Fair k-Centers via Maximum Matching

Matthew Jones [*1] **Huy Lê Nguyễn** [*1] **Thy Nguyen** [*1]

Abstract

The field of algorithms has seen a push for fairness, or the removal of inherent bias, in recent history. In data summarization, where a much smaller subset of a data set is chosen to represent the whole of the data, fairness can be introduced by guaranteeing each "demographic group" a specific portion of the representative subset. Specifically, this paper examines this fair variant of the k-centers problem, where a subset of the data with cardinality k is chosen to minimize distance to the rest of the data. Previous papers working on this problem presented both a 3-approximation algorithm with a super-linear runtime and a linear-time algorithm whose approximation factor is exponential in the number of demographic groups. This paper combines the best of each algorithm by presenting a linear-time algorithm with a guaranteed 3-approximation factor and provides empirical evidence of both the algorithm's runtime and effectiveness.

1. Introduction

The power of Machine Learning systems has lead to a widespread increase in their use, with roles in improving the quality of healthcare and education, selecting items in a social media or news feed, job searching, and more (Holstein et al., 2019). However, the potential for these algorithms to operate in an unfair manner is not only recognized, but has been directly observed, such as in an automated hiring system favoring candidates based on age, gender, or race (Tambe et al., 2019; Cowgill, 2018; Datta et al., 2015) and in an advertisement selection algorithm which was found more likely to produce results related to searching criminal histories when using keywords which were names "typically associated with black people" (Sweeney, 2013). The existence of this underlying bias has led to a recent popularity of fair ML algorithms, which attempt to correct this social issue.

The problem in this paper particularly concerns the problem of data summarization, which involves finding a small subset of a large data set which is "approximately representative" of the whole data set. To see how bias would exist in this space, imagine an image search tool which returns some images related to a set of keywords. Inherent bias in the set of matching images can lead to bias in the set of images which should summarize all the matching images. For example, a Google Images search for the keyword "CEO" returns a much higher fraction of men than the actual fraction of male CEOs (Kay et al., 2015).

In the k-centers problem for data summarization, a few centers are chosen as the representative set of the data by choosing the set of centers such that for any data point in the set, there is some center which is sufficiently similar to the data point. Specifically, the ultimate goal of k-centers is to minimize the maximum dissimilarity between any data point and its most similar center. To introduce fairness to this problem, consider the case where the data set is partitioned into a set of groups. A k-centers algorithm can be thought of as a fair algorithm if the proportion of centers which belong to a particular group is approximately equal to the proportion of the data which belongs to the same group. This idea will be formalized in the next section, but it gives a path for introducing fairness to the k-centers algorithm: restrict the number of centers in each group such that the approximate equality between any group's share of the centers and share of the whole data set is forced.

It is known that the k-centers problem is NP-hard to solve, and is even NP-hard to approximate within a factor less than 2. Furthermore, there exists an algorithm which takes linear time in the size of the data set which yields a 2-approximation for the k-centers problem (Gonzalez, 1985). It is clear, then, that it is likewise NP-hard to solve the fair variant of the k-centers problem. However, there exist algorithms which are able to approximately solve the fair k-centers problem. Currently, for this fair variant of the k-centers problem there exists both an algorithm which takes linear time with respect to the size of the data set and

[*]Equal contribution [1]Khoury College of Computer Science, Northeastern University, Boston, Massachusetts, U.S.A.. Correspondence to: Matthew Jones <jones.m@northeastern.edu>, Huy Lê Nguyên <hu.nguyen@northeastern.edu>, Thy Nguyen <nguyen.thy2@northeastern.edu>.

Proceedings of the 37^{th} International Conference on Machine Learning, Online, PMLR 119, 2020.

achieves an approximation factor which is exponential in the number of groups (Kleindessner et al., 2019a) , as well as an algorithm which takes at least quadratic time with respect to the size of the data set but achieves a 3-approximation (Chen et al., 2016). In this paper, we bridge the gap between these two algorithms by providing an algorithm which is a strict 3-approximation algorithm for this variant of the k-centers problem and also runs in linear time with respect to the size of the data set.

2. The Problem

Given a set S and a metric d on domain $S \times S$, the k-centers problem involves choosing k centers such that the maximum distance of any item in S to a center is minimized. Formally, the optimal solution to a k-centers problem is given by

$$\arg\min_{C=\{c_1,c_2,\ldots,c_k\}\subseteq S} \max_{s\in S} \min_{c\in C} d(s,c).$$

That is, choose the set $C \subseteq S$ with cardinality k such that the maximum distance between any point $s \in S$ and the point in C closest to s is minimized. As stated earlier, the k-centers problem is NP-hard, and is additionally NP-hard to approximate within a factor better than 2 (Gonzalez, 1985). Before working on the fair variant, this paper will briefly review Gonzalez's 2-approximation algorithm for the unfair k-centers problem in Section 4.

Fairness can be introduced to the k-centers problem in the case that there are demographic groups which compose S. Formally, suppose that S is divided into m demographic groups, such that S_i is the set of points in S from demographic group i, where

$$\bigsqcup_{i=1}^{m} S_i = S.$$

To introduce fairness to the k-centers problem, each of the m demographic groups is given a value k_i, where k_i is the number of centers (out of the total k centers) which may come from demographic group i. We will assume that

$$\sum_{i=1}^{m} k_i = k$$

and the solution to the fair k-center problem is given as

$$\arg\min_{\substack{C=\{c_1,c_2,\ldots,c_k\}\subseteq S \\ \forall 1\le i\le m: |C\cap S_i|=k_i}} \max_{s\in S} \min_{c\in C} d(s,c).$$

The fact that the unfair version of the problem can be solved using the fair variant tells us that the fair variant is also NP-hard up to a 2-approximation. The best existing polynomial-time approximation for the fair k-centers problem is a 3-approximation (Kleindessner et al., 2019a). However, existing algorithms for the 3-approximation all run in a time complexity which is at least quadratic in $|S|$. This paper presents a similar algorithm which gives a 3-approximation in time linear with $|S|$.

3. Related Works

As mentioned above, our work is an improvement over (Kleindessner et al., 2019a) in approximation factor and (Chen et al., 2016) in runtime complexity. Specifically, (Kleindessner et al., 2019a) proposed an $(3 \cdot 2^{m-1} - 1)$-approximation algorithm in time $O(nkm^2 + km^4)$ for the fair k-center problem. Note that the approximation factor is not known to be tight. (Chen et al., 2016) framed the fair k-center problem as a matroid center problem, and proposed matroid intersection for a 3-approximation algorithm in $\Omega(n^2 \log n)$ time. Note that our algorithm is a $3-$approximation in time $O(nk)$, maintaining the approximation factor in (Chen et al., 2016) while having time complexity linear with the size of the dataset as in (Kleindessner et al., 2019a). In the experimental section, we also observe that not only our algorithm has better objective value than (Kleindessner et al., 2019a), it also has significantly better runtime as the number of groups increases.

(Kale, 2019) extends the idea of (Guha, 2009) for matroid center problem in streaming setting. They provide a $((17+7\epsilon)(1+\epsilon))$-approximation one-pass algorithm and a $(3+\epsilon)$-approximation two-pass algorithm with running time $O((nr + r^{3.5}) \log(1/\epsilon)/\epsilon + r^2 \log(\Delta)/\epsilon)$, where r is the rank of the matroid and Δ is the aspect ratio of the metric.

A different fairness constraint on the selected points is studied in (Chierichetti et al., 2017). For a clustering scheme to be fair, (Chierichetti et al., 2017) requires that proportion of groups in each cluster must be similar to that in the whole. This line of work defines fairness according to the disparate impact doctrine (Feldman et al., 2015). (Chierichetti et al., 2017) designed an algorithm for this fair variant of the k-center and k-medoid problems for the case of two groups. (Rösner & Schmidt, 2018) studied the fair k-center problems for multiple groups. (Bera et al., 2019) generalizes the work of (Chierichetti et al., 2017) by allowing group overlap, being applicable to any l_p norm objective, and the incorporation of restricted domination and minority protection as parameter of the fairness problem. Furthermore, (Schmidt et al., 2020) studied the fair k-means problem in the streaming model under Euclidean metric and (Kleindessner et al., 2019b) considers the spectral clustering framework with the fairness notion of (Chierichetti et al., 2017).

Another similar line of work is (Celis et al., 2018). Their paper studies the problem of fair summarization. They have the same fairness constraints that the chosen points must have k_f elements from each group. However, their

Algorithm 1 Gonzalez's 2-approximation for k-centers

Input: a set of points $S = \{s_1, ..., s_n\}$, a distance metric d, an integer k s.t. $1 \leq k \leq n$
Output: A set C such that $C \subseteq S$ and $|C| = k$
Choose an arbitrary item $s \in S$ and initialize $C = \{s\}$
Initialize an array A of size n with values $A[i] = d(s_i, s)$
for $i = 2$ **to** k **do**
  Set $c_i = \arg\max_j A[j]$
  Set $C = C \cup \{s_{c_i}\}$
  for $j = 1$ **to** n **do**
    Set $A[j] = \min\{A[j], d(s_{c_i}, s_j)\}$
  end
end
return C

objective is to maximize the diversity score of the selected points. This score is defined as the squared volume of the parallelepiped spanned by the selected subset. The key difference of this work from our work is that our method does not need access to feature representation of the data but only to the metric on the dataset.

4. The 2-Approximation Algorithm for Unfair k-Centers

Algorithm 1 is Gonzalez's 2-approximation for the k-centers problem. It builds the solution by choosing the first of the k points arbitrarily, and then choosing the remaining points sequentially as the point farthest from the points actively in the solution. We use the following well-known theorem:

Theorem 4.1. *Algorithm 1 gives a 2-approximation for the k-centers problem. (Gonzalez, 1985)*

This algorithm runs in time complexity $O(nk)$, since the algorithm involves k linear sweeps each taking $O(n)$ time.

Gonzalez's algorithm will serve as the foundation for our algorithm in this paper, similar to the algorithm in (Chen et al., 2016). Specifically, we will use Gonzalez's algorithm to solve the unfair problem, "shift" some of the k centers to satisfy fairness and guarantee a 3-approximation, and replace centers which could not be shifted using some heuristic.

5. The 3-Approximation Algorithm for Fairness

Now, we include the fairness constraint. Recall that each item $s_i \in S$ has a demographic group value f_i, and the number of centers which can be chosen for each demographic group value f is given by a value k_f. It is implicit that $k = \sum_f k_f$, so in order to have k centers in total each demographic group value f must be the demographic group value of exactly k_f centers. We use m to denote the number of demographic group values and use S_f to denote $\{s_i | s_i \in S, f_i = f\}$. Also, note that we often use talk about satisfying the fairness constraints for each k_f as an upper bound, since we will often deal with sets with less than k items as being fair.

For ease of discussion, we will also add another constraint called the *fair shift constraint*. A set of points $S' \subseteq S$ satisfies the fair shift constraint iff there is some function g from S' to S such that

- $\forall s' \in S' : d(s', g(s')) < \min_{s1,s2 \in S', s1 \neq s2} \frac{d(s_1,s_2)}{2}$
- The set $\{c \mid \exists s' \in S : g(s') = c\}$ satisfies the fairness constraint

To understand what this constraint is saying, first denote d' as the minimum distance between any pair of distinct items in S'. Balls of radius $d'/2$ around items in S' are pairwise disjoint. The mapping g on an item $s' \in S'$ must return an item which is within the $(d'/2)$-ball around s'. In addition, the set given by $g(S')$ must satisfy the fairness constraint. Therefore, the fair shift constraint is passed for a set S' if there is a set S'' with the same cardinality as S' which satisfies the fairness constraint and such that each item in S' is within distance $d'/2$ of some item in S''. We design Algorithm 3, used to test the fair shift constraint, to take the radius of the balls as an argument rather than calculating it from S', because this allows the algorithm to be useful for optimizing the fair shift later in Algorithm 2 by testing different radii for the same set S'.

5.1. A High-Level Overview: Approximation Analysis

For our complete fair k-centers algorithm, the strategy is to first use Gonzalez's algorithm for the case without fairness, and then modify the solution of Gonzalez's algorithm by moving the centers to satisfy the fairness constraint. Algorithm 2 is the outline of the overall algorithm. A complete algorithm is in section 2 of the appendix. This algorithm use graph matching to make the solution fair similar to (Chen et al., 2016) but we use carefully constructed binary searches in small domains to optimize the radius in $O(\log k)$ iterations. In the rest of this section, we will expand the outline into a complete algorithm and prove both the asymptotic runtime and competitive ratio for Algorithm 2.

We see that step 1 of the algorithm is similar to Gonzalez's algorithm, except we need to keep track of the order in which items are added to the set C in Gonzalez's algorithm. This is a simple modification, and has no effect on the asymptotic time complexity. We denote the sequence returned by this step as $\{a_1, a_2, ..., a_k\}$. We also introduce the value d_i, which is the distance between a_i and the set $\{a_1, ..., a_{i-1}\}$. Specifically, d_i is the minimum distance between a_i and an item in the sequence which precedes a_i.

Algorithm 2 Outline of the 3-approximation algorithm for k-centers with fairness

Input: a set of points $S = \{s_1, ..., s_n\}$ each with a demographic group value f_i, a distance metric d, the values k_f

Output: A set C such that $C \subseteq S$ and $|\{s_i | s_i \in C \bigwedge f_i = f\}| = k_f$ for all demographic group values f

1 Compute a sequence of k centers by choosing the first arbitrarily from S, and then iteratively choosing the point $s \in S$ which is farthest from the set S by distance.
2 Find the largest integer h such that the first h items in the sequence satisfy the fair shift constraint.
3 Find the set of points which can substitute the first h items in the sequence such that the new set of h points has at most k_f points for each demographic group f and the maximum distance between an item and its substitution is minimized.
4 Choose the remaining centers arbitrarily such that the fairness constraint is satisfied.
5 Return the k centers.

A helpful lemma at this point is:

Lemma 5.1. *The sequence $(d_2, d_3, ..., d_k)$ is non-increasing*

Proof.

$$\begin{aligned} d_{i+1} &= d(a_{i+1}, \{a_j\}_{j<i+1}) \\ &= \min\{d(a_{i+1}, a_i), d(a_{i+1}, \{a_j\}_{j<i})\} \\ &\leq d(a_{i+1}, \{a_j\}_{j<i}) \\ &\leq d(a_i, \{a_j\}_{j<i}) \\ &= d_i \end{aligned}$$

The first and last equalities hold by definition. The second equality holds because the distance from a point p to a set is the minimum of the distance between p and any item in that set. The first inequality holds by the definition of minimum, and the last inequality holds because a_i was chosen before a_{i+1} in the sequence, so it must be no closer than a_{i+1} to $\{a_j\}_{j<i}$. This simplifies to $d_{i+1} \leq d_i$, which shows inductively that the sequence is non-increasing. □

Therefore, as the number of sequence items we are considering increases, two things occur which limit the ability of the prefix with that size to pass the fair shift constraint. First, we add another item from the sequence which must be matched in order to shift centers and maintain fairness. Second, for every sequence item p we already need to match, the number of points which are "near" p is non-increasing, since $d_i/2$ is non-increasing with i by Lemma 5.1), so p has no new points it can shift to, and possibly has less such points. In these ways, our matching becomes more restricted as the number of sequence items we are considering increases.

Let the optimal solution of a specific k-centers problem with fairness require balls of radius r^* around the centers to cover S. On the same problem, we give the following lemma and prove it directly:

Lemma 5.2. $d_i > 2r^* \implies$ *the set $\{a_j\}_{j \leq i}$ satisfies the fair shift constraint*

Proof. Assume that $d_i > 2r^*$. Each item in $\{a_j\}_{j \leq i}$ must be distance $\leq r^*$ from an optimal center, by the definition of r^*. Since $d(a_j, a_k) \geq d_i > 2r^*$ for all $1 \leq j < k \leq i$, each a_j's closest optimal center is unique, by the triangle inequality. So, we can match each a_j with its closest optimal center. This gives a matching between the items in $\{a_j\}_{j \leq i}$ and points at distance $\leq r^* < d_i/2$ which satisfy the fairness constraint. □

From there, we can show that the set found by shifting $\{a_j\}_{j \leq h}$ gives us a 3-approximation:

Lemma 5.3. *Let C_h be the set of points returned by step 3 of Algorithm 2. Then, any set C such that*

- *C satisfies the fairness constraints (up to equality on all k_f)*
- *$C_h \subseteq C$*

is a 3-approximation for the k-centers problem with fairness.

Proof. For ease of notation, we define $A_h = \{a_j\}_{j \leq h}$.

First, it must be the case that $d_{h+1} \leq 2r^*$. If $d_{h+1} > 2r^*$, then the integer $h+1$ satisfies step 2 by Lemma 5.2, contradicting h as a maximum. Since $d_{h+1} \leq 2r^*$, we see $d(a_{h+1}, A_h) = d_{h+1} \leq 2r^*$. Since $d(a_{h+1}, A_h) \geq d(a, A_h)$ for all $a \in S \setminus A_h$ due to the way the sequence (a_i) was chosen, it follows that

$$\forall s_i \in S, d(s_i, A_h) \leq 2r^*$$

Finally, we must show that step 3 only shifts items in A_h by at most r^* to obtain C_h. If every item in A_h is within distance $d_h/2$ of its covering optimal center, then it can shift to that center, which must be a distance at most r^*. If any item in A_h is not within distance $d_h/2$ of its covering optimal center, then it must be the case that $d_h/2 < r^*$, and every item in A_h shifts at most $d_h/2 < r^*$. Therefore,

$$\forall a_j \in A_h, d(a_j, C_h) \leq r^*$$

By the triangle inequality,

$$\forall s_i \in S, d(s_i, C_h) \leq \min_{a_i \in A_h} d(s_i, a_i) + d(a_i, C_h)$$

$$\leq d(s_i, A_h) + r^* \leq 3r^*$$

Adding any item to C_h cannot increase $d(s_i, C_h)$. So, if $C_h \subseteq C$ then

$$\forall s_i \in S, d(s_i, C) \leq 3r^*$$

□

Which also sufficiently proves the more concise theorem:

Theorem 5.4. *Algorithm 2 gives a 3-approximation for the problem of k-centers with fairness.*

5.2. Filling in the Details: Runtime Analysis

At this point, we will expand the algorithm with details, and explain how to implement each step correctly and efficiently. As discussed earlier, we can run step 1 of Algorithm 2 similar to Gonzalez's algorithm, in $O(nk)$ time.

Steps 2 and 3 each require us to test whether we can shift to a fair set of points within some bounded pairwise distance to the original set. Specifically, given a set of points $A \subseteq S$ and a value d', we want to know if there exists a fair set $B \subseteq S$ such that the following requirements are met:

- $|B| \leq |A|$
- $\forall a \in A, d(a, B) \leq d'$

It is worth noting here that the algorithm will not require $|B| = |A|$, but by the nature of how we choose d' the sets B which we care about will always have $|B| = |A|$, since the balls of radius d' around A will be disjoint. Solving this problem is best accomplished using the faster of either a maximum bipartite matching or a maximum-flow algorithm. Note that instead of rebuilding the graph every time Algorithm 3 is invoked, we build them incrementally by editing the same graph to avoid recomputation. We will discuss how to do this shortly.

Algorithm 3 tests the fair shift constraint. First, we prove correctness of this algorithm. If there is a way to shift points in $\{a_j\}_{j \leq \ell}$ to a fair set for some ℓ such that each point moves at most d' to the new set, then the distance between a point a_j and its shift destination b_j is at most d'. Therefore, the edge from a_j to the demographic group of b_j in G is added, with label b_j. Flow from s to t can be built by edge-disjoint unit flows of the form $s \to a_j \to f_{b_j} \to t$. Exactly ℓ such flows exist, and therefore the maximum flow value is ℓ. Conversely, if a flow with value ℓ exists, then every edge from s to V_A is at capacity, and the ℓ flows from V_f to t show that the items given by the edges with labels b_j satisfy fairness up to an upper bound, since flow through each $v_f \in V_f$ is upper bounded by k_f and corresponds to the number of centers shifted into demographic group f. The ℓ labels of edges from V_A to V_f at capacity then give a set of points which are one-to-one with A with pairwise distance at most d' (since the edges exist in G) and are a fair set. Therefore, the conditions on B imposed by this algorithm are met, and we can use this to test the fair shift constraint when d' is chosen correctly.

Algorithm 3 Testing the fair-shift constraint

Input: a set of points $A = \{a_1, ..., a_\ell\}$, a value d'
Output: A fair set B such that $|A| \geq |B|$ and $\forall a \in A, d(a, B) \leq d'$, OR the empty set

1 Create a directed graph $G = \{V = \{s, t\}, E = \emptyset\}$
2 Create a vertex set V_A of size ℓ with a one-to-one mapping with points in A
3 Create a vertex set V_f with size m with a one-to-one mapping with demographic group values
4 Update V with value $V \sqcup V_A \sqcup V_f$
5 **for** *all demographic group values* f **do**
6 Update E with an edge from the vertex for f in V_f to t with weight k_f
7 **for** $i = 1$ **to** ℓ **do**
8 **if** $d(S_f, a_i) < d'$ **then**
9 Update E with an edge from the vertex for a_i in V_A to the vertex for f in V_f with weight 1 and labeled by $\arg\min_{b \in S_f} d(b, a_i)$
10 **end**
11 **end**
12 **end**
13 **for** $i = 1$ **to** ℓ **do**
14 Update E with an edge from s to the vertex for a_i in V_A with weight 1
15 **end**
16 Run Dinic's Maximum Flow Algorithm on G from s to t
17 **if** *flow value* $= \ell$ **then**
18 Calculate B from the labels of edges used in the flow
 Return B
19 **else**
20 Return $\emptyset$
21 **end**

The graph used here has size bounds $|V| = \ell + m + k + 2 = O(k)$ and $|E| \leq n + \ell + 2k = O(n)$. We get the bound n on the number of edges between V_A and V_f when we are analyzing Algorithm 3 in the context of Algorithm 2 since d' is always chosen such that balls of radius d' around A are disjoint, so each $s \in S$ yields at most one edge between V_A and V_f. Therefore, the time complexity of this algorithm is the time required to construct G, plus $O(n\sqrt{k})$ time for line 19 using a special case of Dinic's algorithm. We omit the time complexity of building G for a bit longer.

Now, we are ready to examine the time complexity and

complete correctness of Algorithm 2 using Algorithm 3, which will yield the following result:

Theorem 5.5. *There exists an algorithm which gives a 3-approximation for the problem of k-centers with fairness and which runs in time $O(nk)$.*

Line 1 of Algorithm 2 can be executed in $O(nk)$ time similar to Gonzalez's algorithm, and lines 4 and 5 can be executed in $O(n)$ time.

To accomplish line 2, we use Algorithm 3 with a binary search. That is, binary search the value ℓ over the integer range $[0, k]$, with $d' = d_\ell/2$ at each iteration. If a fair shift exists then we need $h \geq \ell$, and if a fair shift does not exist then we need $h < \ell$. This finds the correct value h (recall h is the maximum value such that a fair shift exists). We can accomplish line 3 in a similar fashion. This time, we fix $\ell = h$, and we vary the value d', to find the minimum value r such that a fair shift exists. This time, if a fair shift exists then we need $r \leq d'$, and if a fair shift does not exist then we need $r > d'$. Note that we can search over discrete values of d' corresponding to distances from $\{a_i | 1 \leq i \leq \ell\}$ to $\{S_f |$ all demographic groups $f\}$. There are at most $mk = O(k^2)$ such distances, hence both binary searches require $O(\log k)$ levels.

Since the time complexity of Algorithm 3 is $O(n\sqrt{k})$ without the graph construction, the time complexity of Algorithm 2 is the sum of 3 time complexities:

- $O(nk) + O(n) + O(n\sqrt{k}\log(k)) = O(nk)$
- The time to build all graphs G for both binary searches
- The time to find the discrete values for d' in the second binary search.

The third item is easier to address than the second, so we'll start with that. Each $s \in S$ has distance $\leq d'$ to at most one point in A_h, and these provide the candidate distances for the optimal fair shift. Hence, there are at most n such distances. We can compute these distances in $O(nk)$ time, and store the $O(n)$ distances. We reduce the number of candidate distances at each level by half using the approach of binary search, so we can use linear-time rank-finding to find all the values d' for the search in $O(n)$ time total, so the third item takes $O(nk)$ time total.

To do the second item efficiently, we don't construct G from scratch every time. Instead, we keep a base graph G' for our binary search, which represents the graph's state at some stage in the binary search, and we update G' and build G from G' throughout the binary search. In line 2 of Algorithm 2, we see that when we increase the value ℓ in the binary search, two significant changes occur with respect to G:

- The value d' decreases (or at least doesn't increase), so existing edges between V_A and V_f may need to be removed. We do this by giving each edge a threshold value r_e when it is created, so we can check each edge between V_A and V_f and remove edges with $d' < r_e$.
- We add new vertices to G, and new edges incident to those vertices. This can be accomplished by checking distances between each point in S with each new point a_i we consider, and adding edges (and setting their value r_e) as needed.

Consider a level j in the binary search in line 2 of Algorithm 2, where the search is restricted to the integer range $[l_j, u_j]$ and we need to test $\ell = (l_j + u_j)/2$. We maintain the invariant that G' is the graph when we tested $\ell = l_j$. Initially, $[l_1, u_1] = [0, k]$ and $G' = \{V = \{s, t\}, E = \emptyset\}$. Any time we add an edge, we set r_e to the distance of the shift corresponding to that edge. To compute the next value G at each level of the search, start with $G = G'$, and modify G as described earlier: check edges and remove all edges with $d' < r_e$, and add vertices and corresponding edges for each a_i, where i is between the level of G' and ℓ (including ℓ). Furthermore, when a binary search level would set $l_{j+1} \leftarrow \ell$, such that we would continue by searching the upper half of the current range, then update G' with the current value of G, since we will never need to remove any of the current vertices of G. Each iteration of the binary search then requires time $O(n)$ to check edges for $d' < r_e$, time $O(n + k) = O(n)$ to possibly copy G or G', and time $O(n(\Delta\ell))$ to obtain G from G' where $\Delta\ell$ is given by $\max\{|l_j - \ell|, |u_j - \ell|\}$ and represents the step size at that level of the binary search. The values $\Delta\ell$ are decreasing geometrically, as the step size in binary search roughly halves on each iteration. The sum over $\Delta\ell$ then equals $O(|range|) = O(k)$, hence the time to construct all the graphs for line 2 of Algorithm 2 is $O(n(k + \log k)) = O(nk)$.

In line 3 of Algorithm 2, we can keep the vertex sets constant for all graphs G, and we modify the edges between V_A and V_f. We have $O(n)$ edges which can ever exist at any level of this binary search, which we partition into three sets during the binary search: present edges, potential edges, and absent edges. If our binary search is looking at values of d' in the ranges $[r_{low}, r_{high}]$, then present edges have $r_e < r_{low}$, absent edges have $r_e > r_{high}$, and potential edges have $r_{low} \leq r_e \leq r_{high}$. By this definition, the names become clearer: absent edges will never exist in G in future binary search levels, present edges will always exist in G in future binary search levels, and potential edges are not yet decided and may or may not be included in G depending on future binary decisions. By deciding which edges are present or absent, we are only required to check potential edges on each iteration to obtain G. With this intuition, when r_{low} is increased (i.e. we are searching on

the upper half of our binary search range in later iterations), then edges with $r_{low} \leq r_e \leq d'$ become present edges. On the other hand, when r_{high} is decreased because we are searching on the lower half of our current range, then edges with $d' < r_e \leq r_{high}$ become absent edges. We only have to test potential edges at each level and the number of potential edges decreases geometrically over iterations. Thus, G' can be updated at each level to indefinitely contain present edges. For a simple upper bound on the runtime analysis, testing edges to build G and the time to copy to and from G and G' takes $O(n)$ time at each of the $O(\log k)$ levels. Therefore, the time complexity of this binary search is given by $O(n(\ell + \log k)) = O(n(k + \log k)) = O(nk)$, since we need to compute the value r_e and the closest center a_i (with $i \in [1, \ell]$) for each $s \in S$.

Therefore, our total time complexity is $O(nk) + O(nk) + O(nk) = O(nk)$, and our algorithm is a correct 3-approximation for the fair k-centers problem by Theorem 5.4, which yields Theorem 5.5.

6. Experiments

In this section, we report experimental results of our proposed fair k-center algorithm on real and synthetic datasets, in comparison with the $(3 \cdot 2^{m-1} - 1)$-approximation algorithm and heuristic methods proposed in (Kleindessner et al., 2019a). The first heuristic (Heuristic A in (Kleindessner et al., 2019a)) runs algorithm 1 with $k = k_f$ for each member of the same group f and returns the union of centers returned by each run. The second heuristic (Heuristic B in (Kleindessner et al., 2019a)) chooses centers similarly to algorithm 1 but only from groups that does not have enough centers. We also designed another heuristic (Heuristic C) that also runs algorithm 1 on each subgroup. However, instead of choosing the center that maximizes the minimum distance to the only centers in each subgroup, it selects new centers that maximize the minimum distance to all centers. Heuristic C is an improvement over Heuristic A, as the latter does not take into account distance to centers from other groups when choosing the next center. We tested two variants of algorithm 2. The difference is at the final step, once it is already guaranteed the 3-approximation and the algorithm has to pick additional centers to satisfy the fairness constraint up to equality. The fist variant is Algorithm 2-Seq, which sequentially goes through points in the datasets and selects points to be centers from groups that does not reach the required number of centers. The second variant is Algorithm 2-Heuristic, which selects the remaining centers by running Heuristic B with $k = k'_f$ for each f, where k'_f is the required number of centers from group f minus the number centers that's already selected in that group after step 3. It is also possible to run algorithm 2 with other heuristics methods.

For each algorithm, we collect objective values and runtime on every dataset for 100 random runs (performed on a PC with 3.7 GHz i3 / 8GB DDR4). For the method in (Kleindessner et al., 2019b), we utilized the implementation provided by the author for the experiments. The original implementation in (Kleindessner et al., 2019b) calculates the whole distance matrix of the data before running the clustering algorithm. To compare the runtime of the algorithms, our implementation computed the distances on the fly. For fair comparison, we slightly modified the implementation of (Kleindessner et al., 2019b) so the distance between each point is computed when needed.

6.1. Simulated Data

We generate m 4-dimensional Gaussian isotropic blobs with identity covariance matrix. The total number of points is fixed at 4000 and divided equally among each blob. Each point is randomly assigned to a group. The total number of group is m. We set $k_f = 1$ for all f, to require the algorithms to output m centers satisfying the constraint that one member from each group is represented. We report the the objective value and running time with varied m for Euclidean metric.

Table 1 shows the mean and standard deviation of objective value on simulated data. As seen in table 1, our proposed algorithms outperformed the other methods. It is also important to note that our algorithm's performance has the least variance among those tested. This shows that our algorithms are less sensitive to the choice of the initial center in algorithm 1. Figure 1 shows our algorithm tends toward significantly better runtime than (Kleindessner et al., 2019a). Although both algorithms are linear with respect to the number of samples, the algorithm's time complexity in (Kleindessner et al., 2019a) has the large order term at $O(nkm^2)$, while ours is at $O(nk)$. This explains why our algorithm is much faster as the number of groups grows. It is also worth noting that there does not seem to be a clear performance advantage to running Heuristic B over sequentially adding points after algorithm 2.

We also include more experiments for simulated data in the Appendix. In the additional experiments, we unbalance the groups by forcing one group to have disproportionately more centers than the rest.

6.2. Real data

We apply our algorithm on 4 real datasets. For each dataset, we use numeric features for clustering and selected categorical attributes as group assigment:

- **Adult (A)** dataset (Kohavi & Becker, 1996) contains socioeconomic records of 48842 individuals for prediction of whether income exceeds 50k/year. We use

Table 1. Mean and standard deviation of objective value on simulated data

Algorithms	50 Groups	100 Groups	200 Groups	400 Groups
Alg 2-Seq	**6.89 (0.2)**	6.52 (0.31)	**6.5 (0.41)**	6.46 (0.38)
Alg 2-Heu B	6.91 (0.26)	**6.48 (0.25)**	6.51 (0.43)	**6.44 (0.38)**
Kleindessner	7.01 (0.46)	6.88 (0.75)	7.45 (0.78)	7.26 (0.51)
Heuristic A	21.38 (2.84)	17.7 (1.55)	16.61 (1.57)	13.87 (1.33)
Heuristic B	7.66 (1.09)	8.16 (0.94)	7.81 (0.71)	7.8 (0.62)
Heuristic C	7.26 (1.17)	7.43 (0.87)	7.44 (0.6)	7.42 (0.62)

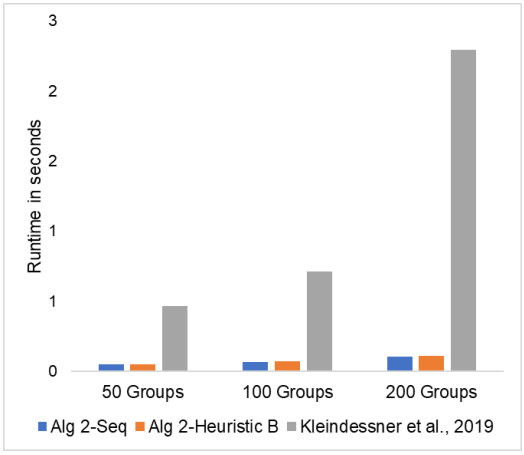

Figure 1. Mean runtime in seconds on simulated data

gender and *race* for group assignment.

- **Wholesale (W)** dataset (Cortez, 2014) contains monetary spending on various products of clients of a wholesale distributor. We select the *location* attribute for group assignment.
- **Student (S)** dataset (Cardoso, 2014) includes information about grades, socioeconomic, and school data relevant to predicting academic performance of students on Math. We use features *sex*, *address* (urban or rural) and student's *school* for group assignment.

The required number of centers from each group is proportional to the size of that group. This makes sure that every group is fairly represented in selected centers. Our constraint is based on the fairness notion of disparate impact (Feldman et al., 2015) for classification setting. Equivalently, we require the selected centers to include a percent p of members from every group. For A-Gender and A-Race, we select $p = 0.4\%$. For the other datasets, $p = 5\%$. Table 2 shows the summary of the real datasets. In the appendix, we include further experiments with the constraint that requires an equal number of centers for each group. We utilize the Euclidean distance as the metric for both settings.

Table 3 show the mean and standard deviation of objective values on real data. Our algorithms perform at least similar or better than the algorithm proposed in (Kleindessner et al., 2019a). Similar to the simulated data, we observe little performance difference between Alg-2 Seq and Alg2-Heu B. In constrast to the simulated data, heuristics methods have reasonably good performance on real dataset. This is also observed in (Kleindessner et al., 2019a). Figure 2 and 3 show our algorithms have better runtime than (Kleindessner et al., 2019a) in all settings. It is worth noting that our algorithms have a similar or better objective value on the real datasets, while having significantly better runtime.

Table 2. Summary of real dataset: number of samples, dimensionality, and count of each class for each attribute

Adult: n=32561, d=6, gender = (10771, 21790), race = (311, 1039, 3124, 271, 27816)
Student: n=649, d=16, school = (423, 226), address = (197, 452), sex = (383, 266)
Wholesale: n=440, d=6, region = (77, 47, 316)

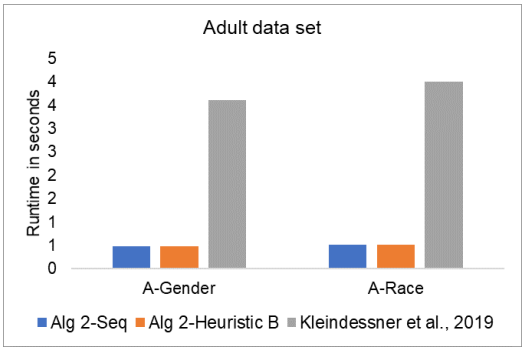

Figure 2. Mean runtime in seconds on adult dataset

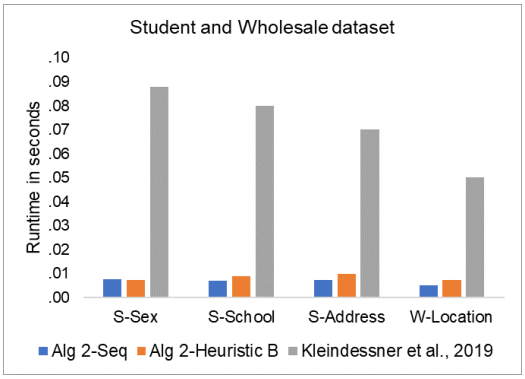

Figure 3. Mean runtime in seconds on student and wholesale dataset

6.3. High Number of Centers

We provide the experiments for synthetic data for case when the number of centers is large. Specially, we require that each demographic group has $p = 70\%$ of its points selected as centers. We set $n = 4000$ and use the same approach

Table 3. Mean and standard deviation of objective value on real data

Algorithms	A-Gender	A-Race	S-Sex	S-School	S-Address	W-Location
Alg 2-Seq	**0.32 (0.01)**	**0.32 (0.01)**	1.29 (0.04)	1.3 (0.04)	1.31 (0.05)	0.26 (0.01)
Alg 2-Heu B	**0.32 (0.01)**	**0.32 (0.01)**	**1.28 (0.03)**	**1.28 (0.04)**	**1.3 (0.04)**	0.26 (0.01)
Kleindessner	0.36 (0.03)	0.34 (0.02)	1.29 (0.05)	1.29 (0.06)	**1.3 (0.05)**	0.27 (0.03)
Heuristic A	0.41 (0.02)	0.35 (0.03)	1.36 (0.02)	1.39 (0.04)	1.37 (0.04)	0.28 (0.01)
Heuristic B	0.37 (0.02)	**0.32 (0.01)**	1.29 (0.03)	1.3 (0.04)	**1.3 (0.04)**	0.27 (0.01)
Heuristic C	0.4 (0.02)	**0.32 (0.02)**	1.29 (0.03)	1.29 (0.02)	1.35 (0.05)	**0.24 (0.02)**

in section 6.1 to generate synthetic data. Our experiments confirm that our algorithm still has favorable performance and better runtime.

As observed in table 4 and figure 4, our approach is much faster and provide better performance compared to (Kleindessner et al., 2019b).

Table 4. Mean and standard deviation of objective value on stimulated data for larger number of centers

Algorithm	50 Groups	100 Groups	200 Groups	400 Groups
Alg 2-Seq	**1.06 (0.01)**	1.47 (0.01)	**1.73 (0.02)**	2.54 (0.04)
Alg 2-Heu B	**1.06 (0.02)**	1.46 (0.01)	1.74 (0.04)	2.52 (0.05)
Kleindessner	1.21 (0.06)	1.43 (0.06)	1.84 (0.1)	**2.36 (0.15)**
Heuristic A	2.37 (0.16)	3.23 (0.18)	3.66 (0.25)	4.10 (0.29)
Heuristic B	1.03 (0.06)	**1.31 (0.16)**	**1.73 (0.06)**	2.58 (0.16)
Heuristic C	1.65 (0.04)	1.98 (0.22)	2.74 (0.01)	3.01 (0.23)

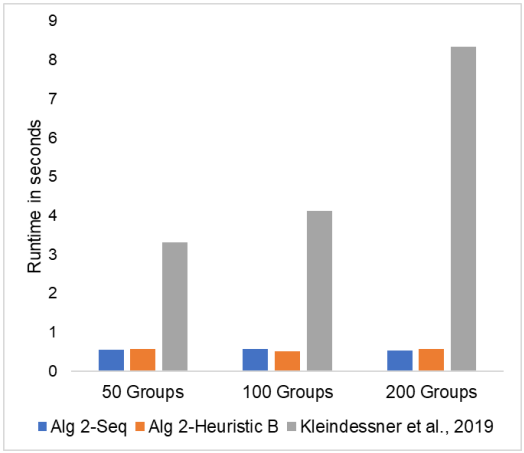

Figure 4. Mean runtime in seconds on stimulated data for large number of centers

7. Conclusion

The results of the experiments suggest that the algorithm introduced in this paper shows an improved runtime over existing algorithms for the same problem without sacrificing the quality of the result. This paper's algorithm has an asymptotic complexity of $O(nk)$, as opposed to the time complexity of $O(nkm^2 + km^4)$ in the algorithm from (Kleindessner et al., 2019a). We make the assumption that $m \leq k \leq n$, which is reasonable: we should have less centers than data points, and since we only consider demographic groups with at least one center required, we see that $m \leq k$ with equality only when each demographic group gets exactly one center. Given these assumptions, it is understandable why the prior time complexity may show better performance, especially as m increases. The experiments reflect this. On most datasets, Algorithm 2 is significantly faster, by factors between 2 and 4. The difference in runtimes is fairly stable on the real datasets, with runtimes scaling together. The significance of the m^2 factor is most noticeable in the synthetic dataset, in Figure 1, where the ratio between the previous algorithm and Algorithm 2 blows up as the number of groups increases.

The performance of Algorithm 2 is also strong, compared to the previous algorithm in (Kleindessner et al., 2019a). On the simulated data, Algorithm 2 performed the strongest of all the algorithms, as seen in Table 1. The real data results in Table 3 show that Algorithm 2 is competitive with existing algorithms. All of these datasets have very low variation in the objective values of the various algorithms, and Algorithm 2 showed the best performance in all except the Wholesale data and displayed the lowest variance.

In conclusion, Algorithm 2 shows a couple significant improvements over existing algorithms for the same problem. The algorithm is the first which is both linear in the size of the dataset and has a strict constant bound on the objective value of the result. Additionally, the time complexity and the competitiveness of Algorithm 2 are not significantly dependent on the number of demographic groups m, given $m \leq k$. In all cases, Algorithm 2 shows effective speed and competitiveness for adding fairness to the k-centers problem.

Acknowledgments

We thank the reviewers for their insightful comments and suggestions. The authors were supported by NSF grants CCF 1909314 and CCF 1750716.

References

Bera, S., Chakrabarty, D., Flores, N., and Negahbani, M. Fair algorithms for clustering. In *Advances in Neural Information Processing Systems*, pp. 4955–4966, 2019.

Cardoso, M. UCI machine learning repository. archive.ics.uci.edu/ml/datasets/wholesale+customers, 2014.

Celis, L. E., Keswani, V., Straszak, D., Deshpande, A., Kathuria, T., and Vishnoi, N. K. Fair and diverse dpp-based data summarization. *arXiv preprint arXiv:1802.04023*, 2018.

Chen, D. Z., Li, J., Liang, H., and Wang, H. Matroid and knapsack center problems. *Algorithmica*, 75(1):27–52, 2016.

Chierichetti, F., Kumar, R., Lattanzi, S., and Vassilvitskii, S. Fair clustering through fairlets. In *Advances in Neural Information Processing Systems*, pp. 5029–5037, 2017.

Cortez, P. UCI machine learning repository. archive.ics.uci.edu/ml/datasets/student+performance, 2014.

Cowgill, B. Bias and productivity in humans and algorithms: Theory and evidence from resume screening. *Columbia Business School, Columbia University*, 29, 2018.

Datta, A., Tschantz, M. C., and Datta, A. Automated experiments on ad privacy settings. *Proceedings on privacy enhancing technologies*, 2015(1):92–112, 2015.

Feldman, M., Friedler, S. A., Moeller, J., Scheidegger, C., and Venkatasubramanian, S. Certifying and removing disparate impact. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 259–268. ACM, 2015.

Gonzalez, T. F. Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, 38: 293–306, 1985.

Guha, S. Tight results for clustering and summarizing data streams. In *Proceedings of the 12th International Conference on Database Theory*, pp. 268–275, 2009.

Holstein, K., Wortman Vaughan, J., Daumé III, H., Dudik, M., and Wallach, H. Improving fairness in machine learning systems: What do industry practitioners need? In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, pp. 600. ACM, 2019.

Kale, S. Small space stream summary for matroid center. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2019)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2019.

Kay, M., Matuszek, C., and Munson, S. A. Unequal representation and gender stereotypes in image search results for occupations. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pp. 3819–3828. ACM, 2015.

Kleindessner, M., Awasthi, P., and Morgenstern, J. Fair k-center clustering for data summarization. *arXiv preprint arXiv:1901.08628*, 2019a.

Kleindessner, M., Samadi, S., Awasthi, P., and Morgenstern, J. Guarantees for spectral clustering with fairness constraints. In *International Conference on Machine Learning*, pp. 3458–3467, 2019b.

Kohavi, R. and Becker, B. UCI machine learning repository. archive.ics.uci.edu/ml/datasets/adult, 1996.

Rösner, C. and Schmidt, M. Privacy preserving clustering with constraints. In *45th International Colloquium on Automata, Languages, and Programming (ICALP 2018)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

Schmidt, M., Schwiegelshohn, C., and Sohler, C. Fair coresets and streaming algorithms for fair k-means. In Bampis, E. and Megow, N. (eds.), *Approximation and Online Algorithms*, pp. 232–251, Cham, 2020. Springer International Publishing. ISBN 978-3-030-39479-0.

Sweeney, L. Discrimination in online ad delivery. *Communications of the ACM*, 56(5):44–54, 2013.

Tambe, P., Cappelli, P., and Yakubovich, V. Artificial intelligence in human resources management: challenges and a path forward. *California Management Review*, 61(4): 15–42, 2019.