Fair k-Center Clustering in MapReduce and Streaming Settings

Suman K. Bera
Katana Graph
sumankalyanbera@gmail.com

Syamantak Das
IIIT Delhi
syamantak@iiitd.ac.in

Sainyam Galhotra
University of Chicago
sainyam@uchicago.edu

Sagar Sudhir Kale
Faculty of Computer Science, University of Vienna
sagar.kale@univie.ac.at

ABSTRACT

Center-based clustering techniques are fundamental to many real-world applications such as data summarization and social network analysis. In this work, we study the problem of fairness aware k-center clustering over large datasets. We are given an input dataset comprising a set of n points, where each point belongs to a specific demographic group characterized by a protected attribute, such as race or gender. The goal is to identify k clusters such that all clusters have considerable representation from all groups and the maximum radius of these clusters is minimized.

The majority of the prior techniques do not scale beyond $100K$ points for $k = 50$. To address the scalability challenges, we propose an efficient 2-round algorithm for the MapReduce setting that is guaranteed to be a 9-approximation to the optimal solution. Additionally, we develop a 2-pass streaming algorithm that is efficient and has a low memory footprint. These theoretical results are complemented with an empirical evaluation on million-scale datasets, demonstrating that our techniques are effective to identify high-quality fair clusters and efficient as compared to the state-of-the-art.

CCS CONCEPTS

• **Theory of computation → Unsupervised learning and clustering**.

KEYWORDS

fairness, k-center clustering, disparate impact

ACM Reference Format:
Suman K. Bera, Syamantak Das, Sainyam Galhotra, and Sagar Sudhir Kale. 2022. Fair k-Center Clustering in MapReduce and Streaming Settings. In *Proceedings of the ACM Web Conference 2022 (WWW '22), April 25–29, 2022, Virtual Event, Lyon, France.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3485447.3512188

1 INTRODUCTION

In recent years, a significant body of work have analyzed bias in various mainstream web applications, for example, pricing on e-commerce sites [16], discriminatory service offerings on the basis of race and gender in online freelance market places [17] and targeted ad placement [29]. Clustering is a fundamental algorithmic task in a typical pipeline for many of the above applications in data integration [14], recommender systems [27], customer segmentation [10], and feature generation [22, 23]. Naturally, recently, fairness in clustering has emerged as an important area of research. This is reflected in the flurry of recent works on this topic [3, 4, 6, 7, 11, 18, 19, 26] that started with the seminal work of Chierichetti et al. [11].

In this paper, we study fair clustering in the distributed and streaming models. These computational models are used when dealing with massive-scale data. We focus on the popular k-center clustering problem. In the classical k-center problem, we are given a dataset $\mathcal{X}$ of n data points embedded in some metric space and a parameter k. The objective is to find a set of k points $C \subseteq \mathcal{X}$, referred to as *centers*, and a *clustering* of $\mathcal{X}$ around these k centers. The objective is to minimize the maximum distance between a point and its closest center in C. In the *fair k-center* problem, we are additionally given a set of protected groups $X_1, X_2, \ldots, X_\ell$ ($X_i \subseteq \mathcal{X}, \forall\, i$). Fair clustering, motivated by the notion of *group fairness*, requires that the representation of the protected groups in all clusters should be commensurate with their proportion in the entire dataset.

The fair k-center problem was introduced in the seminal work of Chierichetti et al. [11], and it has been studied extensively [4, 6, 7, 18, 19, 26]. While the initial works focused on two or three protected groups [11, 26], subsequently, more general fairness constraints were considered. Ahmadian et al. [4] consider arbitrary number of non-overlapping protected groups. Their fairness criterion imposes a restriction against over-representation of any group in each cluster. Bera et al. [6] and Bercea et al. [7] strengthened this notion by assuring protection against under-representation of any group in each cluster. It is one of the most extensively studied fairness notion in the context of clustering so far. In this paper, we study the k-center clustering problem under this notion of fairness.

While the existing fair clustering algorithms are reasonably efficient in terms of computational resource for modest-sized datasets ([6] runs for $n = 30,000$ and $k = 50$ in less than 24 hours), once n and/or k increase, these sequential algorithms become impractical. The main bottleneck in all the existing approaches is solving an intermediate linear program (LP) with roughly nk variables and $O(n)$ constraints. We empirically evaluate the time taken by CPLEX, a well-known LP solver, for varying n and k. For web scale datasets like pokec (1M), HMDA (15M) and $k = 100$, the LP contains more than $\approx 10^8$ variables and CPLEX [20] does not run within 48 hours.

Motivated by the lack of existing scalable solutions, in this work, we study the problem of fair clustering for large-scale datasets. Two

WWW '22, April 25–29, 2022, Virtual Event, Lyon, France

ACM ISBN 978-1-4503-9096-5/22/04…$15.00
https://doi.org/10.1145/3485447.3512188

models are particularly popular in practice when dealing with large-scale data: the *massively parallel computation* (MPC) model and the *streaming model*. The MPC model effectively provides an abstraction for various distributed models such as the MapReduce model [5, 12]. The classical k-center problem, even under noisy settings, is well-studied in the MapReduce model [8, 13, 21, 24]. The streaming model provides a mechanism to deal with large volumes of data in a limited-memory single-core processor by making sequential passes over the data. There have been several works on the classical k-center problem in the streaming model as well [8, 9, 25].

In this work, we overcome the computational bottleneck of state-of-the-art fair k-center algorithms by designing efficient algorithms in the MPC and streaming models. Our main technical contribution is to show that one can utilize an LP of much smaller size in order to massively speed up the computation. We reduce the given problem to this small-sized LP by computing a small "summary" of the given dataset. This summary preserves small-radius properties of the original dataset. We, in fact, show that this summary can be computed quickly in the distributed or streaming environments without incurring too much loss in the quality of the solution.

Our Contributions. Our main contributions are efficient algorithms for fair k-center clustering in the MPC and streaming models of computation. We give the *first constant factor approximation algorithms* for this problem in both these models. For the MPC setting, we assume that there are m machines to process data in parallel and one central machine which we call Machine 1. Also, let Γ be the total set of protected groups in the FAIR k-CENTER instance. Our MPC algorithm uses two rounds of communication and produces a 9-approximation to the optimal FAIR k-CENTER solution.

THEOREM 1.1. *Assuming that the data is initially partitioned equally across different machines, there exists an MPC algorithm that requires two rounds of communication among the machines, communicates $O(mk|\Gamma|\log n)$ amount of data, and achieves 9-approximation for the* FAIR k-CENTER *problem with constant additive violation. Moreover, the memory used by any machine is $O(\max\{n/m, mk|\Gamma|\log n\})$.*

Our streaming algorithm makes two passes over the data stream and uses space almost linear in $k|\Gamma|$ to give a $(7+\varepsilon)$-approximation for any given $\varepsilon > 0$. We state the following result in terms of the aspect ratio Δ of the metric, which is defined as the ratio of the largest possible distance to the smallest possible distance.

THEOREM 1.2. *There is a $(7+\varepsilon)$-approximation two-pass streaming algorithm for the* FAIR k-CENTER *problem, with constant additive violation, that uses $O(k|\Gamma|\varepsilon^{-1}\log(\Delta))$ space.*

We perform an extensive evaluation of our techniques on four real-world datasets of varied scale. We used two large scale social networks, `gplus` [1] containing $\approx$ 100K records of individuals in a Google+ circle and `pokec` dataset [2] consisting of 1.6M individuals. Gender is considered sensitive in both of these datasets. Other datasets used in experiments are `credit-card` dataset with 30K data points, each referring to a credit card holder from Taiwan with Marital status as the sensitive attribute and `HMDA` dataset comprising of around 15M records of loan-level information about mortgages in the US market with race as the sensitive attribute. We demonstrate that our fair-clustering technique is upto 7 times more efficient than the state-of-the-art for million-scale datasets.

High Level Idea. Prior fair k-center clustering algorithms follow a two-step procedure. The first step is to identify a set of "potentially good" centers. An obvious choice for this is the centers identified by the vanilla k-center clustering algorithm. Vanilla k-center itself is NP-hard but one can use any standard classical approximation algorithms, e.g., greedy 2-approximation [15]. Why are these centers good in terms of the cost of the solution? Imagine any optimal fair k-center solution and say a point $x \in \mathcal{X}$ is assigned to a center $c \in C$ in that solution. Now, suppose c' is the center in the greedy k-center solution which is nearest to c and hence at a distance of at most twice the optimal cost. Now assigning x to c' instead of c only incurs an additional cost of at most twice the optimal.

Once these centers are identified, the next step is to assign all points in $x \in \mathcal{X}$ such that the fairness constraints are satisfied. It turns out to be a non-trivial problem and requires sophisticated tools based on rounding an LP. However, solving an LP is the main computational bottleneck of the existing solutions. Even the state-of-the-art [18] contains $O(\min\{2^{k-1}, nk\})$ variables in the LP and hence is difficult to scale for large values of n and k. We overcome this drawback using the distributed and streaming environments.

In the distributed MPC model (Section 3), we have the data distributed across m machines. We first compute a vanilla k-center solution on each machine in parallel. On any machine, these local centers act as "representatives" of the original data points. We encode this information using a notion of weights associated with each center that reflects the total number of points in each group that are represented by this center. Our algorithm communicates total $O(mk|\Gamma|\log n)$ bits of data in one round of MPC communication. Now we solve a weighted version of the above assignment problem on the central machine. However, the size of the LP reduces dramatically—we need only $mk^2|\Gamma|$ variables. We show a careful analysis that this local summarization and then solving the assignment problem returns a solution with constant approximation.

In the streaming model (Section 4), given a guess of the optimum value, we compute a set C of k points (potentially good centers) in the first pass such that each forgotten point has a close representative in C. At the end of the pass, we compute a fair assignment of the appropriately weighted representatives to the centers in C using the same procedure as in the MPC setting. The second pass is then used to output the assigned center for each point. We run the above procedure for $O(\varepsilon^{-1}\log(\Delta))$ guesses of the optimum value.

2 DEFINITIONS AND PRELIMINARIES

In this section, we present the notation and formally define the problem statement. Let $\mathcal{X}$ denote a set of n points embedded in a metric space with d denoting the pairwise distance function. We define the distance d between any point $x \in \mathcal{X}$ and subset $S \subseteq \mathcal{X}$ as $d_S(x) = \min_{x' \in S} d(x, x')$. Let k be an integer parameter denoting the desired number of clusters. For any positive integer p, let $[p]$ denote the set $\{1, 2, 3, \cdots p\}$. We now define the different groups of points characterized by their protected attributes.

Definition 2.1. (Protected Groups). The set $\mathcal{X}$ is partitioned into t disjoint protected groups, $\mathcal{X}_1, \mathcal{X}_2, \cdots \mathcal{X}_t$ where all points in a protected group denote the same attribute value (also referred to as color). We use Γ to denote the set of t protected attribute values.

Throughout the paper, we interchangeably use the terms protected groups and colors to mean the above partition of point set.

Definition 2.2. (k-clustering). A k-clustering of input points consists of a set of k points (also known as centers), $C = \{c_1, c_2, \cdots c_k\}$, $C \subseteq \mathcal{X}$ and an assignment function $\phi : \mathcal{X} \to C$ of each point in $\mathcal{X}$ that maps each point to exactly one center in C. All points that are mapped to the same center $c_i \in C$ are referred to as a cluster and we denote the clustering as (C, ϕ).

Problem 2.3. *(vanilla k-center) Given an input set of points $\mathcal{X}$ and a parameter k, the* vanilla k-center *problem aims to generate a k-clustering (C, ϕ) that minimizes the maximum distance of points from respective centers,* $\max_{x \in \mathcal{X}} d(x, \phi(x))$.

Note that for vanilla k-center , the ϕ corresponds to simply assigning $x \in \mathcal{X}$ to the closest center in C. The classical 2-approximation greedy algorithm by Gonzalez [15] is the best known algorithm to solve vanilla k-center —we refer to this algorithm as Greedy-kC .

Problem 2.4. *(fair k-center). Consider an input set of points $\mathcal{X}$ and a parameter k. Additionally, for each such group $\ell \in \Gamma$, we are given two input fairness parameters α_ℓ and β_ℓ. The* fair k-center *problem aims to generate a k-center clustering C, ϕ that satisfies* group fairness*: for each cluster center $c_i \in C$, and group $\ell \in \Gamma$,*

$$\alpha_\ell \leq \frac{|\{x \in \mathcal{X}_\ell : \phi(x) = c_i\}|}{|\{x \in \mathcal{X} : \phi(x) = c_i\}|} \leq \beta_\ell$$

The objective function is to minimize $\max_{x \in \mathcal{X}} d(x, \phi(x))$.

In other words, the fairness constraints require fraction of points from group ℓ in cluster C_i lies between α_ℓ and β_ℓ.

While describing our algorithms, we denote the fairness parameters concisely as vectors $(\vec{\alpha}, \vec{\beta}) \in [0, 1]^{|\Gamma|}$. We denote the optimal solution cost of vanilla k-center and fair k-center to be OPT_v and OPT_f, respectively. The following observation follows from that fact that the optimal fair k-center solution is also a feasible solution for vanilla k-center .

Observation 2.5. *For any instance of* fair k-center , $OPT_v \leq OPT_f$.

Our algorithm to solve fair k-center uses a reduction to a problem called weighted fair assignment which we define next. We are given a set of points $U \subseteq \mathcal{X}$ and let $U_1, U_2, \cdots U_t$ denote the disjoint protected groups, where $U_\ell \subseteq U$ and Γ denotes the set of all possible protected classes. Further, each point $u \in U_\ell, \ell \in \Gamma$, is associated with an integer weight w_u. We further have a set of *fixed* k centers $C = \{c_1, c_2, \ldots, c_k\}$.

Problem 2.6. *(weighted fair assignment .) The* weighted fair assignment *problem asks for an assignment function $\phi : (U \times C) \to \mathbb{N} \cup \{0\}$ such that $\sum_{c \in C} \phi(u, c) = w_u, \forall u \in U$. The assignment function, further, needs to satisfy the fairness condition. Let $W_c = \sum_{u \in U} \phi(u, c)$ and $W_{c,\ell} = \sum_{u \in U_\ell} \phi(u, c)$ be the total weight of all points and total weight of points of color $\ell \in \Gamma$, respectively, assigned to the center $c \in C$. Then the fairness condition is $\alpha_\ell \leq W_{c,\ell}/W_c \leq \beta_\ell$.*

The objective is to find ϕ that minimizes $\max_{u \in U} d_C(u)$*, where $d_C(u)$ denotes the maximum distance between a point $u \in U$ and a center $c \in C$ such that $\phi(u, c) > 0$.*

Informally, we want to "distribute" the total weight of points in U among the set of fixed centers in C in a way that satisfies the weighted fairness constraints.

Big Data Models. In this work, we consider two computational models for dealing with large volume of data: the *streaming model* and the *massively parallel computational (MPC) model.*

In the streaming model, we have limited working memory and the input data, usually too large to fit into the memory, is streamed in a sequential manner. We can, however, make multiple passes over the data (the relative ordering of the data remains the same). The goal in this model is to optimize the memory requirement as well as the number of passes over the data.

The MPC model is an abstraction of various distributed data models such as the MapReduce [5] model. In this model, the input data is distributed arbitrarily across various machines. The computation occurs in rounds. In each round, each machine performs some local computation on its input and communicates with other machines at the end of the round. In our algorithm, all machines communicated data to and from a central machine which we refer to as Machine 1. The goal of this model is to optimize the communication overhead as well as the number of rounds.

3 FAIR k-CENTER IN THE MPC SETTING

In this section, we describe the first constant factor approximation algorithm for fair k-center in the MPC setting. Algorithm 1 gives the pseudocode of the algorithm. The algorithm assumes that the input set of points are partitioned across m machines, denoted as $\{X_1, \ldots, X_m\}$ where $\bigcup_{i=1}^{m} \mathcal{X}_i = \mathcal{X}$. Our MPC algorithm operates in three phases. In the first phase, it solves vanilla k-center separately on each machine using the classical greedy algorithm [15] - let (C_i, ϕ_i) be the solution on machine i. Based on this solution, we define a weighted set of points U_i of size $|\Gamma|k$ as follows. For each center $c_i^j \in C_i, j = 1, 2, \cdots k$, and each color $\ell \in \Gamma$, we add a co-located point $x_i^{(j,\ell)}$ with color ℓ and weight $w_i^{(j,\ell)}$ equal to the total number of points $x_\ell \in X_1$ of color ℓ for which $\phi_i(x_\ell) = c_i^j$. Each machine $i = 1, 2 \cdots m$ communicates this weighted set U_i to Machine 1. On Machine 1 we solve weighted fair k-center on the set $U = \cup_{i=1}^{m} U_i$. This is also carried out in two stages. First, we obtain a set of k centers $C = \{c_1, c_2, \cdots c_k\}$ using Greedy-kC on the set U. The following lemma and theorem from [24] can be invoked directly

Lemma 3.1. *On every machine $i \in [m]$, for every point $x_i \in X_i$ $d(\phi_i(x_i), x_i) \leq 2OPT_v$.*

Theorem 3.2. *There exists a center $c \in C$ for every point $x \in \mathcal{X}$ such that $d(c, x) \leq 4OPT_v$.*

Using Observation 2.5, we can conclude that there exists a center $c \in C$ for every point $x \in \mathcal{X}$ at a distance of at most $4OPT_f$.

The final clustering is obtained by solving weighted fair assignment on the set of points U and centers C.

3.1 Algorithm for weighted fair assignment

The input to weighted fair assignment is the set of centers C as obtained in the previous section and a set of weighted points $U = \cup_{i=1}^{m} U_i$. Recall that by construction of the sets $U_i, i = 1, 2, \cdots m$

Algorithm 1: A distributed algorithm for fair k-center

Input: Points $\mathcal{X}_1, \dots \mathcal{X}_m$ distributed across m machines, number of clusters k, fairness bounds $\vec{\alpha}, \vec{\beta}$, a guess OPT_f

for $i \in \{1, \dots, m\}$ **do** /* run in parallel on individual machines */
  $(C_i, \phi_i) \leftarrow$ Run Greedy-kC (X_i, k) to obtain k centers
  $U_i \leftarrow \{\}$
  for $j \in \{1, 2, \dots, k\}, l \in \Gamma$ **do**
    $x_i^{(j,l)} \leftarrow$ Add a point co-located at c_j^i with color l
    $w_i^{(j,\ell)} = |\{v : v \in \mathcal{X}_\ell \cap X_i, \phi_i(v) = c_j^i\}|$
    $U_i \leftarrow U_i \cup \{x_i^{j,l}\}$
All weighted sets $U_i, i \in [m]$ are sent to the first machine.
$(C, \phi) \leftarrow$ Greedy-kC $(\bigcup_{i=1}^m U_i, k)$
$\gamma \leftarrow$ WtFairAssign$(\bigcup_{i=1}^m U_i, d, C, \vec{\alpha}, \vec{\beta}, 7OPT_f)$
for $i \in \{1, 2, \dots, m\}$ **do** /* run in parallel on respective machines */
  for $t \in \{1, 2, \dots, k\}, x \in U_i$ **do**
    Greedily assign points $\gamma(x, c_t)$ points from $\{v : v \in \mathcal{X}_\ell \cap X_i, \phi_i(v) = x\}$ to c_t

Algorithm 2: Algorithm for weighted fair assignment; we call this algorithm by name WtFairAssign.

Input: Weighted points U, distance metric d, set of centers C , fairness bounds $\vec{\alpha}, \vec{\beta}$ and a parameter $r \geq 0$

Initialize $\phi(x, c) \leftarrow 0, \forall x \in U, c \in C$
Solve LP-WFA : let $z^\star$ be an optimal solution.
$\phi(x, c) \leftarrow \lfloor z^\star_{x,c} \rfloor, \forall x \in U, c \in C$
Compute $\overline{w}_x, Z_c, Z_{c,\ell}, \forall x \in U, c \in C, \ell \in \Gamma$ as given in (2), (3) and (4)
Construct LP-RES using the definitions above.
while $\exists x \in U$ *such that* $\sum_{c \in C} \phi(x, c) \neq \overline{w}_x$ **do**
  Solve LP-RES : let $\overline{z}$ be a vertex solution
  for each $\overline{z}_{x,c} = 0$, remove the variable from LP-RES .
  for each $\overline{z}_{x,c} = 1$, $\phi(x, c) \leftarrow \phi(x, c) + 1$. Reduce $Z_c, Z_{c,\ell}$ by 1, remove variable $\overline{z}_{x,c}$ from LP-RES
  for each $c \in C$, if $|\{\overline{z}_{x,c} : 0 < \overline{z}_{x,c} < 1, x \in U\}| \leq 3$, remove the respective constraints of type (5b) from LP-RES
  for each $c \in C, \ell \in \Gamma$, if $|\{\overline{z}_{x,c} : 0 < \overline{z}_{x,c} < 1, x \in \mathcal{U}_\ell\}| \leq 3$, remove the respective constraints of type (5c) from LP-RES
Return the assignment ϕ

in the previous section, each point $x_i^{j,\ell} \in U_i$ corresponds to a local center $c_i^j \in C_i, j = 1, 2, \cdots k$ and belongs to color class $\ell \in \Gamma$. For notational convenience, we drop the indices i, j, ℓ while referring to a point in U in the subsequent description of the algorithms. We denote the color classes as $\mathcal{U}_1, \mathcal{U}_2, \cdots \mathcal{U}_t$. Any point $x \in \mathcal{U}_\ell$ for some $\ell \in \Gamma$ should be thought of as simply a weighted point belonging to color class ℓ in Γ. The weighted fair assignment problem can be naturally defined as an assignment problem on the bipartite graph G, where U and C form the two sides. Suppose we can correctly guess the value of OPT_f (this assumption can be easily removed through standard binary search techniques). Now define an edge between $x \in U$ and $c \in C$ if and only if $d(x, c) \leq r$, where r is a parameter which we specify shortly. The following is a linear programming relaxation for weighted fair assignment .

$$\text{LP-WFA} : \sum_{c \in C} z_{x,c} = w_x, \ \forall x \in U \tag{1a}$$

$$\beta_\ell \sum_{x \in U} z_{x,c} \leq \sum_{x' \in \mathcal{U}_\ell} z_{x',c}, \forall c \in C, \ell \in \Gamma \tag{1b}$$

$$\alpha_\ell \sum_{x \in U} z_{x,c} \geq \sum_{x' \in \mathcal{U}_\ell} z_{x',c}, \forall c \in C, \ell \in \Gamma \tag{1c}$$

$$z_{x,c} \geq 0, \forall x \in U, c \in C \tag{1d}$$

We solve the above LP to obtain a feasible solution $z^\star$, which might be fractional solution. The following crucial lemma proves that such a feasible solution always exists for a suitable choice of r.

Lemma 3.3. LP-WFA *has a feasible solution for* $r = 7OPT_f$.

The following claim in the main ingredient to prove Lemma 3.3. We show that the optimal clusters $C^\star$ can be transformed to a feasible solution of weighted fair assignment where every point is within $3OPT_f$ from its center (Claim 3.4). We can extend this to consider the output of vanilla k-center and show that the feasible solution of LP-WFA is within $7OPT_f$, thus proving the lemma.

Claim 3.4. *Let* $C^\star = \{c_1^\star, c_2^\star, \cdots c_k^\star\}$ *denote the set of centers in an optimal* fair k-center *solution for the original point set* $\mathcal{X}$*. Then there exists a solution to* weighted fair assignment *on the set of points* U *with* $C^\star$ *as the centers such that for every point* U *is assigned to a center in* $C^\star$ *within a distance* $3OPT_f$.

Proof. We show that the optimal solution of fair k-center on $\mathcal{X}$ with centers $C^\star$ can be converted to a feasible solution of weighted fair assignment such that any point in U is assigned to a center within a distance of $3OPT_f$. We iterate over all points $x_i^{(j,\ell)} \in U_i$ for $i \in [m]$, $j \in [k]$ and $\ell \in \Gamma$ and assign its total weight $w_i^{(j,\ell)}$ to different centers in C. Recall that $w_i^{(j,\ell)}$ is the number of ℓ color points in set X_i - call this set X' - such that $x_i^{(j,\ell)}$ is the closest center for all points in X' in the local vanilla k-center solution on machine i . Suppose a particular point $x' \in X'$ is assigned to center $c_p^\star$ in the optimal fair k-center solution. Then in the weighted fair assignment solution, we assign 1 unit of weight of the point $x_i^{(j,\ell)}$ to $c_p^\star$. Clearly $d(x_i^{(j,\ell)}, c_p^\star) \leq d(x_i^{(j,\ell)}, x') + d(x', c_p^\star) \leq 2OPT_v + OPT_f \leq 3OPT_f$. In the above inequality, we used the fact that $d(x_i^{(j,\ell)}, y) \leq 2OPT_v \leq 2OPT_f$ which follows from Lemma 3.1 and Observation 2.5.

Further, observe that in the above procedure, the total weight of any particular color assigned to a center $c_p^\star \in C, p \in [k]$, is exactly equal to the number of points of this color assigned to $c_p^\star$ in the fair k-center solution. Hence, the fairness constraints remain feasible in the created weighted fair assignment solution. □

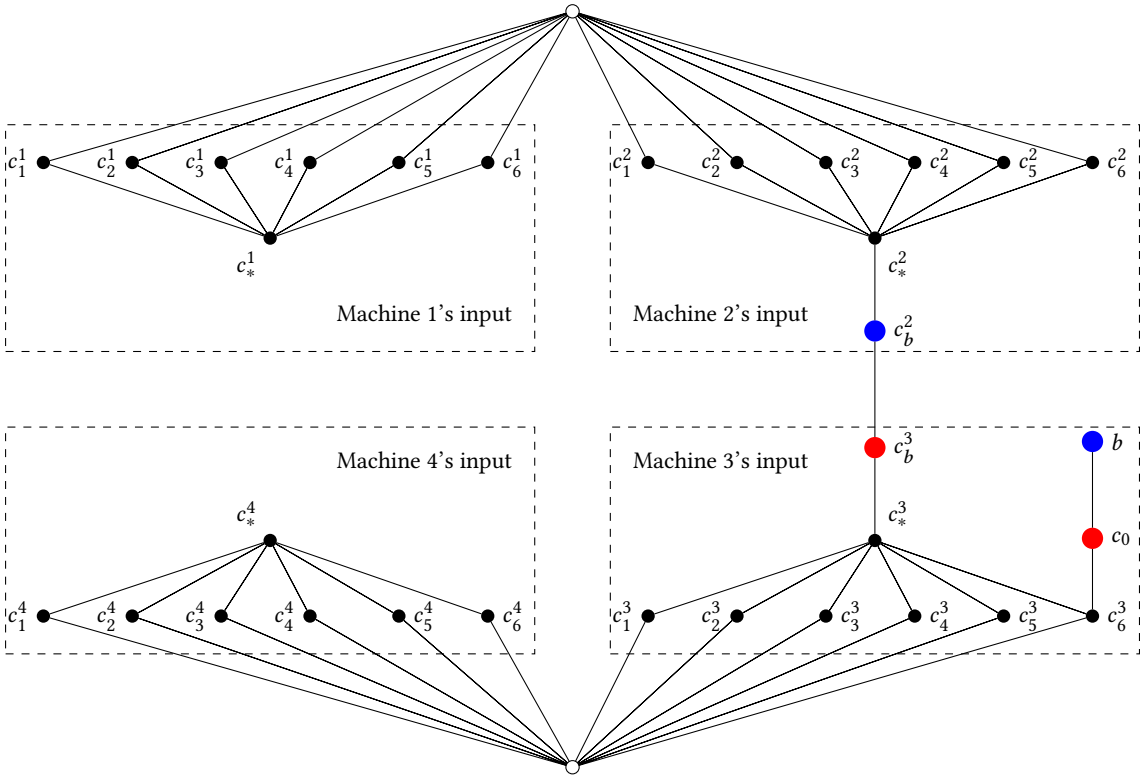

Figure 1: A tight example for the MPC algorithm with $k = 6$. Each solid circle represents a point, and the metric is the shortest path metric on the depicted graph. A hollow circle represents a vertex which is used to define the metric but is not part of the input. The fill-color of the circle denotes the color of the point: all points except b, c_0, c_b^2, and c_b^3 are black. A black point is thought of as one blue and one red point co-located. Fairness constraints: $\alpha_{\text{blue}} = \alpha_{\text{red}} = 2/3$, and $\beta_{\text{blue}} = \beta_{\text{red}} = 1/3$;

Rounding the solution $z^\star$. We present a mechanism to construct an integral solution to WEIGHTED FAIR ASSIGNMENT from the feasible fractional solution $z^\star$. We first do partial integral assignment based on $z^\star$ in the natural way – assign a weight of $\lfloor z^\star_{x,c} \rfloor$ to the edge $(x, c), \forall x \in U, c \in C$. However, note that this integral assignment is not a feasible solution to WEIGHTED FAIR ASSIGNMENT . To round the fractional parts of $z^\star$, we define a residual LP as follows.

$$\overline{w}_x = w_x - \sum_{c \in C} \lfloor z^\star_{x,c} \rfloor \tag{2}$$

$$Z_c = \sum_{x \in U} \left(z^\star_{x,c} - \lfloor z^\star_{x,c} \rfloor \right), \forall c \in C \tag{3}$$

$$Z_{c,\ell} = \sum_{x \in \mathcal{U}_\ell} \left(z^\star_{x,c} - \lfloor z^\star_{x,c} \rfloor \right), \forall \ell \in \Gamma, c \in C \tag{4}$$

The quantity $\overline{w}_x$ can be thought of as the residual weight of the point $x \in U$ after the partial assignment. Note that, since w is an integer vector, so is $\overline{w}$. Now we are ready to define the residual LP.

$$\text{LP-RES}: \sum_{c \in C} \overline{z}_{x,c} = \overline{w}_x, \ \forall x \in U \tag{5a}$$

$$\lfloor Z_c \rfloor \leq \sum_{x \in U} \overline{z}_{x,c} \leq \lceil Z_c \rceil, \ \forall c \in C \tag{5b}$$

$$\lfloor Z_{c,\ell} \rfloor \leq \sum_{x \in \mathcal{U}_\ell} \overline{z}_{x,c} \leq \lceil Z_{c,\ell} \rceil, \ \forall c \in C, \ell \in \Gamma \tag{5c}$$

$$0 \leq \overline{z}_{x,c} \leq 1, \forall x \in U, c \in C \tag{5d}$$

Claim 3.5. *Let $\overline{z}^{int}$ be a feasible integral solution to* LP-RES *. Then $\lfloor z \rfloor + \overline{z}^{int}$ forms a feasible integral solution to* LP-WFA *and hence a feasible solution to the* WEIGHTED FAIR ASSIGNMENT *instance.*

Algorithm 2 rounds a feasible fractional solution of LP-RES iteratively. We use Algorithm 2 and the following theorem from [6] for this mechanism.

Theorem 3.6. *Given a feasible solution to* LP-WFA *where the fractional assignment of each point in U is restricted to a center at distance at most r, Algorithm 2 outputs an integral assignment of all points which can violate any fairness constraint by an additive factor of at most 7. Further, any weighted point $u \in U$ is assigned to a center $c \in C$ such that $d(u, c) \leq r$.*

Remark 3.7. *We note that the algorithm of Shmoys and Tardos [28] for the generalized assignment problem can be adapted to round* LP-RES *with a violation of +2 instead of +7 that we get in Theorem 3.6. However, in practice, we found that the iterative rounding approach is scalable and have minimal violation.*

Remark 3.8. *Our algorithm can be easily adapted to the case where a single point can belong to multiple color classes in Γ. Specifically, suppose Λ is the maximum number of color classes to which a point belongs. Then our* LP-WFA *will have $O(2^\Lambda mk|C|)$ many variables. All other bounds remain exactly the same as in Theorem 3.6.*

Theorem 3.6 can now be used to prove Theorem 3.2. Firstly, by Lemma 3.3, LP-WFA has a feasible solution for $r = 7OPT_f$ and hence by Theorem 3.6 an integral solution of the same cost. Finally, we need to use this solution for the weighted point to produce an assignment of the original points in $\mathcal{X}$. To this end, any point in $x \in \mathcal{X}$ is assigned to a center $c \in C$ where c is the center to which the representative weighted point of x in U is assigned in the integral WEIGHTED FAIR ASSIGNMENT solution. Due to Lemma 3.1, we incur an additional distance of $2OPT_v \le 2OPT_f$ in the above process in the assignment cost of x to c.

3.2 A tight example on which the MPC algorithm achieves 9-approximation

Figure 1 shows an example which proves that our analysis for Algorithm 1 is tight. An optimum solution is $\{c^1_*, c^2_*, c^3_*, c^4_*\}$ with one of $\{c^2_b, c^3_b\}$ and one of $\{b, c_0\}$, say c^2_b and c_0. Each black point can be assigned to its nearest picked center, c^2_b and c^3_b can be assigned to c^2_b and b and c_0 can be assigned to c_0; this maintains the fairness conditions keeping the cost to be 1. Note that a black point is thought of as one blue and one red point co-located. Each Machine i, sends the set $\{c^i_1, \ldots, c^i_6\}$ of centers to the coordinator machine: when Machine i runs GREEDY-kC , say it starts with c^i_1, keeps adding centers that are farthest, i.e., distance 2 away, from chosen centers, and must end up with $\{c^i_1, \ldots, c^i_6\}$ (this is called U_i in Algorithm 1). Note that on Machine 3, we can further assume that GREEDY-kC does not select b in the worst case. We use Machine 1 as the coordinator machine in our algorithm that computes the set, say, $\{c^1_1, c^1_2, c^1_3, c^4_1, c^4_2, c^4_3\}$: suppose it starts with c^1_1, next it must select one of $c^4_1, \ldots, c^4_6$, all of which are at distance 9 from c^1_1. Then all the remaining points to choose from are distance 2 away from the first two points, and the worst the algorithm can do is select 3 points from Machine 1's summary U_1 and 3 points from Machine 2's summary U_2. Now, Machine 2 assigns the blue point c^2_b to, say, c^2_1, and Machine 3 assigns the red point c^3_b to, say c^3_1. Moreover, Machine 3 assigns the points b and c_0 to c^3_6. (By assign, we mean defining appropriately weighted point at the location; see the algorithm description in Section 3.) Based on these assignments by the machines, a fair assignment is computed by the coordinator machine. Now, c^3_1 and c^3_6 are at distance 7 from any of $\{c^1_1, c^1_2, c^1_3\}$, and c^2_1 is at distance 7 from any of $\{c^4_1, c^4_2, c^4_3\}$. In any fair assignment, either the blue point at c^2_1 must be assigned to one of $\{c^4_1, c^4_2, c^4_3\}$ or the red point at c^3_1 must be assigned to one of $\{c^1_1, c^1_2, c^1_3\}$. This means that the cost of any fair assignment is at least 7. Moreover, the fair-assignment routine can output that the blue point at c^3_6 be assigned to one of $\{c^1_1, c^1_2, c^1_3\}$. The blue point at c^3_6 is in fact b, which is at distance 9 from any of $\{c^1_1, c^1_2, c^1_3\}$, giving us an approximation ratio of 9. To address the fact that our algorithm allows for constant additive violation in fairness constraints, we just increase the number of "copies" of each point by a large enough constant.

4 A TWO-PASS STREAMING ALGORITHM

We give a $(7+\varepsilon)$-approximation two-pass streaming algorithm that uses $O(k|\Gamma|\varepsilon^{-1}\log(\Delta))$ space, where Δ is the aspect ratio of the metric, which is the ratio of the largest possible distance to the smallest possible distance. Suppose we are given a guess τ for the optimum. In the first pass, the algorithm computes a maximal set of centers C such that the distance between any two centers is greater than 2τ; then $|C| \le k$. Each point x is within distance 2τ of C. The algorithm chooses a center $c \in C$ as a representative of x that arrived earliest among all centers within 2τ of x; this center is called $c(x)$. This gives us an instance of fair k-center that is more succinct to describe than the original instance. This instance has a fair assignment of cost at most 5τ (we will prove this). Using WTFAIRASSIGN, we solve this instance, and use the second pass to output for each x the assigned center from C; this assignment has cost at most 7τ, which gives us the approximation ratio. See Algorithm 3: for a point x and an ordered set of points C, define Earliest(x, C, R) to be the earliest point c in C such that $d(x, c) \le R$. Also, in the pseudocode, the operation $\bigcup_{c\in C}$ Profile$[c]$ is to be thought of as for each $c \in C$, creating Γ copies of c with weight of copy γ to be Profile$[c][\gamma]$ and color set to γ. Hence, for each forgotten point x, its "copy" is present at its representative $c(x)$.

Algorithm 3: A streaming algorithm for fair k-center

Input: Set of points $\mathcal{X}, \vec{\alpha}, \vec{\beta}$, Guess τ

$C \leftarrow \emptyset$
for $x \in \mathcal{X}$ **do** // The first pass
  if $d(x, C) > 2\tau$ **then**
    $C \leftarrow C \cup \{x\}$
    Profile$[x] \leftarrow 0^\Gamma$; Profile$[x]$[Color(x)] $\leftarrow 1$
  else
    $c \leftarrow$ Earliest$(x, C, 2\tau)$
    Profile$[c]$[Color(x)] $\leftarrow$ Profile$[c]$[Color(x)] $+ 1$
Asgn $\leftarrow$ WTFAIRASSIGN$(\bigcup_{c\in C}$ Profile$[c], d, C, \vec{\alpha}, \vec{\beta}, 5\tau)$
Asgn(c, γ, c') denotes the number of points represented by
    c of group γ that are assigned to c'
for $x \in \mathcal{X}$ **do** // The second pass
  $c(x) \leftarrow$ Earliest$(x, C, 2\tau)$.
  Let c' be earliest center in C such that
      Asgn$(c(x)$, Color$(x), c') > 0$.
  x is assigned to the center c'.
  Decrement Asgn$(c(x)$, Color$(x), c')$ by 1.

Let I be the original fair k-center instance. We denote by $I(C)$ the instance of fair k-center where at each center $c \in C$, we place Profile$[c][\gamma]$ points of color γ.

LEMMA 4.1. *Let $\tau \le$ cost$(OPT_f(I))$. Then* cost$(OPT_f(I(C))) \le 5\tau$, *and* WTFAIRASSIGN *returns a fair assignment ϕ'_c with constant additive violation and cost at most 5τ.*

PROOF. We show that there is a fair assignment of cost at most 5τ for $I(C)$. Consider any point x. Fix an optimum $OPT_f(I)$ for I. Let $\phi^*(x)$ denote the optimum center for x in I; then $d(x, \phi^*(x)) \le \tau$. For $x \in \mathcal{X}$, define $\phi(x) := c(\phi^*(x))$ (recall that $c(x)$ is defined in the algorithm). Now, ϕ is a fair assignment because all points in a cluster in $OPT_f(I)$ are assigned to the same center in C.

Moreover, by the triangle inequality and the facts $\phi(x) = c(\phi^*(x))$ and $d(\phi^*(x), c(\phi^*(x))) \le 2\tau$:

$$d(x, \phi(x)) \le d(x, \phi^*(x)) + d(\phi^*(x), c(\phi^*(x))) \le 3\tau. \quad (6)$$

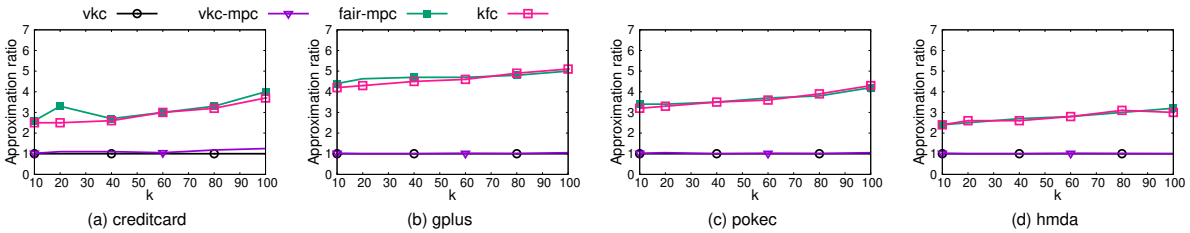

Figure 2: k-center objective comparison for varying k in the MPC setting. All values are relative to vkc.

We define ϕ_c that maps points in $I(C)$ to C based on ϕ in a natural way. Consider a point x_c in $I(C)$. Let $x(x_c)$ denote the (original) point in $\mathcal{X}$ whose "copy" is x_c. Then define $\phi_c(x_c) := \phi(x(x_c))$; by construction, ϕ_c is also fair. Now we bound $\text{cost}(OPT_f(I(C)))$. For any point x_c in $I(C)$, the distance:

$$d(x_c, \phi_c(x_c)) \leq d(x_c, x(x_c)) + d(x(x_c), \phi(x(x_c))) \leq 5\tau,$$

by the triangle inequality, noting that $\phi_c(x_c) = \phi(x(x_c))$, and (6). The lemma follows by the guarantees of WtFairAssign. □

Lemma 4.2. *Algorithm 3 outputs a fair solution of cost at most 7τ with constant additive violation if $\tau \leq cost(OPT_f(I))$.*

Proof. In the second pass, our algorithm assigns a center to each point x in $\mathcal{X}$ based on the output of WtFairAssign. This assignment is similar to first moving x to $c(x)$, which is at most 2τ away, then using the assignment ϕ'_c referred in Lemma 4.1, resulting in another additive 5τ in the cost. The proof is finished by Lemma 4.1. □

If we have a lower bound L and an upper bound U such that $L \leq \text{cost}(OPT_f(I)) \leq U$, then by running Algorithm 3 for $\tau = L, L(1+\varepsilon), L(1+\varepsilon)^2, \ldots, U$, we get a $(7+\varepsilon)$-approximation algorithm. If we do not have such bounds, then we can use the aspect ratio Δ.

Theorem 4.3. *There is a $(7+\varepsilon)$-approximation two-pass streaming algorithm for the fair k-center problem, with constant additive violation, that uses $O(k|\Gamma|\varepsilon^{-1}\log(\Delta))$ space.*

5 EXPERIMENTS

In this section, we evaluate our techniques over a variety of real-world datasets. We answer the following questions. **Q1:** Are the presented algorithms effective in identifying fair clusters and ensuring low k-center objective? **Q2:** How does the quality of generated clusters compare with the state-of-the-art? **Q3:** Are the presented algorithms scalable to million-scale datasets?

Datasets. We consider the following real-world datasets.

(1) `credit-card` dataset contains 30K data points, each referring to a credit card holder from Taiwan. Marital status of individuals is used as the sensitive attribute to generate two protected groups, and the individuals are clustered based on their billing details.
(2) `gplus` [1] contains around 100K records of individuals in a Google+ circle. Gender is considered sensitive and embedding of textual features are used for distance estimation.
(3) `pokec` dataset [2] contains 1.6M records of individuals from one of the largest social networks in Slovakia. Gender is considered sensitive and individuals are compared based on their features including hobbies, interests, etc.
(4) `HMDA` dataset comprises of around 15M records of loan-level information about mortgages in the US market and race is considered as the sensitive attribute.

Baselines. We compare our techniques with the following baselines for the MPC setting.

(1) `vkC` denotes the greedy single-machine algorithm for vanilla k-center clustering. `vkC` is known to achieve 2-approximation of the optimal clustering objective, and identifying clusters that identify clusters with lower radius is NP-hard.
(2) `fkC` denotes the fair k-center clustering algorithm [6] that assumes access to all points on a single machine.
(3) `vkc-mpc` denotes the vanilla k-center clustering algorithm in the MPC setting that does not guarantee fairness. It is a 2-round algorithm that is guaranteed to achieve 4-approximation of the optimal clustering objective.
(4) `kfc` [18] corresponds to the state-of-the-art fair clustering technique for k-center objective. It is a two-step procedure that first runs `vkC` to identify k-centers and then generates an LP with $O(\min(2^{k-1}, nk))$ variables.

Our algorithm is denoted by `fair-mpc` for the MPC setting and `fair-str` for the streaming setting. In the streaming setting, we compare `fair-str` with the state-of-the-art coreset based fair k-center algorithm [19] with its parameter $\epsilon = 0.5$ (denoted as `coreset`). To compare the objective values, we plot the ratio of the k-center objective of different techniques with that of `vkC`.

Setup. All techniques were implemented in Python 3.6 and we used IBM CPlex to solve the linear program. The MPC algorithm is tested on a cluster of 10 machines. We used the original implementation of `kfc`. We chose the default parameter settings for `kfc` and used $\delta = 0.01$ fairness parameter for all algorithms. For `fair-mpc`, we increased the number of rounds of `vkc-mpc` such that the number of variables in the LP is less than 50K.

5.1 Evaluation Results

Solution Quality. Figure 2 compares the quality of clusters identified by our technique along with other baselines. Since the clustering objective has been normalized with respect to the best clustering objective, `vkC` achieves the normalized approximation ratio of 1. Theoretically,`vkc-mpc` provides a 4-approximation of the best k-center objective as compared to 2-approximation by `vkc`. However,

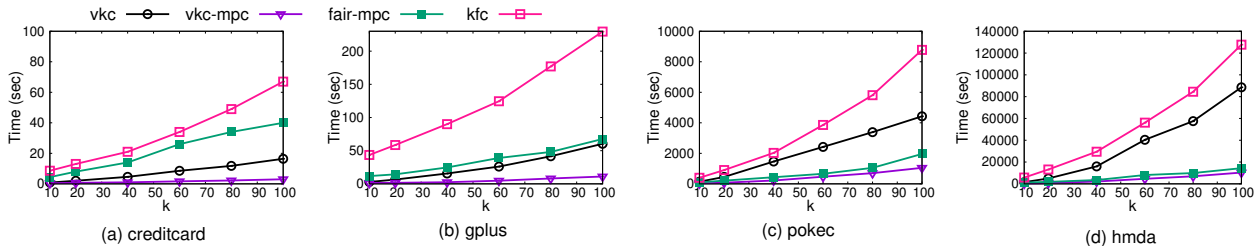

Figure 3: Running time comparison in the MPC setting.

we observe that it is consistently within a factor of 1.3 of vkc, validating its effectiveness to identify low radius clusters in practice. However, vkC and vkc-mpc do not ensure fairness and the identified clusters contain a skewed representation of different colors.

Among the techniques that ensure fairness, we do not plot fkc as it did not finish for $k > 50$ for (creditcard and gplus, and $k > 10$ for million-scale datasets. fair-mpc and kfc achieve similar clustering objective for all values of k. The k-center objective of fkc and fair-mpc is generally expected to be much worse than vkc due to an additional fairness constraint incorporated while clustering. However, empirically these techniques have approximation ratio less than 7 across all datasets (better than 4 for the majority of the cases). One of the reasons for this behavior is the presence of nodes of different colors across different regions in the euclidean space.

Running Time. Figure 3 compares the running time for MPC setting. Across all datasets and values of k, vkc-mpc requires the least amount of time because of parallel processing of points across different machines and it does not ensure fairness. fkC is the slowest technique and does not run for larger values of k across all datasets. fkc requires more than two times the time taken by kfc for smaller datasets and more than 9 times for large scale datasets. We provide a high level overview of these techniques to explain these differences. Both fair-mpc and kfc perform vanilla clustering over the input points to identify k centers and then use an LP solver to optimize for fair assignment of points. fair-mpc runs vkc-mpc to identify k centers and then writes an LP with $O(k^2)$ variables (independent of n). In contrast, kfc uses vkC to identify centers and the assignment LP contains $O(\min(2^{k-1}, nk))$ variables. Since LP solver is the bottleneck in the execution of these techniques, the running time of kfc grows faster than that of fair-mpc.

Effect of number of machines. In this experiment, we considered the credit-card dataset with $k = 50$ and varied the number of machines from 5 to 50. We observed a minor improvement in the quality of clusters but it increased the number of variables in the assignment LP. Therefore, the time taken by assignment phase increased from around 20 seconds to 450 seconds for 50 machines.

Streaming Setting. In this experiment, we test the effectiveness of fair-str as compared to coreset and other baselines. We observe that the quality of clusters identified by fair-str is very similar to that of fair-mpc (Figure 4) but the running time is similar to that of vkC (Figure 5). The main reason for this is $O(nk)$ complexity to identify the k centers. The overhead of solving a linear program is not high because the number of variables is independent of n. Among baselines, coreset generates fair clusters but is slower than fair-str because the number of variables in its fair assignment

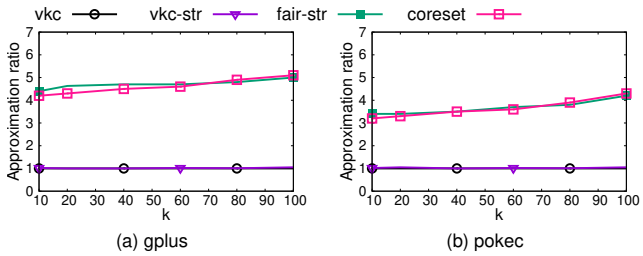

Figure 4: k-center objective comparison in the streaming setting for varying k.

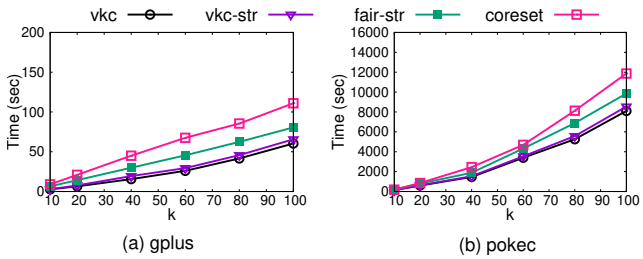

Figure 5: Running time comparison in the streaming setting.

LP rely exponentially on the dataset dimension as compared to fair-str, which contains $O(k)$ variables in the LP.

> **Key Takeaways.** Our experiments in MPC setting demonstrate that the running time of our methods is more than 7 times better than kfc for million scale datasets. Additionally, the quality of identified clusters is stable across parameters like number of machines. Overall, our presented techniques are highly efficient and generate low-radius clusters for all values of k.

6 CONCLUSION

In this work, we design scalable techniques to perform fair k-center clustering over large-scale datasets. We consider two commonly studied models: the massively parallel computation model and the streaming model. Our key contribution is a simple algorithm to formulate a linear program where the number of variables and constraints is independent of n. We prove constant factor approximation of our techniques and empirically demonstrate its efficacy.

REFERENCES

[1] [n.d.]. Google+ dataset, SNAP, http://snap.stanford.edu/data/ego-Gplus.html.

[2] [n.d.]. Pokec dataset, SNAP, http://snap.stanford.edu/data/soc-Pokec.html.

[3] Saba Ahmadi, Sainyam Galhotra, Barna Saha, and Roy Schwartz. 2020. Fair Correlation Clustering. *CoRR* abs/2002.03508 (2020). arXiv:2002.03508 https://arxiv.org/abs/2002.03508

[4] Sara Ahmadian, Alessandro Epasto, Ravi Kumar, and Mohammad Mahdian. 2019. Clustering Without Over-Representation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 267–275.

[5] Paul Beame, Paraschos Koutris, and Dan Suciu. 2017. Communication Steps for Parallel Query Processing. *Journal of the ACM (JACM)* 64, 6 (2017), 40.

[6] Suman Bera, Deeparnab Chakrabarty, Nicolas Flores, and Maryam Negahbani. 2019. Fair algorithms for clustering. In *Conference on Neural Information Processing Systems*. 4955–4966.

[7] Ioana O. Bercea, Martin Groß, Samir Khuller, Aounon Kumar, Clemens Rösner, Daniel R. Schmidt, and Melanie Schmidt. 2019. On the cost of essentially fair clusterings. In *International Workshop on Approximation Algorithms for Combinatorial Optimization Problems*.

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

[9] Deeparnab Chakrabarty, Sanjeev Khanna, and Shi Li. 2015. On (1, ε)-restricted assignment makespan minimization. In *Annual ACM-SIAM Symposium on Discrete Algorithms*.

[10] Daqing Chen, Sai Laing Sain, and Kun Guo. 2012. Data mining for the online retail industry: A case study of RFM model-based customer segmentation using data mining. *Journal of Database Marketing & Customer Strategy Management* 19, 3 (2012), 197–208.

[11] Flavio Chierichetti, Ravi Kumar, Silvio Lattanzi, and Sergei Vassilvitskii. 2017. Fair clustering through fairlets. In *Proc. 31st Conference on Neural Information Processing Systems*. 5029–5037.

[12] Jeffrey Dean and Sanjay Ghemawat. 2004. MapReduce: Simplified Data Processing on Large Clusters. In *6th Symposium on Operating System Design and Implementation (OSDI 2004), San Francisco, California, USA, December 6-8, 2004*. 137–150.

[13] Alina Ene, Sungjin Im, and Benjamin Moseley. 2011. Fast clustering using MapReduce. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. 681–689.

[14] Sainyam Galhotra. 2021. Robust Algorithms for Clustering with Applications to Data Integration. (2021).

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

[16] Aniko Hannak, Gary Soeller, David Lazer, Alan Mislove, and Christo Wilson. 2014. Measuring Price Discrimination and Steering on E-Commerce Web Sites. In *Proceedings of the 2014 Conference on Internet Measurement Conference (IMC '14)*. Association for Computing Machinery, New York, NY, USA, 305–318.

[17] Anikó Hannák, Claudia Wagner, David Garcia, Alan Mislove, Markus Strohmaier, and Christo Wilson. 2017. Bias in Online Freelance Marketplaces: Evidence from TaskRabbit and Fiverr. In *Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing*. Association for Computing Machinery, New York, NY, USA, 1914–1933.

[18] Elfarouk Harb and Ho Shan Lam. 2020. KFC: A Scalable Approximation Algorithm for k-center Fair Clustering. In *Advances in Neural Information Processing Systems*. 14509–14519.

[19] Lingxiao Huang, Shaofeng Jiang, and Nisheeth Vishnoi. 2019. Coresets for clustering with fairness constraints. In *Proc. 33rd Conference on Neural Information Processing Systems*. 7587–7598.

[20] IBM. 2019. IBM ILOG CPLEX 12.9. (2019).

[21] Sungjin Im and Benjamin Moseley. 2015. Fast and better distributed mapreduce algorithms for k-center clustering. In *Proceedings of the 27th ACM symposium on Parallelism in Algorithms and Architectures*. 65–67.

[22] Srinivasa KG, K Venugopal, and L Patnaik. 2006. Feature extraction using fuzzy c-means clustering for data mining systems. *IJCSNS* 6, 3A (2006), 230.

[23] Bjornar Larsen and Chinatsu Aone. 1999. Fast and effective text mining using linear-time document clustering. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*. 16–22.

[24] Gustavo Malkomes, Matt J Kusner, Wenlin Chen, Kilian Q Weinberger, and Benjamin Moseley. 2015. Fast distributed k-center clustering with outliers on massive data. *Advances in Neural Information Processing Systems* 28 (2015), 1063–1071.

[25] Richard Matthew McCutchen and Samir Khuller. 2008. Streaming algorithms for k-center clustering with outliers and with anonymity. In *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques*. Springer, 165–178.

[26] Clemens Rösner and Melanie Schmidt. 2018. Privacy Preserving Clustering with Constraints. In *Proc. 45th International Colloquium on Automata, Languages and Programming*. 96:1–96:14.

[27] Badrul M Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2002. Recommender systems for large-scale e-commerce: Scalable neighborhood formation using clustering. In *Proceedings of the fifth international conference on computer and information technology*, Vol. 1. 291–324.

[28] David B. Shmoys and Éva Tardos. 1993. An approximation algorithm for the generalized assignment problem. *Mathematical programming* 62, 1-3 (1993), 461–474.

[29] Till Speicher, Muhammad Ali, Giridhari Venkatadri, Filipe Nunes Ribeiro, George Arvanitakis, Fabrício Benevenuto, Krishna P Gummadi, Patrick Loiseau, and Alan Mislove. 2018. Potential for Discrimination in Online Targeted Advertising. In *Conference on Fairness, Accountability and Transparency*. 5–19.