center clustering with outliers, in *Proc. 3rd Int. Workshop on Knowledge Discovery from Sensor Data*, Paris, France, 2009, pp. 39–48.

[11] M. Charikar, S. Khuller, D. M. Mount, and G. Narasimhan, Algorithms for facility location problems with outliers, in

Proc. 12th Annu. ACM-SIAM Symp. Discrete Algorithms, Washington, DC, USA, 2001, pp. 642–651.

[12] S. Guha, Y. Li, and Q. Zhang, Distributed partial clustering, *ACM Trans. Parallel Comput.*, vol. 6, no. 3, p. 11, 2019.

[13] X. Guo and S. Li, Distributed k-clustering for data with heavy noise, in *Proc. 32nd Int. Conf. Neural Information Processing Systems*, Montreal, Canada, 2018, pp. 7849–7857.

[14] G. Malkomes, M. J. Kusner, W. Chen, K. Q. Weinberger, and B. Moseley, Fast distributed k-center clustering with outliers on massive data, in *Proc. 28th Int. Conf. Neural Information Processing Systems*, Montreal, Canada, 2015, pp. 1063–1071.

[15] N. Ailon, R. Jaiswal, and C. Monteleoni, Streaming k-means approximation, in *Proc. 22nd Int. Conf. Neural Information Processing Systems*, Vancouver, Canada, 2009, pp. 10–18.

[16] S. Guha, A. Meyerson, N. Mishra, R. Motwani, and L. O'Callaghan, Clustering data streams: Theory and practice, *IEEE Trans. Knowl. Data Eng.*, vol. 15, no. 3, pp. 515–528, 2003.

[17] R. M. McCutchen, and S. Khuller, Streaming algorithms for k-center clustering with outliers and with anonymity, in *Proc. 11th Int. Workshop, APPROX 2008, and 12th Int. Workshop, RANDOM 2008 on Approximation, Randomization and Combinatorial Optimization: Algorithms and Techniques*, Boston, MA, USA, 2008, pp. 165–178.

[18] M. Shindler, A. Wong, and A. Meyerson, Fast and accurate k-means for large datasets, in *Proc. 24th Int. Conf. Neural Information Processing Systems*, Granada, Spain, 2011, pp. 2375–2383.

[19] B. Bahmani, B. Moseley, A. Vattani, R. Kumar, and S. Vassilvitskii, Scalable k-means++, arXiv preprint arXiv: 1203.6402, 2012.

[20] M. F. Balcan, S. Ehrlich, and Y. Liang, Distributed k-means and k-median clustering on general topologies, in *Proc. 26th Int. Conf. Neural Information Processing Systems*, Lake, Tahoe, NV, USA, 2013, pp. 1995–2003.

[21] A. Ene, S. Im, and B. Moseley, Fast clustering using MapReduce, in *Proc. 17th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, San Diego, CA, USA, 2011, pp. 681–689.

[22] B. Mirzasoleiman, A. Karbasi, R. Sarkar, and A. Krause, Distributed submodular maximization: Identifying representative elements in massive data, in *Proc. 26th Int. Conf. Neural Information Processing Systems*, Lake Tahoe, NV, USA, 2013, pp. 2049–2057.

[23] V. Cohen-Addad, C. Schwiegelshohn, and C. Sohler, Diameter and k-center in sliding windows, in *Proc. 43rd Int. Colloquium on Automata, Languages, and Programming*, Rome, Italy, 2016, pp. 19:1–9:12.

[24] P. Pellizzoni, A. Pietracaprina, and G. Pucci, Dimensionality-adaptive k-center in sliding windows, in *Proc. 2020 IEEE 7th Int. Conf. Data Science and Advanced Analytics* (*DSAA*), Sydney, Australia, 2020, pp. 197–206.

[25] P. Pellizzoni, A. Pietracaprina, and G. Pucci, k-center clustering with outliers in sliding windows, *Algorithms*, vol. 15, no. 2, p. 52, 2022.

[26] H. Ding, H. Yu, and Z. Wang, Greedy strategy works for k-center clustering with outliers and coreset construction, in *Proc. 27th Annu. European Symp. Algorithms* (*ESA 2019*), Munich/Garching, Germany, 2019, pp. 40:1–40:16.

[27] H. Zarrabi-Zadeh, Core-preserving algorithms, in *Proc. 20th Annu. Canadian Conference on Computational Geometry*, Montreal, Canada, 2008, pp. 159–162.

[28] F. Yuan, L. Diao, D. Du, and L. Liu, Distributed fair k-Center clustering problems with outliers, in *Proc. 22nd Int. Conf. Parallel and Distributed Computing, Applications and Technologies*, Guangzhou, China, 2022, pp. 430–440.

[29] S. Kale, Small space stream summary for matroid center, arXiv preprint arXiv: 1810.06267, 2020.

[30] L. E. Celis, V. Keswani, D. Straszak, A. Deshpande, T. Kathuria, and N. K. Vishnoi, Fair and diverse DPP-based data summarization, in *Proc. 35th Int. Conf. Machine Learning*, Stockholm, Sweden, 2018, pp. 716–725.

[31] S. Ahmadian, A. Epasto, R. Kumar, and M. Mahdian, Clustering without over-representation, in *Proc. 25th ACM SIGKDD Int. Conf. on Knowledge Discovery & Data Mining*, Anchorage, AK, USA, 2019, pp. 267–275.

[32] S. K. Bera, D. Chakrabarty, N. J. Flores, and M. Negahbani, Fair algorithms for clustering, in *Proc. 33rd Int. Conf. Neural Information Processing Systems*, Vancouver, Canada, 2019, pp. 4954–4965.

[33] S. Bandyapadhyay, T. Inamdar, S. Pai, and K. Varadarajan, A constant approximation for colorful k-center, arXiv preprint arXiv: 1907.08906, 2019.

[34] F. Chierichetti, R. Kumar, S. Lattanzi, and S. Vassilvitskii, Fair clustering through fairlets, in *Proc. 31st Int. Conf. Neural Information Processing Systems*, Long Beach, CA, USA, 2017, pp. 5029–5037.

[35] X. Jia, K. Sheth, and O. Svensson, Fair colorful k-center clustering, *Math. Program.*, vol. 192, nos. 1&2, pp. 339–360, 2022.

[36] M. Schmidt, C. Schwiegelshohn, and C. Sohler, Fair coresets and streaming algorithms for fair k-means, in *Proc. 17th Int. Workshop on Approximation and Online Algorithms*, Munich, Germany, 2019, pp. 232–251.

[37] I. O. Bercea, M. Groβ, S. Khuller, A. Kumar, C. Rösner, D. R. Schmidt, and M. Schmidt, On the cost of essentially fair clusterings, in *Proc. Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, Cambridge, MA, USA, 2019, pp. 18:1–18:22.

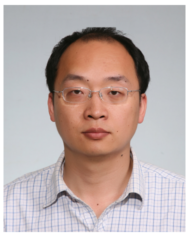
[38] M. Böhm, A. Fazzone, S. Leonardi, C. Menghini, and C. Schwiegelshohn, Algorithms for fair k-clustering with multiple protected attributes, *Oper. Res. Lett.*, vol. 49, no. 5, pp. 787–789, 2021.

[39] M. Hajiaghayi, R. Khandekar, and G. Kortsarz, Budgeted red-blue median and its generalizations, in *Proc. 18th Annu. European Symp. Algorithms*, Liverpool, UK, 2010, pp. 314–325.

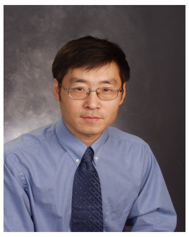
[40] R. Krishnaswamy, A. Kumar, V. Nagarajan, Y. Sabharwal, and B. Saha, The matroid median problem, in *Proc. 2011 Annu. ACM-SIAM Symp. Discrete Algorithms* (*SODA*), San Francisco, CA, USA, 2011, pp. 1117–1130.

[41] M. E. Dyer and A. M. Frieze, A simple heuristic for the *p*-centre problem, *Oper. Res. Lett.*, vol. 3, no. 6, pp. 285–288, 1985.

Fan Yuan received the BS and MS degrees from Hunan Normal University, China in 2011 and 2014, respectively. He is currently a PhD candidate at Beijing University of Technology, China. His research interests mainly include combinatorial optimization, approximation algorithm, and machine learning.



Luhong Diao received the BEng and MEng degrees in computer science from Shandong University, China in 2000 and 2003, respectively, and the PhD degree in computer science and technology from the Institute of Computing Technology, Chinese Academy of Sciences, China in 2007. From 2007 to 2010, he was a lecturer at College of Applied Sciences, Beijing University of Technology, China. Since 2010, he has been an associate professor at the Beijing Institute for Scientific and Engineering Computing, Faculty of Science, Beijing University of Technology, China. He is the author of more than 30 articles. His research interests include machine learning and computer vision. He is a member of Chinese Computer of Federation, Chine Graphics Society, China Society of Industrial and Applied Mathematics, and ACM.



Donglei Du received the PhD degrees from Chinese Academy of Sciences, China in 1996 and University of Texas at Dallas, USA in 2003, respectively. Currently, he is a professor in operations research at the Faculty of Management, University of New Brunswick, Canada. His main research interests are quantitative investment management, combinatorial optimization, approximations algorithms, robust optimization, social network analysis, algorithmic game theory, supply chain management, facility location, and machine scheduling.

Lei Liu received the BEng and MEng degrees in computer science from Shandong Normal University, China in 2000 and 2003, respectively, and the PhD degree in computer science and technology from the Institute of Computing Technology, Chinese Academy of Sciences, China in 2007. Since 2009, he has been an associate professor at the Beijing Institute for Scientific and Engineering Computing, Faculty of Science, Beijing University of Technology, China. His research interests include machine learning and text mining.