F^{3}KM: Federated, Fair, and Fast k-means

SHENGKUN ZHU, School of Computer Science, Wuhan University, China
QUANQING XU, OceanBase, Ant Group, China
JINSHAN ZENG, School of Computer and Information Engineering, Jiangxi Normal University, China
SHENG WANG*, School of Computer Science, Wuhan University, China
YUAN SUN, La Trobe Business School, La Trobe University, Australia
ZHIFENG YANG, OceanBase, Ant Group, China
CHUANHUI YANG, OceanBase, Ant Group, China
ZHIYONG PENG, School of Computer Science & Big Data Institute, Wuhan University, China

This paper proposes a federated, fair, and fast k-means algorithm (F^{3}KM) to solve the fair clustering problem efficiently in scenarios where data cannot be shared among different parties. The proposed algorithm decomposes the fair k-means problem into multiple subproblems and assigns each subproblem to a client for local computation. Our algorithm allows each client to possess multiple sensitive attributes (or have no sensitive attributes). We propose an in-processing method that employs the alternating direction method of multipliers (ADMM) to solve each subproblem. During the procedure of solving subproblems, only the computation results are exchanged between the server and the clients, without exchanging the raw data. Our theoretical analysis shows that F^{3}KM is efficient in terms of both communication and computation complexities. Specifically, it achieves a better trade-off between utility and communication complexity, and reduces the computation complexity to linear with respect to the dataset size. Our experiments show that F^{3}KM achieves a better trade-off between utility and fairness than other methods. Moreover, F^{3}KM is able to cluster five million points in one hour, highlighting its impressive efficiency.

CCS Concepts: • **Computing methodologies → Cluster analysis**.

Additional Key Words and Phrases: Federated, Fair, Fast, k-means, ADMM

ACM Reference Format:
Shengkun Zhu, Quanqing Xu, Jinshan Zeng, Sheng Wang, Yuan Sun, Zhifeng Yang, Chuanhui Yang, and Zhiyong Peng. 2023. F^{3}KM: Federated, Fair, and Fast k-means. *Proc. ACM Manag. Data* 1, 4 (SIGMOD), Article 241 (December 2023), 25 pages. https://doi.org/10.1145/3626728

1 INTRODUCTION

The rise of big data has emphasized the crucial role of clustering analysis in data science. This statistical technique facilitates the exploration of a dataset's internal structure by organizing data

*Corresponding author

Authors' addresses: Shengkun Zhu, whuzsk66@whu.edu.cn, School of Computer Science, Wuhan University, China; Quanqing Xu, OceanBase, Ant Group, China, xuquanqing.xqq@oceanbase.com; Jinshan Zeng, School of Computer and Information Engineering, Jiangxi Normal University, China, jinshanzeng@jxnu.edu.cn; Sheng Wang, School of Computer Science, Wuhan University, China, swangcs@whu.edu.cn; Yuan Sun, La Trobe Business School, La Trobe University, Australia, yuan.sun@latrobe.edu.au; Zhifeng Yang, OceanBase, Ant Group, China, zhuweng.yzf@oceanbase.com; Chuanhui Yang, OceanBase, Ant Group, China, rizhao.ych@oceanbase.com; Zhiyong Peng, School of Computer Science & Big Data Institute, Wuhan University, China, peng@whu.edu.cn.

2836-6573/2023/12-ART241 $15.00
https://doi.org/10.1145/3626728

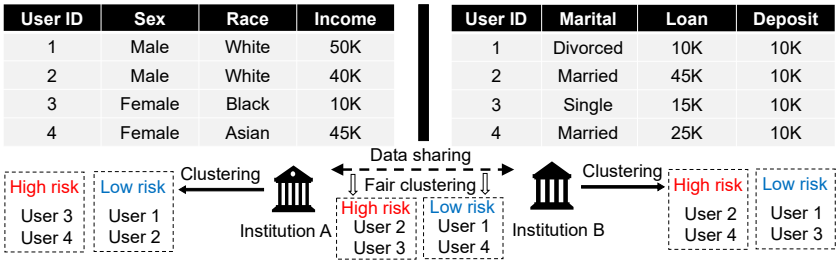

Fig. 1. When data sharing is prohibited and fairness is disregarded, institutions A and B obtain inaccurate and biased results. However, enabling data sharing and implementing fair clustering can lead to accurate and unbiased results.

points into clusters [5]. As one of the most classical clustering algorithms, k-means aims to partition data points into k clusters such that the points within a cluster are as close as possible [49]. k-means has found extensive applications in various domains, including healthcare [26, 32, 42, 45], finance [14, 20, 37, 55], education [29, 43], etc. Its usage has facilitated data understanding, improved decision-making efficiency, and provided high-quality services.

However, applying k-means in these domains may pose some issues due to legal mandates [21, 28]. Let us consider the following scenario in Figure 1: two financial institutions, A and B, seek to conduct collaborative k-means clustering analysis on a set of customers by using the credit card transaction data to identify distinct credit risk analysis. Due to the sensitivity of financial data, such as income and transaction records, each institution is restricted to accessing its own portion of customer transaction data solely, thereby preventing the direct sharing of raw data. Furthermore, both institutions aim to ensure that the clustering results adhere to the principle of fairness, whereby the proportion of protected groups, comprising demographics such as gender and race, is approximately equal across clusters [15]. However, due to data unavailability for sharing and insufficient attention to fairness, institutions A and B cluster the same users but obtain different results with unfairness. Specifically, institution A clusters low-income black and Asian females together, while institution B clusters individuals who are married with high loans. If financial institutions are able to obtain additional user features while ensuring fairness, they can improve the accuracy of credit risk assessment and enhance customer satisfaction [6, 33].

Vertical federated learning (VFL) is a promising framework for addressing the issue of non-sharable data. It is a distributed machine learning approach that divides features among clients so that each client possesses an independent set of features while sharing the same set of users [23, 34, 35, 39, 51, 52]. VFL enables the training and updating of models using data from multiple clients without exposing raw data. However, current VFL frameworks for clustering face communication bottlenecks [31]. Specifically, the communication complexity of existing frameworks is highly contingent on the dataset size. Ding et al. [19] developed a constant approximation scheme for k-means clustering, which exhibit linear communication complexity with respect to the dataset size. Huang et al. [31] proposed a technique for reducing communication complexity by constructing a coreset, which results in sublinear complexity with respect to the dataset size. However, such communication complexity remains intolerable in the era of big data.

Fairness in federated k-means is also a focal point of our concern. Specifically, we are concerned with *group fairness*, which is a notion that emerged from the disparate impact doctrine [22] and was initially introduced by Chierichetti et al. [15]. In the context of clustering, group fairness can be characterized by the proportional representation of protected groups in each cluster. More

precisely, the outcome of clustering should ensure that the proportion of protected groups in each cluster is approximately equivalent. For example, in Figure 1, the clustering results of institution A either consist entirely of females or males, which is evidently unfair. Current methods for ensuring fairness in k-means clustering primarily rely on pre-processing [9, 15] or post-processing [11, 30] techniques. Specifically, Chierichetti et al. [15] proposed a pre-processing approach that involved transforming the data into a specific format called *fairlet*, followed by conducting clustering on the *fairlet*. On the other hand, Bera et al. [11] proposed a post-processing method that employed linear programming (LP) to ensure fairness after performing the clustering. However, these methods cannot control the optimization objective directly, which limits the applicability of these methods in the context of VFL. Furthermore, adjustment of the data or result may cause legal implications and an inexplicable model, which might be inconsistent with the data protection rules with regard to interpretability [40]. Moreover, the state-of-the-art method for solving fair k-means problem entails solving a LP problem consisting of nk variables [11], where n denotes the number of data points and k denoted the number of clusters. However, when n surpasses the million-level threshold, this method struggles to solve the fair k-means problem rapidly within a truncated period.

To solve the fair k-means problem in the VFL scenario efficiently, we propose a novel federated, fair, and fast k-means algorithm, called F^{3}KM. Our contributions can be summarized as follows:

- We propose a novel approach to address the fair k-means problem within the VFL framework, which effectively mitigates the concern of compromising sensitive data. Specifically, we decompose the fair k-means problem into multiple subproblems that can be efficiently solved on the client-side. By doing so, the client is only required to transmit the computed results instead of the raw data to the central server (Section 4.1).
- We propose an in-processing method for solving the fair k-means problem. Specifically, we transform the fair k-means problem into a single-variable optimization problem with multiple inequality constraints and employ *alternating direction method of multipliers* (ADMM) to solve it. Our method ensures fairness for multiple protected groups by making modifications solely to the clustering model during the solving process, thereby avoiding direct adjustments to the data or results (Section 4.2).
- Our theoretical analysis shows that the proposed F^{3}KM algorithm, which leverages ADMM to iteratively solve the fair k-means problem for both primal and dual variables, achieves linear computational complexity with respect to the dataset size n. This makes our method more efficient than the state-of-the-art method that requires solving the LP with nk variables. Furthermore, we propose a *block coordinate descent* (BCD) method to optimize primal variables, achieving a better trade-off between utility and communication complexity. Specifically, it enables a linear speedup of $1/n_b$ in communication complexity, where n_b denotes the number of points in each block (Section 4.3).
- We demonstrate the effectiveness and efficiency of F^{3}KM over ten real-world datasets in our experimental evaluations. Our experimental findings indicate that F^{3}KM outperforms the state-of-the-art method regarding the trade-off betweem utility and fairness. In addition, F^{3}KM can complete fair k-means on a dataset comprising 5 million points within one hour, whereas the state-of-the-art method is only capable of achieving fair k-means on a dataset containing 0.5 million points (Section 5).

2 RELATED WORK

We review relevant literature across three domains: federated clustering, fair clustering, and fast clustering.

Federated clustering. Most existing studies for federated clustering have been conducted in the context of horizontal federated learning (HFL) [8, 10, 18, 25, 44]. Among these studies, the

algorithm proposed by Dennis et al. [18] for heterogeneous networks achieved a one-shot communication complexity, making it one of the most advanced federated clustering algorithms. The existing research on vertical federated clustering is limited [19, 31, 36]. Ding et al. [19] proposed an approximation algorithm for the distributed dimensions scenarios of k-means clustering. This algorithm is executed by computing the global centers, which are derived from the product of local centers. Huang et al. [31] conducted research on communication-efficient approaches for vertical federated clustering, with a particular focus on scalability. They developed a comprehensive paradigm focused on the coreset. Li et al. [36] introduced a method to ensure differential privacy in vertical federated clustering with multiple clients and an untrusted server by having each data party generate a differentially private data synopsis.

Fair clustering. In recent years, various approximation algorithms have been proposed for fair clustering. One significant contribution in this area is the work by Chierichetti et al. [15], which extended the concept of disparate impact to clustering problems. To address the problem of fair clustering when there are only two groups (e.g., males or females), they proposed the concept of *fairlets*. Several studies have aimed to extend the this notion since its inception [4, 11, 12, 24, 27, 30, 54]. As Chierichetti et al. [15] focused solely on two protected groups, Bera et al. [11] introduced a more general concept that applies to any protected group in each cluster, making it one of the most progressive notions of fairness. In addition, Bera et al. [11] proposed a post-processing method for fair clustering by solving LP problems. Their method enables the conversion of vanilla clustering results into fair patterns. Subsequently, several studies [12, 27, 30, 46] have extended the framework proposed by Bera et al. [11].

Fast clustering. In the context of VFL, fast clustering usually entails the consideration of reducing both communication and computational complexities. For communication complexity, the method proposed by Ding et al. [19] provides a communication complexity of $O(nT)$. Huang et al. [31] provided a coreset construction method whose communication complexity is $O(n)$. Li et al. [36] introduced a differentially private framework for vertical federated clustering, which achieves a communication complexity that is independent of n. Notably, this algorithm has the lowest observed communication complexity among existing approaches. Regarding the computational complexity, the state-of-the-art method for fair k-means is the Fair-LP proposed by Bera et al. [11], which involves solving the LP with nk variables. Harb and Lam [27] improved the efficiency of fair k-center objective by reducing the number of LP variables. Bera et al. [12] proposed to use a MapReduce framework to speed up the fair k-center. Along with [27], these approaches are unsuitable for the k-means objective. Meanwhile, Huang et al. [30] proposed a coreset for fair clustering to improve computational efficiency. However, this technique impacts both the clustering utility and the fairness of the resultant clusters. Nie et al. [41] introduced a coordinate descent method for k-means that exhibits equivalent computational complexity to that of the well-known Lloyd's heuristic, yet possesses the potential to converge to better local minima. This paper does not delve into alternative techniques that facilitate the acceleration of k-means, such as those based on indexing or hardware. Interested readers may refer to the review provided by Wang et al. [47] for a comprehensive treatment of these methods.

Remarks. 1) Existing fair k-means implementations rely on pre-processing and post-processing, potentially leading to legal implications and an inexplicable model; 2) Existing k-means works rely on Lloyd's heuristic, but it often converges to a low-quality local minimum, resulting in suboptimal clustering utility; 3) Current VFL framework for k-means involves cumbersome computation with a weighted grid based on Cartesian product of clients' local centers; 4) Existing federated and fair clustering research faces efficiency issues in communication and computation complexity. When

dealing with large datasets, such as those containing millions of points, existing methods cannot provide a satisfactory solution efficiently.

3 PRELIMINARIES

As is well known, the k-means problem is a bivariate optimization problem, and Lloyd's heuristic solves the k-means problem by assigning each point to its nearest center and refining the center iteratively. Recently, Nie et al. [41] proposed to transform the bivariate optimization problem of k-means into a univariate one, where only the assignment of sample points needs to be considered, without solving for the centers. They also introduced a coordinate descent method for solving k-means (CDKM). Here, we provide the specific form of the k-means without centers and the solving process of CDKM. Finally, we elaborate on the notion of fairness proposed by Bera et al. [11] and transfer it into a matrix multiplication form.

3.1 k-means without Centers

Nie et al. [41] proposed a *coordinate descent* (CD) method for k-means clustering. They converted the objective function of k-means problem into a trace minimization problem where the optimization variable is an indicator matrix $\boldsymbol{F}$. Note that the centers are eliminated in this objective function, and the indicator matrix $\boldsymbol{F}$ is the unique optimization variable. Then they solve $\boldsymbol{F}$ by CD from the first row to the last row. Now we first give a brief introduction of their problem formulations. Let $\boldsymbol{X} \in \mathbb{R}^{d\times n}$ denote a matrix of n data points, each column of $\boldsymbol{X}$ represents a data point denoted as $\boldsymbol{x}_i \in \mathbb{R}^d$. k-means clustering problem aims to find a set $\mathcal{Z} = \{Z_1, Z_2, \cdots, Z_k\}$ of k clusters such that the sum of squared error is minimized,

$$\min_{Z} \sum_{j=1}^{k} \sum_{\boldsymbol{x}_i \in Z_j} ||\boldsymbol{x}_i - \boldsymbol{c}_j||_2^2, \tag{1}$$

where $\boldsymbol{c}_j = \frac{1}{|Z_j|} \sum_{\boldsymbol{x}_i \in Z_j} \boldsymbol{x}_i$ is the j-th column of center matrix $\boldsymbol{C} \in \mathbb{R}^{d\times k}$. The problem (1) is equivalent to

$$\min_{\boldsymbol{F}\in Ind, \boldsymbol{C}} \sum_{i=1}^{n} \sum_{j=1}^{k} ||\boldsymbol{x}_i - \boldsymbol{c}_j||_2^2 f_{ij} = \min_{\boldsymbol{F}\in Ind, \boldsymbol{C}} ||\boldsymbol{X} - \boldsymbol{C}\boldsymbol{F}^{\mathrm{T}}||_{\mathrm{F}}^2, \tag{2}$$

where $||\cdot||_{\mathrm{F}}$ denotes *Frobenius norm* defined as the square root of the sum of the squared values of all elements in the matrix, and $\boldsymbol{F} \in \mathbb{R}^{n\times k}$ is an indicator matrix composed of n one-hot vectors. For example, if $\boldsymbol{x}_i \in Z_j$, then $f_{ij} = 1$, other elements in the i-th row of $\boldsymbol{F}$ are zero. By taking the derivative with respect to $\boldsymbol{C}$ and setting it to zero, the equation $\boldsymbol{C} = \boldsymbol{X}\boldsymbol{F}(\boldsymbol{F}^{\mathrm{T}}\boldsymbol{F})^{-1}$ is obtained. Substituting this equation into (2), we can reformulate (2) to a trace minimization problem,

$$\min_{\boldsymbol{F}\in Ind} \phi(\boldsymbol{F}) = \min_{\boldsymbol{F}\in Ind} -Tr((\boldsymbol{F}^{\mathrm{T}}\boldsymbol{F})^{-1}\boldsymbol{F}^{\mathrm{T}}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}\boldsymbol{F}). \tag{3}$$

Since $\boldsymbol{F}^{\mathrm{T}}\boldsymbol{F}$ is a diagonal matrix with (j, j) element equaling to $\boldsymbol{f}_j^{\mathrm{T}}\boldsymbol{f}_j$, where $\boldsymbol{f}_j$ is the j-th column of $\boldsymbol{F}$, the optimization problem (3) can be rewritten as follows:

$$\min_{\boldsymbol{F}\in Ind} \phi(\boldsymbol{F}) = \min_{\boldsymbol{F}\in Ind} -\sum_{j=1}^{k} \frac{\boldsymbol{f}_j^{\mathrm{T}}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}\boldsymbol{f}_j}{\boldsymbol{f}_j^{\mathrm{T}}\boldsymbol{f}_j}. \tag{4}$$

Note that (4) is a nonconvex optimization problem whose optimization variable is an indicator matrix. Conventional gradient-based optimization methods, such as stochastic gradient descent (SGD), cannot be applied to this optimization problem due to its non-differentiability. To address this issue, Nie et al. [41] proposed a CD method to solve problem (4). Next, we review this method.

Table 1. Summary of notations

Notation	Description
$\boldsymbol{X} \in \mathbb{R}^{d\times n}$	The dataset
$\mathcal{Z} = \{Z_1, Z_2, \cdots, Z_k\}$	The set of k clusters
$\boldsymbol{C} \in \mathbb{R}^{d\times k}$	The center matrix
$\boldsymbol{F} \in \mathbb{R}^{n\times k}$	The indicator matrix
$\boldsymbol{P} \in \mathbb{R}^{n\times L}$	The color matrix
$\{G_1, G_2, \cdots, G_L\}$	The set of protected groups
$\boldsymbol{\Theta}_h \in \mathbb{R}^{L_h\times k}$	The auxiliary variable on each client side
$\boldsymbol{U}_h \in \mathbb{R}^{L_h\times k}$	The dual variable on each client side
$\boldsymbol{\alpha}, \boldsymbol{\beta} \in \mathbb{R}^{L\times k}$	The upper and lower bounds in RD and MP

3.2 Description of CDKM

Nie et al. [41] considered a CD method to update each row of $\boldsymbol{F}$ by decomposing (4) into n subproblems. Let $\{\boldsymbol{F}_i^{(1)}, \boldsymbol{F}_i^{(2)}, \cdots, \boldsymbol{F}_i^{(k)}\}$ be a set of k matrices when updating the i-th row of $\boldsymbol{F}$, the only difference between $\boldsymbol{F}_i^{(m)}$ and $\boldsymbol{F}_i^{(l)}$ is the (i,m)-element and the (i,l)-element, $m \neq l$ and $m, l \in [k]$, where $[k]$ is short for $\{1, 2, \cdots, k\}$. The (i,m)-element of $\boldsymbol{F}_i^{(m)}$ is 1, the rest elements are 0, while the (i,l)-element of $\boldsymbol{F}_i^{(l)}$ is 1, the rest elements are 0. For simplicity, we denote j-th column of $\boldsymbol{F}_i^{(m)}$ by $\boldsymbol{f}_j^{(m)}$ where we omit the subscript i. Then the subproblems are given as follows:

$$\min_{m\in[k]} \phi(\boldsymbol{F}_i^{(m)}) = \min_{m\in[k]} -\sum_{j=1}^{k} \frac{(\boldsymbol{f}_j^{(m)})^{\mathrm{T}} \boldsymbol{X}^{\mathrm{T}} \boldsymbol{X} \boldsymbol{f}_j^{(m)}}{(\boldsymbol{f}_j^{(m)})^{\mathrm{T}} \boldsymbol{f}_j^{(m)}}. \tag{5}$$

To reduce the complexity and simplify the objective function, an auxiliary variable $\boldsymbol{F}_i^{(0)} \in \mathbb{R}^{n\times k}$ is introduced. Elements in the i-th row of $\boldsymbol{F}_i^{(0)}$ are zero, while the other elements are same with $\boldsymbol{F}_i^{(m)}$. Next, we present an equivalent formulation for problem (5):

$$\begin{aligned}
\min_{m\in[k]} \varphi(i,m) &= \min_{m\in[k]} \phi(\boldsymbol{F}_i^{(m)}) - \phi(\boldsymbol{F}_i^{(0)}) \\
&= \min_{m\in[k]} -\sum_{j=1}^{k} \frac{(\boldsymbol{f}_j^{(m)})^{\mathrm{T}} \boldsymbol{X}^{\mathrm{T}} \boldsymbol{X} \boldsymbol{f}_j^{(m)}}{(\boldsymbol{f}_j^{(m)})^{\mathrm{T}} \boldsymbol{f}_j^{(m)}} + \sum_{j=1}^{k} \frac{(\boldsymbol{f}_j^{(0)})^{\mathrm{T}} \boldsymbol{X}^{\mathrm{T}} \boldsymbol{X} \boldsymbol{f}_j^{(0)}}{(\boldsymbol{f}_j^{(0)})^{\mathrm{T}} \boldsymbol{f}_j^{(0)}} \\
&= \min_{m\in[k]} \frac{(\boldsymbol{f}_m^{(0)})^{\mathrm{T}} \boldsymbol{X}^{\mathrm{T}} \boldsymbol{X} \boldsymbol{f}_m^{(0)}}{(\boldsymbol{f}_m^{(0)})^{\mathrm{T}} \boldsymbol{f}_m^{(0)}} - \frac{(\boldsymbol{f}_m^{(m)})^{\mathrm{T}} \boldsymbol{X}^{\mathrm{T}} \boldsymbol{X} \boldsymbol{f}_m^{(m)}}{(\boldsymbol{f}_m^{(m)})^{\mathrm{T}} \boldsymbol{f}_m^{(m)}}.
\end{aligned} \tag{6}$$

Note that the problem (6) has only two terms, while the problem (5) has k terms. However, both optimization problems are equivalent, and this equivalence can greatly reduce the complexity. When updating the i-th row of $\boldsymbol{F}$, $k+1$ variables, namely $\boldsymbol{F}_i^{(0)}, \cdots, \boldsymbol{F}_i^{(k)}$, must be stored in memory. To optimize memory usage, Nie et al. [41] propose the replacement of $\boldsymbol{F}_i^{(m)}, m \in [k]$, with $\boldsymbol{F}$ - the current indicator matrix pre-stored in memory. The index of element 1 in the i-th row of $\boldsymbol{F}$ is stored as $r, r \in [k]$. Denoting $\boldsymbol{f}_m^{(m)}$ as the m-th column of $\boldsymbol{F}_i^{(m)}$, $\boldsymbol{f}_m^{(0)}$ as the m-th column of $\boldsymbol{F}_i^{(0)}$, and $\boldsymbol{f}_m$ as the m-th column of $\boldsymbol{F}$, two situations arise for (6):

- **Situation 1**: $m = r$, which means the i-th element of $\boldsymbol{f}_m^{(m)}$ and $\boldsymbol{f}_m$ is 1. Then we have $\boldsymbol{f}_m^{(m)} = \boldsymbol{f}_m$. Let $\boldsymbol{\delta}_m = \boldsymbol{f}_m - \boldsymbol{f}_m^{(0)}$, that is the i-th element of $\boldsymbol{\delta}_m$ is 1, the other elements are 0. Then we have

the following equations:

$$\begin{aligned} \boldsymbol{X}\boldsymbol{f}_m &= \boldsymbol{X}(\boldsymbol{f}_m^{(0)} + \boldsymbol{\delta}_m) = \boldsymbol{X}\boldsymbol{f}_m^{(0)} + \boldsymbol{x}_i, \\ (\boldsymbol{f}_m^{(0)})^{\mathrm{T}}\boldsymbol{f}_m^{(0)} &= (\boldsymbol{f}_m - \boldsymbol{\delta}_m)^{\mathrm{T}}(\boldsymbol{f}_m - \boldsymbol{\delta}_m) = \boldsymbol{f}_m^{\mathrm{T}}\boldsymbol{f}_m - 1. \end{aligned}$$

Substitute the above equations into (6), we have:

$$\varphi(i,m) = \frac{\boldsymbol{f}_m^{\mathrm{T}}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}\boldsymbol{f}_m - 2\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{X}\boldsymbol{f}_m + \boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{x}_i}{\boldsymbol{f}_m^{\mathrm{T}}\boldsymbol{f}_m - 1} - \frac{\boldsymbol{f}_m^{\mathrm{T}}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}\boldsymbol{f}_m}{\boldsymbol{f}_m^{\mathrm{T}}\boldsymbol{f}_m}. \tag{7}$$

- **Situation 2**: $m \neq r$, which means the i-th element of $\boldsymbol{f}_m^{(0)}$ and $\boldsymbol{f}_m$ is 0. Then we have $\boldsymbol{f}_m^{(0)} = \boldsymbol{f}_m$. Let $\boldsymbol{\delta}_m = \boldsymbol{f}_m^{(m)} - \boldsymbol{f}_m$, that is the i-th element of $\boldsymbol{\delta}_m$ is 1, the other elements are 0. Then we have the following equations:

$$\begin{aligned} \boldsymbol{X}\boldsymbol{f}_m^{(m)} &= \boldsymbol{X}(\boldsymbol{f}_m + \boldsymbol{\delta}_m) = \boldsymbol{X}\boldsymbol{f}_m + \boldsymbol{x}_i, \\ (\boldsymbol{f}_m^{(m)})^{\mathrm{T}}\boldsymbol{f}_m^{(m)} &= (\boldsymbol{f}_m + \boldsymbol{\delta}_m)^{\mathrm{T}}(\boldsymbol{f}_m + \boldsymbol{\delta}_m) = \boldsymbol{f}_m^{\mathrm{T}}\boldsymbol{f}_m + 1. \end{aligned}$$

Substitute the above equations into (6), we have:

$$\varphi(i,m) = \frac{\boldsymbol{f}_m^{\mathrm{T}}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}\boldsymbol{f}_m}{\boldsymbol{f}_m^{\mathrm{T}}\boldsymbol{f}_m} - \frac{\boldsymbol{f}_m^{\mathrm{T}}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}\boldsymbol{f}_m + 2\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{X}\boldsymbol{f}_m + \boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{x}_i}{\boldsymbol{f}_m^{\mathrm{T}}\boldsymbol{f}_m + 1}. \tag{8}$$

Combining the above two situations, we rewrite the objective function as follows:

$$\varphi(i,m) = \begin{cases} \frac{\boldsymbol{f}_m^{\mathrm{T}}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}\boldsymbol{f}_m - 2\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{X}\boldsymbol{f}_m + \boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{x}_i}{\boldsymbol{f}_m^{\mathrm{T}}\boldsymbol{f}_m - 1} - \frac{\boldsymbol{f}_m^{\mathrm{T}}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}\boldsymbol{f}_m}{\boldsymbol{f}_m^{\mathrm{T}}\boldsymbol{f}_m}, & m = r, \\ \frac{\boldsymbol{f}_m^{\mathrm{T}}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}\boldsymbol{f}_m}{\boldsymbol{f}_m^{\mathrm{T}}\boldsymbol{f}_m} - \frac{\boldsymbol{f}_m^{\mathrm{T}}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}\boldsymbol{f}_m + 2\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{X}\boldsymbol{f}_m + \boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{x}_i}{\boldsymbol{f}_m^{\mathrm{T}}\boldsymbol{f}_m + 1}, & m \neq r. \end{cases} \tag{9}$$

Based on (9) and the nature of CD, the updating function of the i-th row of $\boldsymbol{F}$ is given as follows:

$$f_{is} = \begin{cases} 1, & s = \arg\min_m \varphi(i,m), \\ 0, & \text{otherwise}, \end{cases} \tag{10}$$

where s represents the updated index of $\boldsymbol{F}$, f_{is} is the (i,s) element in $\boldsymbol{F}$. To accelerate convergence, Nie et al. [41] propose an incremental algorithm, also referred to as incremental updating [17]. It reduces computational costs by computing only the changed terms, leading to faster results generation and reduced storage space requirements compared to recomputing the entire terms. Consider the objective function defined in equation (9), which contains four terms: $\boldsymbol{f}_m^{\mathrm{T}}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}\boldsymbol{f}_m$, $\boldsymbol{X}\boldsymbol{f}_m$, $\boldsymbol{f}_m^{\mathrm{T}}\boldsymbol{f}_m$, and $\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{x}_i$. The value of $\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{x}_i$ can be precomputed and stored in memory. For updating the other three terms, two scenarios arise: when $r = s$, the indicator matrix $\boldsymbol{F}$ remains unchanged, and consequently, the three terms do not require updating. However, when $r \neq s$, the (i,r) and (i,s) elements of the indicator matrix $\boldsymbol{F}$ should be updated, that is, $\boldsymbol{f}_{ir}$ and $\boldsymbol{f}_{is}$ should be exchanged. We present the updated formulas as follows:

$$\begin{aligned} \boldsymbol{X}\boldsymbol{f}_r \leftarrow \boldsymbol{X}\boldsymbol{f}_r - \boldsymbol{x}_i,\ \boldsymbol{X}\boldsymbol{f}_s \leftarrow \boldsymbol{X}\boldsymbol{f}_s + \boldsymbol{x}_i, \\ \boldsymbol{f}_r^{\mathrm{T}}\boldsymbol{f}_r \leftarrow \boldsymbol{f}_r^{\mathrm{T}}\boldsymbol{f}_r - 1,\ \boldsymbol{f}_s^{\mathrm{T}}\boldsymbol{f}_s \leftarrow \boldsymbol{f}_s^{\mathrm{T}}\boldsymbol{f}_s + 1, \\ \boldsymbol{f}_r^{\mathrm{T}}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}\boldsymbol{f}_r \leftarrow \boldsymbol{f}_r^{\mathrm{T}}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}\boldsymbol{f}_r - 2\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{X}\boldsymbol{f}_r + \boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{x}_i, \\ \boldsymbol{f}_s^{\mathrm{T}}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}\boldsymbol{f}_s \leftarrow \boldsymbol{f}_s^{\mathrm{T}}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}\boldsymbol{f}_s + 2\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{X}\boldsymbol{f}_s + \boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{x}_i. \end{aligned} \tag{11}$$

As shown in Algorithm 1, CDKM solves $\boldsymbol{F}$ by computing the objective values and updating the parameters (Lines 5-7) iteratively until converge. Next, we review the notion of fairness.

Algorithm 1: Coordinate descent for k-means

Input: X: the dataset, k: # of clusters, n: # of points.
Output: F: the indicator matrix.

Initialize F by k-means++;
Compute and store Xf_m, $f_m^{\mathrm{T}} f_m$, $f_m^{\mathrm{T}} X^{\mathrm{T}} X f_m$, and $x_i^{\mathrm{T}} x_i$, $m \in [k], i \in [n]$;
while *not converge* **do**
  for $i = 1$ *to* n **do**
    Compute $\varphi(i, m)$, $m \in [k]$ by (9);
    Update the i-th row of F by (10);
    Update Xf_m, $f_m^{\mathrm{T}} f_m$, and $f_m^{\mathrm{T}} X^{\mathrm{T}} X f_m$ by (11);
return F;

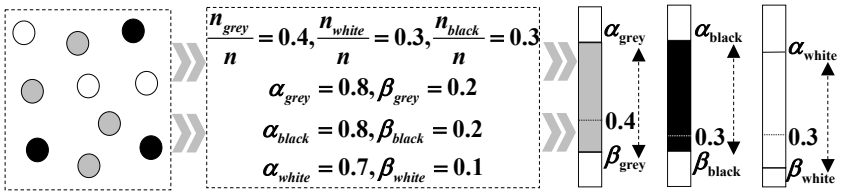

Fig. 2. An example of RD and MP, where white, black, and grey points represent three protected groups.

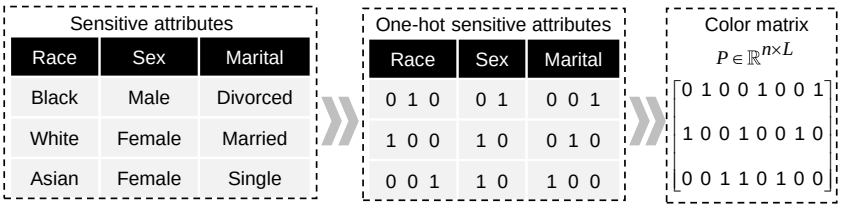

Fig. 3. An example of the color matrix P that has three sensitive attributes: Race, Sex, and Marital; each of them has several protected groups: Black, White, and Asian; Male and Female; Married, Single, and Divorced.

3.3 Fairness for k-means

A naive approach to achieve fair k-means is by removing sensitive attributes, known as *fairness through unawareness* [40]. However, this approach has the drawback that proxy attributes can still reflect sensitive attributes. For example, attributes like height and weight can reflect gender. Therefore, a more effective definition of fair clustering is to find an assignment that ensures all protected groups have approximately equal representation in the clusters while simultaneously minimizing the loss function. We adopt the fair definition in [11] under the disparate impact doctrine [22]. So far, this definition of fairness, which takes into account multiple sensitive attributes, has been extensively investigated within the domain of clustering [12]. Each point is additionally given L groups, namely $\{G_1, G_2, \cdots, G_L\}$. Bera et al. [11] proposed an explicit definition of fairness:

- *Restricted Dominance* (RD) requires the fraction of samples of group l in any cluster is at most α_l:

$$|\{x_i | x_i \in Z_j, x_i \in G_l\}| \leq \alpha_l \cdot |\{x_i | x_i \in Z_j\}|, j \in [k], l \in [L]. \tag{12}$$

- *Minority Protection* (MP) requires the fraction of samples of group l in any cluster is at least β_l:

$$|\{x_i | x_i \in Z_j, x_i \in G_l\}| \geq \beta_l \cdot |\{x_i | x_i \in Z_j\}|, j \in [k], l \in [L]. \tag{13}$$

Figure 2 shows an example of RD and MP. Bera et al. [11] also proposed loosely fair constraints which allow for λ-additive violation in RD and MP for any $j \in [k], l \in [L]$:

$$|\{x_i|x_i \in Z_j, x_i \in G_l\}| \leq \alpha_l \cdot |\{x_i|x_i \in Z_j\}| + \lambda,$$
$$|\{x_i|x_i \in Z_j, x_i \in G_l\}| \geq \beta_l \cdot |\{x_i|x_i \in Z_j\}| - \lambda.$$

We can rewrite the notion of fairness in terms of matrix multiplication. Let $\boldsymbol{P} \in \mathbb{R}^{n \times L}$ be a color matrix of $\boldsymbol{X}$, where each row of $\boldsymbol{P}$ is a concatenated one-hot vector. An example of $\boldsymbol{P}$, which contains three sensitive attributes and eight protected groups, is shown in Figure 3. By utilizing the color matrix, we can redefine the concepts of RD and MP as follows:

$$\beta_l \leq \frac{\boldsymbol{p}_l^{\mathrm{T}} \boldsymbol{f}_j}{\boldsymbol{f}_j^{\mathrm{T}} \boldsymbol{f}_j} \leq \alpha_l, j \in [k], l \in [L], \tag{14}$$

where $\boldsymbol{p}_l$ is the l-th column of $\boldsymbol{P}$, $\boldsymbol{p}_l^{\mathrm{T}} \boldsymbol{f}_j$ is equal to $|\{x_i|x_i \in Z_j, x_i \in G_l\}|$ and $\boldsymbol{f}_j^{\mathrm{T}} \boldsymbol{f}_j$ is equal to $|\{x_i|x_i \in Z_j\}|$.

4 PROPOSED F^{3}KM

In this section, we aim to address the following three challenges:

- How to address the issue of low iteration efficiency in CDKM, where n inner iterations need to be performed within a single outer iteration (Line 4 in Algorithm 1)?
- How to partition the fair k-means problem into multiple subproblems and enable each client to solve their respective subproblem?
- How to formulate the fair k-means problem as an optimization problem and solve it using an in-processing method?

In addressing these challenges, we propose the following strategies: 1) we introduce block coordinate descent for k-means (BCDKM) as an alternative to CDKM to improve the efficiency of iterations; 2) we propose partitioning the dataset based on the feature dimensions and dividing the k-means problem into multiple subproblems. We solve these subproblems in a distributed scenario, naming this approach DisBCDKM; 3) utilizing the techniques of DisBCDKM, we propose dividing the fair k-means problem into multiple subproblems and solving them using the ADMM.

4.1 Proposed DisBCDKM

Next, we propose a block coordinate descent method for k-means (BCDKM) by solving a block of subproblems instead of solving a subproblem, and updating a block of rows of $\boldsymbol{F}$ instead of updating a row of $\boldsymbol{F}$. Moreover, we propose a distributed version of BCDKM by partitioning the dataset from the feature dimensions. Figure 4 shows the comparison among CDKM, BCDKM, and DisBCDKM.

4.1.1 Block coordinate descent for k-means. Block coordinate descent (BCD) is an iterative algorithm commonly used for solving optimization problems [53]. Similar to CD, BCD is a variable-wise optimization method. However, BCD differs from CD in that it optimizes a block of variables concurrently instead of optimizing individual variables sequentially. Let $b = \lceil n/n_b \rceil$ denote the number of blocks, where $n_b \leq n$ represents the number of points in each block[1], we provide the details of the BCDKM in Algorithm 2. There are two key differences between the CDKM and BCDKM algorithms. Firstly, in each iteration of BCDKM, the computing step is performed by solving a block of subproblems instead of a single subproblem (Line 6). Secondly, the update step is performed by updating a block of rows in the indicator matrix $\boldsymbol{F}$ and adjusting the terms associated

[1]Here, we assume that each block contains an equal number of points, except for the last block, which may contain a different number of points. Specifically, the number of points in the last block equals $n - (b-1) \cdot n_b$.

Algorithm 2: Block coordinate descent for k-means

Input: $\boldsymbol{X}$: the dataset, k: # of clusters, n: # of points, b: # of blocks.
Output: $\boldsymbol{F}$: the indicator matrix.

Initialize $\boldsymbol{F}$ by k-means++;
Divide $\{1, 2, \cdots, n\}$ into b blocks: $\{\mathcal{B}_1, \cdots, \mathcal{B}_b\}$;
Compute and store $\boldsymbol{X}\boldsymbol{f}_m$, $\boldsymbol{f}_m^{\mathrm{T}}\boldsymbol{f}_m$, $\boldsymbol{f}_m^{\mathrm{T}}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}\boldsymbol{f}_m$ and $\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{x}_i$, $m \in [k], i \in [n]$;
while *not converge* **do**
  for $idx \leftarrow 1$ **to** b **do**
    Compute $\varphi(i, m)$, $i \in \mathcal{B}_{idx}$, $m \in [k]$ by (9);
    Update the rows within $\mathcal{B}_{idx}$ of $\boldsymbol{F}$ by (10);
    Update $\boldsymbol{X}\boldsymbol{f}_m$, $\boldsymbol{f}_m^{\mathrm{T}}\boldsymbol{f}_m$ and $\boldsymbol{f}_m^{\mathrm{T}}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}\boldsymbol{f}_m$ by (11);
return $\boldsymbol{F}$;

Algorithm 3: Distributed block coordinate descent for k-means

Input: $\{\boldsymbol{X}_h\}_{h=1}^{H}$: the partitioned dataset, k: # of clusters, n: # of points, b: # of blocks.
Output: $\boldsymbol{F}$: the indicator matrix.

Initialize $\boldsymbol{F}$ by k-means++;
Divide $\{1, 2, \cdots, n\}$ into b blocks: $\{\mathcal{B}_1, \cdots, \mathcal{B}_b\}$;
Compute and store $\boldsymbol{X}_h\boldsymbol{f}_m$, $\boldsymbol{f}_m^{\mathrm{T}}\boldsymbol{f}_m$, $\boldsymbol{f}_m^{\mathrm{T}}\boldsymbol{X}_h^{\mathrm{T}}\boldsymbol{X}_h\boldsymbol{f}_m$ and $(\boldsymbol{x}_i^h)^{\mathrm{T}}\boldsymbol{x}_i^h$, $m \in [k], i \in [n]$ in each machine;
while *not converge* **do**
  for $idx \leftarrow 1$ **to** b **do**
    for $h \in [H]$ *in parallel over machines* **do**
      Compute $\overline{\varphi}(m, i, h)$ by (17) $i \in \mathcal{B}_{idx}$, $m \in [k]$;
      Update the rows within $\mathcal{B}_{idx}$ of $\boldsymbol{F}$ by (18);
      Update $\boldsymbol{X}_h\boldsymbol{f}_m$, $\boldsymbol{f}_m^{\mathrm{T}}\boldsymbol{f}_m$, $\boldsymbol{f}_m^{\mathrm{T}}\boldsymbol{X}_h^{\mathrm{T}}\boldsymbol{X}_h\boldsymbol{f}_m$ by (19);
return $\boldsymbol{F}$;

with $\boldsymbol{F}$ (Lines 7 and 8), as opposed to updating a single row in $\boldsymbol{F}$. Specifically, when $n_b = 1$, CDKM can be considered as a specific instance of BCDKM. One of the primary advantages of BCDKM is its ability to compute a set of objective values in a single block, thus reducing the need for frequent communication during distributed implementation.

Remarks. In CDKM, each parameter is updated n times per iteration. In contrast, in BCDKM, each parameter is updated b times per iteration. Consequently, under the same number of iterations, BCDKM has a lower parameter update frequency compared to CDKM. This difference in parameter update frequency has an impact on the precision of the parameters and the convergence speed. Therefore, it is crucial to select an appropriate n_b to balance the trade-off between utility and the number of iterations.

4.1.2 Distributed implementation. We write $\{\boldsymbol{X}_h\}_{h=1}^{H}$ for the given partition of $\boldsymbol{X}$ along the feature dimension over the H worker machines. We denote the number of features in each machine by d_h. For each point $\boldsymbol{x}_i \in \mathbb{R}^d$, we write $\boldsymbol{x}_i = (\boldsymbol{x}_i^1, \cdots, \boldsymbol{x}_i^H)$, where each $\boldsymbol{x}_i^h \in \mathbb{R}^{d_h}, h \in [H]$, then we

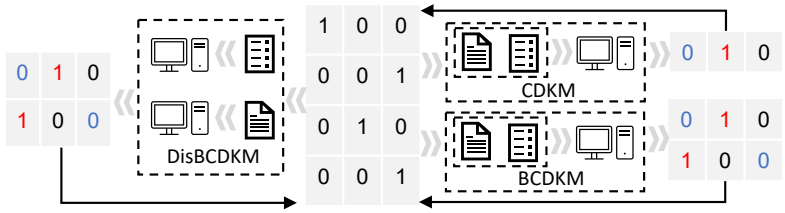

Fig. 4. BCDKM updates a block of $\boldsymbol{F}$ in each inner iteration compared to CDKM. DisBCDKM allows computation to be performed across multiple machines and yields the same results as BCDKM.

rewrite (6) as follows:

$$\begin{aligned}\min_{m\in[k]} \varphi(m,i) &= \min_{m\in[k]} \frac{(\boldsymbol{f}_m^{(0)})^{\mathrm{T}}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}\boldsymbol{f}_m^{(0)}}{(\boldsymbol{f}_m^{(0)})^{\mathrm{T}}\boldsymbol{f}_m^{(0)}} - \frac{(\boldsymbol{f}_m^{(m)})^{\mathrm{T}}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}\boldsymbol{f}_m^{(m)}}{(\boldsymbol{f}_m^{(m)})^{\mathrm{T}}\boldsymbol{f}_m^{(m)}} \\ &= \min_{m\in[k]} \sum_{h=1}^{H} \frac{(\boldsymbol{f}_m^{(0)})^{\mathrm{T}}\boldsymbol{X}_h^{\mathrm{T}}\boldsymbol{X}_h\boldsymbol{f}_m^{(0)}}{(\boldsymbol{f}_m^{(0)})^{\mathrm{T}}\boldsymbol{f}_m^{(0)}} - \frac{(\boldsymbol{f}_m^{(m)})^{\mathrm{T}}\boldsymbol{X}_h^{\mathrm{T}}\boldsymbol{X}_h\boldsymbol{f}_m^{(m)}}{(\boldsymbol{f}_m^{(m)})^{\mathrm{T}}\boldsymbol{f}_m^{(m)}}.\end{aligned} \tag{15}$$

Upon closer examination, one can easily discern the decomposability of (15) with respect to each h. As a result, we formulate a distinct subproblem for each machine:

$$\min_{m\in[k]} \overline{\varphi}(m,i,h) = \min_{m\in[k]} \frac{(\boldsymbol{f}_m^{(0)})^{\mathrm{T}}\boldsymbol{X}_h^{\mathrm{T}}\boldsymbol{X}_h\boldsymbol{f}_m^{(0)}}{(\boldsymbol{f}_m^{(0)})^{\mathrm{T}}\boldsymbol{f}_m^{(0)}} - \frac{(\boldsymbol{f}_m^{(m)})^{\mathrm{T}}\boldsymbol{X}_h^{\mathrm{T}}\boldsymbol{X}_h\boldsymbol{f}_m^{(m)}}{(\boldsymbol{f}_m^{(m)})^{\mathrm{T}}\boldsymbol{f}_m^{(m)}}. \tag{16}$$

Here, we present the objective function directly as the computational process shares similarities with the CDKM:

$$\overline{\varphi}(m,i,h) = \begin{cases} \frac{\boldsymbol{f}_m^{\mathrm{T}}\boldsymbol{X}_h^{\mathrm{T}}\boldsymbol{X}_h\boldsymbol{f}_m - 2(\boldsymbol{x}_i^h)^{\mathrm{T}}\boldsymbol{X}_h\boldsymbol{f}_m + (\boldsymbol{x}_i^h)^{\mathrm{T}}\boldsymbol{x}_i^h}{\boldsymbol{f}_m^{\mathrm{T}}\boldsymbol{f}_m - 1} - \frac{\boldsymbol{f}_m^{\mathrm{T}}\boldsymbol{X}_h^{\mathrm{T}}\boldsymbol{X}_h\boldsymbol{f}_m}{\boldsymbol{f}_m^{\mathrm{T}}\boldsymbol{f}_m}, & m = r, \\ \frac{\boldsymbol{f}_m^{\mathrm{T}}\boldsymbol{X}_h^{\mathrm{T}}\boldsymbol{X}_h\boldsymbol{f}_m}{\boldsymbol{f}_m^{\mathrm{T}}\boldsymbol{f}_m} - \frac{\boldsymbol{f}_m^{\mathrm{T}}\boldsymbol{X}_h^{\mathrm{T}}\boldsymbol{X}_h\boldsymbol{f}_m + 2(\boldsymbol{x}_i^h)^{\mathrm{T}}\boldsymbol{X}_h\boldsymbol{f}_m + (\boldsymbol{x}_i^h)^{\mathrm{T}}\boldsymbol{x}_i^h}{\boldsymbol{f}_m^{\mathrm{T}}\boldsymbol{f}_m + 1}, & m \neq r. \end{cases} \tag{17}$$

To update the i-th row of $\boldsymbol{F}$, we aggregate the objective values computed by (17) from each machine, and incorporate them into our updating formulations:

$$f_{is} = \begin{cases} 1, & s = \arg\min_m \sum_{h=1}^{H} \overline{\varphi}(m,i,h), \\ 0, & \text{otherwise.} \end{cases} \tag{18}$$

Consider the four terms presented in the objective function (17), namely $\boldsymbol{f}_m^{\mathrm{T}}\boldsymbol{f}_m$, $\boldsymbol{f}_m^{\mathrm{T}}\boldsymbol{X}_h^{\mathrm{T}}\boldsymbol{X}_h\boldsymbol{f}_m$, $\boldsymbol{X}_h\boldsymbol{f}_m$, and $(\boldsymbol{x}_i^h)^{\mathrm{T}}\boldsymbol{x}_i^h$. It is worth noting that $(\boldsymbol{x}_i^h)^{\mathrm{T}}\boldsymbol{x}_i^h$ is a constant value and does not require updating. Let r represent the current index of 1 in the i-th row of indicator matrix $\boldsymbol{F}$. If $r = s$, then $\boldsymbol{F}$ remains unchanged. However, if $r \neq s$, the following updated formulas can be directly applied, leveraging the computational similarities with CDKM:

$$\begin{aligned} \boldsymbol{f}_r^{\mathrm{T}}\boldsymbol{f}_r &\leftarrow \boldsymbol{f}_r^{\mathrm{T}}\boldsymbol{f}_r - 1;\ \boldsymbol{f}_s^{\mathrm{T}}\boldsymbol{f}_s \leftarrow \boldsymbol{f}_s^{\mathrm{T}}\boldsymbol{f}_s + 1; \\ \boldsymbol{X}_h\boldsymbol{f}_r &\leftarrow \boldsymbol{X}_h\boldsymbol{f}_r - \boldsymbol{x}_i^h;\ \boldsymbol{X}_h\boldsymbol{f}_s \leftarrow \boldsymbol{X}_h\boldsymbol{f}_s + \boldsymbol{x}_i^h; \\ \boldsymbol{f}_r^{\mathrm{T}}\boldsymbol{X}_h^{\mathrm{T}}\boldsymbol{X}_h\boldsymbol{f}_r &\leftarrow \boldsymbol{f}_r^{\mathrm{T}}\boldsymbol{X}_h^{\mathrm{T}}\boldsymbol{X}_h\boldsymbol{f}_r - 2(\boldsymbol{x}_i^h)^{\mathrm{T}}\boldsymbol{X}_h\boldsymbol{f}_r + (\boldsymbol{x}_i^h)^{\mathrm{T}}\boldsymbol{x}_i^h; \\ \boldsymbol{f}_s^{\mathrm{T}}\boldsymbol{X}_h^{\mathrm{T}}\boldsymbol{X}_h\boldsymbol{f}_s &\leftarrow \boldsymbol{f}_s^{\mathrm{T}}\boldsymbol{X}_h^{\mathrm{T}}\boldsymbol{X}_h\boldsymbol{f}_s + 2(\boldsymbol{x}_i^h)^{\mathrm{T}}\boldsymbol{X}_h\boldsymbol{f}_s + (\boldsymbol{x}_i^h)^{\mathrm{T}}\boldsymbol{x}_i^h. \end{aligned} \tag{19}$$

Algorithm 3 presents a detailed description of DisBCDKM. At each iteration, the central machine engages in $O(b)$ communications with each local machine to compute the objective values and

update the parameters (Lines 6-9). If the CDKM was employed here, the number of communications would increase to $O(n)$.

Remarks. The separability property of the optimization problem (15) ensures that our distributed scheme does not adversely affect solution accuracy, thereby enabling us to obtain results that are fully consistent with those obtained from the centralized scenario.

4.2 Description of F^{3}KM

We present F^{3}KM, a novel algorithm that enables equitable allocation of protected groups among clusters while maintaining approximately equal representation in the context of VFL. To achieve this, we first define a data-local subproblem that can be solved individually by each client, with access restricted to locally-stored data. Using an ADMM on the client side, we solve this subproblem, and the server aggregates the results from each client.

The framework of F^{3}KM is presented in Figure 5. It is noteworthy that in the context of VFL, each client possesses distinct feature data from the same users, and each user's data is stored locally. Figure 6 shows an example of the computation process of F^{3}KM on both the client and server sides. Details of F^{3}KM on the server and client sides are delineated in Algorithms 4 and 5. On the server side, $\boldsymbol{F}$ is initialized, and instructions are sent to each client to initialize their respective parameters (Line 4 in Algorithm 4). Once initialization is complete (Line 2 in Algorithm 5), the server sends instructions to each client (Line 8 in Algorithm 4), and the clients compute the objective values using their owned data and return the computed values to the server (Lines 4 and 5 in Algorithm 5). The server aggregates the objective values and subsequently updates the indicator matrix $\boldsymbol{F}$, then instructions are sent to each client to update their local parameters (Lines 9-12 in Algorithm 4 and Lines 6-11 in Algorithm 5). The iterative process is continued until convergence is achieved, following which resulting indicator matrix $\boldsymbol{F}$ is generated.

4.2.1 Local subproblem in F^{3}KM. In Section 3.3, we have presented the fair constraints in the form of matrix multiplication. Next, we integrate the fairness constraints with the objective function of k-means as follows:

$$\min_{F\in Ind} -\sum_{j=1}^{k} \frac{\boldsymbol{f}_j^{\mathrm{T}}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}\boldsymbol{f}_j}{\boldsymbol{f}_j^{\mathrm{T}}\boldsymbol{f}_j} \quad s.t.\ \beta_l \le \frac{\boldsymbol{p}_l^{\mathrm{T}}\boldsymbol{f}_j}{\boldsymbol{f}_j^{\mathrm{T}}\boldsymbol{f}_j} \le \alpha_l, j \in [k], l \in [L]. \tag{20}$$

In Section 4.1, we demonstrated that the objective function given in (20) can be decomposed over H clients. Next, we prove that the constraints can also be decomposed. To achieve this, we define a local color matrix $\boldsymbol{P}_h \in \mathbb{R}^{n\times L_h}$ with respect to $\boldsymbol{X}_h$, where $\sum_{h=1}^{H} L_h = L$. Then, we can rewrite the constraints in (20) as follows:

$$\beta_l \le \frac{(\boldsymbol{p}_l^h)^{\mathrm{T}}\boldsymbol{f}_j}{\boldsymbol{f}_j^{\mathrm{T}}\boldsymbol{f}_j} \le \alpha_l, \quad j \in [k],\ l \in [L_h],\ h \in [H], \tag{21}$$

where $\boldsymbol{p}_l^h$ is the l-th column of $\boldsymbol{P}_h$. Upon closer examination, one can easily discern the decomposability of (21) with respect to each h. As a result, we formulate a subproblem for each client:

$$\min_{F\in Ind} -\sum_{j=1}^{k} \frac{\boldsymbol{f}_j^{\mathrm{T}}\boldsymbol{X}_h^{\mathrm{T}}\boldsymbol{X}_h\boldsymbol{f}_j}{\boldsymbol{f}_j^{\mathrm{T}}\boldsymbol{f}_j} \quad \text{s.t.}\ \beta_l \le \frac{(\boldsymbol{p}_l^h)^{\mathrm{T}}\boldsymbol{f}_j}{\boldsymbol{f}_j^{\mathrm{T}}\boldsymbol{f}_j} \le \alpha_l, j \in [k], l \in [L_h]. \tag{22}$$

In the context of VFL, we assume that the data owned by each client is composed of different features from the same users [51], such that there is no overlap in the color matrix $\boldsymbol{P}_h$ for each client. For instance, if one client holds the gender feature, then other clients would not hold that

Algorithm 4: F^{3}KM on the server side

Input: n: # of points, k: # of clusters, b: # of blocks.
Output: $\boldsymbol{F}$: the indicator matrix.

Initialize $\boldsymbol{F}$ by k-means++;
Divide $[n]$ into b blocks: $\{\mathcal{B}_1, \cdots, \mathcal{B}_b\}$;
for $h \in [H]$ *in parallel over clients* **do**
    Call `Initialization` in Algorithm 5;
while *not converge* **do**
    for $idx \leftarrow 1$ **to** b **do**
        for $h \in [H]$ *in parallel over clients* **do**
            Compute $\overline{\mathcal{L}}(m, i, h), i \in \mathcal{B}_{idx}, m \in [k]$ by calling `ComputeObj` in Algorithm 5;
            Update the rows within $\mathcal{B}_{idx}$ of $\boldsymbol{F}$ by (29);
            Call `Update_F` in Algorithm 5;
    for $h \in [H]$ *in parallel over clients* **do**
        Call `UpdateParams` in Algorithm 5;
return $\boldsymbol{F}$;

Algorithm 5: F^{3}KM on the client side

Input: $\boldsymbol{X}_h$: local dataset, $\boldsymbol{P}_h$: local color matrix, $\boldsymbol{\rho}$: step size, $\boldsymbol{\alpha}, \boldsymbol{\beta}$: maximum and minimum ratio of protected groups in each cluster, $\boldsymbol{F}, \boldsymbol{\Theta}_h, \boldsymbol{U}_h$: primal and dual variables.

if *the server calls* `Initialization` **then**
    Compute and store $\boldsymbol{X}_h\boldsymbol{f}_m$, $\boldsymbol{f}_m^{\mathrm{T}}\boldsymbol{f}_m$, $\boldsymbol{f}_m^{\mathrm{T}}\boldsymbol{X}_h^{\mathrm{T}}\boldsymbol{X}_h\boldsymbol{f}_m$, $(\boldsymbol{x}_i^h)^{\mathrm{T}}\boldsymbol{x}_i^h$, $(\boldsymbol{p}_l^h)^{\mathrm{T}}\boldsymbol{f}_m$, $m \in [k], i \in [n], l \in [L_h]$ locally;
if *the server calls* `ComputeObj` **then**
    Compute $\overline{\mathcal{L}}(m, i, h), m \in [k], i \in \mathcal{B}_{idx}$ by (27) and (28);
    Upload $\overline{\mathcal{L}}(m, i, h), m \in [k], i \in \mathcal{B}_{idx}$ to the server;
else if *the server calls* `Update_F` **then**
    Update $\boldsymbol{f}_m^{\mathrm{T}}\boldsymbol{X}_h^{\mathrm{T}}\boldsymbol{X}_h\boldsymbol{f}_m$, $\boldsymbol{f}_m^{\mathrm{T}}\boldsymbol{f}_m$ and $\boldsymbol{X}_h\boldsymbol{f}_m$ by (19);
    Update $(\boldsymbol{p}_l^h)^{\mathrm{T}}\boldsymbol{f}_m$ by (30);
else if *the server calls* `UpdateParams` **then**
    Update and store $\boldsymbol{\Theta}_h$ by (31) locally;
    Update and store $\boldsymbol{U}_h$ by (32) locally;

same feature. If feature overlap occurs, each client's data needs to be preprocessed to ensure that each feature is computed only once. For example, clients can locally compute the hash value or summary of features and share it with other clients. The hash value or summary does not reveal the original feature values but enables determining their similarity. Based on the hash value or summary provided by other clients, clients can decide whether to use their own features for training.

4.2.2 ADMM as a local solver. In this section, we present an ADMM approach for the k-means with fair constraints in the VFL scenario. The use of ADMM is motivated by its great performance on non-convex and non-smooth optimization problems [48]. Specifically, for each Gauss-Seidel type minimization subproblem with primal and dual variables in the Lagrangian function, we update all variables cyclically while fixing the remaining variables at their last updated values. Firstly, We consider transforming the inequality constraints in (22) into equality constraints by introducing an

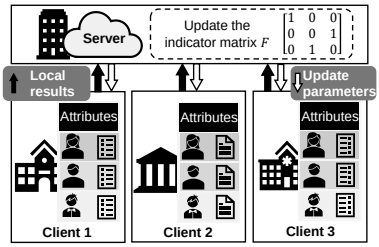

Fig. 5. General framework of F^{3}KM.

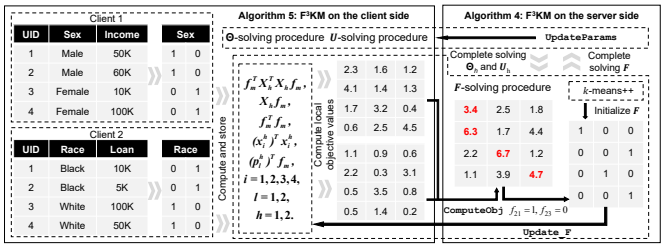

Fig. 6. An example of Algorithms 4 and 5.

auxiliary variable $\boldsymbol{\Theta}_h \in \mathbb{R}^{L_h \times k}$:

$$\begin{aligned} &\min_{\substack{\boldsymbol{F}\in Ind \\ \boldsymbol{\Theta}_h}} -\sum_{j=1}^{k} \frac{\boldsymbol{f}_j^{\mathrm{T}} \boldsymbol{X}_h^{\mathrm{T}} \boldsymbol{X}_h \boldsymbol{f}_j}{\boldsymbol{f}_j^{\mathrm{T}} \boldsymbol{f}_j} \\ &\text{s.t.} \begin{cases} \theta_{lj} = \dfrac{(\boldsymbol{p}_l^h)^{\mathrm{T}} \boldsymbol{f}_j}{\boldsymbol{f}_j^{\mathrm{T}} \boldsymbol{f}_j}, \\ \beta_l \le \theta_{lj} \le \alpha_l, \end{cases} j \in [k], l \in [L_h], \end{aligned} \tag{23}$$

where θ_{lj} is the (l, j)-element of $\boldsymbol{\Theta}_h$. Given our intention to employ the ADMM for the resolution of (23), we proceed to present the Lagrangian function of (23) as follows:

$$\mathcal{L}(\boldsymbol{F}, \boldsymbol{U}_h, \boldsymbol{\Theta}_h) = -\sum_{j=1}^{k} \frac{\boldsymbol{f}_j^{\mathrm{T}} \boldsymbol{X}_h^{\mathrm{T}} \boldsymbol{X}_h \boldsymbol{f}_j}{\boldsymbol{f}_j^{\mathrm{T}} \boldsymbol{f}_j} + \sum_{l=1}^{L_h} \sum_{j=1}^{k} u_{lj} (\theta_{lj} - \frac{(\boldsymbol{p}_l^h)^{\mathrm{T}} \boldsymbol{f}_j}{\boldsymbol{f}_j^{\mathrm{T}} \boldsymbol{f}_j}) + \sum_{l=1}^{L_h} \sum_{j=1}^{k} \frac{\rho}{2} (\theta_{lj} - \frac{(\boldsymbol{p}_l^h)^{\mathrm{T}} \boldsymbol{f}_j}{\boldsymbol{f}_j^{\mathrm{T}} \boldsymbol{f}_j})^2, \tag{24}$$

where $\Theta_h \in \boldsymbol{V}$, and $\boldsymbol{V}$ is a set of box constraints defined as $\boldsymbol{V} = \{\boldsymbol{v} \in \mathbb{R}^{L_h} | \beta_l \le v_l \le \alpha_l, l = 1, 2, \cdots, L_h\}^k$. Moreover, $\boldsymbol{U}_h \in \mathbb{R}^{L_h \times k}$ is a matrix of Lagrangian multipliers, u_{lj} denotes the (l, j)-element of $\boldsymbol{U}_h$, and ρ denotes the penalty parameter [13]. To solve (24), $\boldsymbol{F}, \boldsymbol{\Theta}_h$ and $\boldsymbol{U}_h$ should be solved recurrently until converge:

$$\begin{aligned} \boldsymbol{F}^{(t)} &= \operatorname*{argmin}_{\boldsymbol{F}} \mathcal{L}(\boldsymbol{F}, \boldsymbol{U}_h^{(t-1)}, \boldsymbol{\Theta}_h^{(t-1)}), \\ \boldsymbol{\Theta}_h^{(t)} &= \operatorname*{argmin}_{\Theta_h \in \boldsymbol{V}} \mathcal{L}(\boldsymbol{F}^{(t)}, \boldsymbol{U}_h^{(t-1)}, \boldsymbol{\Theta}_h), \\ \boldsymbol{U}_h^{(t)} &= \boldsymbol{U}_h^{(t-1)} + \rho \frac{\partial \mathcal{L}(\boldsymbol{F}^{(t)}, \boldsymbol{U}_h, \boldsymbol{\Theta}_h^{(t)})}{\partial \boldsymbol{U}_h}, \end{aligned} \tag{25}$$

where t denotes the iteration index. For sake of notion, we will write $\boldsymbol{F}$ instead of $\boldsymbol{F}^{(t)}$, $\boldsymbol{\Theta}$ instead of $\boldsymbol{\Theta}^{(t)}$ and $\boldsymbol{U}$ instead of $\boldsymbol{U}^{(t)}$. Next, we present the solving procedures for each variable.

1) $\boldsymbol{F}$-solving procedure. We consider applying DisBCDKM to solve $\boldsymbol{F}$, while treating $\boldsymbol{U}$ and $\boldsymbol{\Theta}$ as constants during the optimization process. To simplify the Lagrangian function, we adopt the technique described in equation (6), effectively reducing the number of terms in the objective function. Next, we introduce $\{\boldsymbol{F}_i^{(0)}, \boldsymbol{F}_i^{(1)}, \cdots, \boldsymbol{F}_i^{(k)}\}$ to simplify the objective function and reduce

the complexity, the description of these matrices has been given in Section 3.2,

$$\overline{\mathcal{L}}(m,i,h) = \mathcal{L}(\boldsymbol{F}_i^{(m)}, \boldsymbol{U}_h, \boldsymbol{\Theta}_h) - \mathcal{L}(\boldsymbol{F}_i^{(0)}, \boldsymbol{U}_h, \boldsymbol{\Theta}_h) = -\sum_{j=1}^{k} \frac{(\boldsymbol{f}_j^{(m)})^{\mathrm{T}} \boldsymbol{X}_h^{\mathrm{T}} \boldsymbol{X}_h \boldsymbol{f}_j^{(m)}}{(\boldsymbol{f}_j^{(m)})^{\mathrm{T}} \boldsymbol{f}_j^{(m)}} +$$
$$\sum_{j=1}^{k} \frac{(\boldsymbol{f}_j^{(0)})^{\mathrm{T}} \boldsymbol{X}_h^{\mathrm{T}} \boldsymbol{X}_h \boldsymbol{f}_j^{(0)}}{(\boldsymbol{f}_j^{(0)})^{\mathrm{T}} \boldsymbol{f}_j^{(0)}} + \sum_{l=1}^{L_h} \sum_{j=1}^{k} u_{lj} (\theta_{lj} - \frac{(\boldsymbol{p}_l^h)^{\mathrm{T}} \boldsymbol{f}_j^{(m)}}{(\boldsymbol{f}_j^{(m)})^{\mathrm{T}} \boldsymbol{f}_j^{(m)}}) - \sum_{l=1}^{L_h} \sum_{j=1}^{k} u_{lj} (\theta_{lj} -$$
$$\frac{(\boldsymbol{p}_l^h)^{\mathrm{T}} \boldsymbol{f}_j^{(0)}}{(\boldsymbol{f}_j^{(0)})^{\mathrm{T}} \boldsymbol{f}_j^{(0)}}) + \sum_{l=1}^{L_h} \sum_{j=1}^{k} \frac{\rho}{2} (\theta_{lj} - \frac{(\boldsymbol{p}_l^h)^{\mathrm{T}} \boldsymbol{f}_j^{(m)}}{(\boldsymbol{f}_j^{(m)})^{\mathrm{T}} \boldsymbol{f}_j^{(m)}})^2 - \sum_{l=1}^{L_h} \sum_{j=1}^{k} \frac{\rho}{2} (\theta_{lj} - \frac{(\boldsymbol{p}_l^h)^{\mathrm{T}} \boldsymbol{f}_j^{(0)}}{(\boldsymbol{f}_j^{(0)})^{\mathrm{T}} \boldsymbol{f}_j^{(0)}})^2$$
$$= \frac{(\boldsymbol{f}_m^{(0)})^{\mathrm{T}} \boldsymbol{X}_h^{\mathrm{T}} \boldsymbol{X}_h \boldsymbol{f}_m^{(0)}}{(\boldsymbol{f}_m^{(0)})^{\mathrm{T}} \boldsymbol{f}_m^{(0)}} - \frac{(\boldsymbol{f}_m^{(m)})^{\mathrm{T}} \boldsymbol{X}_h^{\mathrm{T}} \boldsymbol{X}_h \boldsymbol{f}_m^{(m)}}{(\boldsymbol{f}_m^{(m)})^{\mathrm{T}} \boldsymbol{f}_m^{(m)}} + \sum_{l=1}^{L_h} u_{lm} (\frac{(\boldsymbol{p}_l^h)^{\mathrm{T}} \boldsymbol{f}_m^{(0)}}{(\boldsymbol{f}_m^{(0)})^{\mathrm{T}} \boldsymbol{f}_m^{(0)}} - \frac{(\boldsymbol{p}_l^h)^{\mathrm{T}} \boldsymbol{f}_m^{(m)}}{(\boldsymbol{f}_m^{(m)})^{\mathrm{T}} \boldsymbol{f}_m^{(m)}})$$
$$+ \sum_{l=1}^{L_h} \frac{\rho}{2} (\frac{(\boldsymbol{p}_l^h)^{\mathrm{T}} \boldsymbol{f}_m^{(m)}}{(\boldsymbol{f}_m^{(m)})^{\mathrm{T}} \boldsymbol{f}_m^{(m)}} + \frac{(\boldsymbol{p}_l^h)^{\mathrm{T}} \boldsymbol{f}_m^{(0)}}{(\boldsymbol{f}_m^{(0)})^{\mathrm{T}} \boldsymbol{f}_m^{(0)}} - 2\theta_{lm}) (\frac{(\boldsymbol{p}_l^h)^{\mathrm{T}} \boldsymbol{f}_m^{(m)}}{(\boldsymbol{f}_m^{(m)})^{\mathrm{T}} \boldsymbol{f}_m^{(m)}} - \frac{(\boldsymbol{p}_l^h)^{\mathrm{T}} \boldsymbol{f}_m^{(0)}}{(\boldsymbol{f}_m^{(0)})^{\mathrm{T}} \boldsymbol{f}_m^{(0)}}). \tag{26}$$

Let $\boldsymbol{F}$ denote the current indicator matrix, which is stored in memory in advance, and we denote the index of element 1 in the i-th row of $\boldsymbol{F}$ as $r, r \in [k]$. Let $\boldsymbol{f}_m^{(m)}$ represent the m-th column of $\boldsymbol{F}_i^{(m)}$, $\boldsymbol{f}_m^{(0)}$ represent the m-th column of $\boldsymbol{F}_i^{(0)}$, $\boldsymbol{f}_m$ represent the m-th column of $\boldsymbol{F}$, then there will be two situations for (26):

- **Situation 1**: $m = r$, which means the i-th element of $\boldsymbol{f}_m^{(m)}$ and $\boldsymbol{f}_m$ is 1. As a result, it follows that $\boldsymbol{f}_m^{(m)} = \boldsymbol{f}_m$. We define $\boldsymbol{\delta}_m = \boldsymbol{f}_m - \boldsymbol{f}_m^{(0)}$, where the i-th element of $\boldsymbol{\delta}_m$ equals 1 and all other elements equal 0. Let p_{il} denote the (i,l)-element of matrix $\boldsymbol{P}_h$, then it holds that:

$$\boldsymbol{X}_h \boldsymbol{f}_m = \boldsymbol{X}_h (\boldsymbol{f}_m^{(0)} + \boldsymbol{\delta}_m) = \boldsymbol{X}_h \boldsymbol{f}_m^{(0)} + \boldsymbol{x}_i^h,$$
$$(\boldsymbol{f}_m^{(0)})^{\mathrm{T}} \boldsymbol{f}_m^{(0)} = (\boldsymbol{f}_m - \boldsymbol{\delta}_m)^{\mathrm{T}} (\boldsymbol{f}_m - \boldsymbol{\delta}_m) = \boldsymbol{f}_m^{\mathrm{T}} \boldsymbol{f}_m - 1,$$
$$(\boldsymbol{p}_l^h)^{\mathrm{T}} \boldsymbol{f}_m^{(0)} = (\boldsymbol{p}_l^h)^{\mathrm{T}} (\boldsymbol{f}_m - \boldsymbol{\delta}_m) = (\boldsymbol{p}_l^h)^{\mathrm{T}} \boldsymbol{f}_m - p_{il}.$$

  Substitute the above equations into (26), we have:

$$\overline{\mathcal{L}}(m,i,h) = -\frac{\boldsymbol{f}_m^{\mathrm{T}} \boldsymbol{X}_h^{\mathrm{T}} \boldsymbol{X}_h \boldsymbol{f}_m}{\boldsymbol{f}_m^{\mathrm{T}} \boldsymbol{f}_m} + \frac{\boldsymbol{f}_m^{\mathrm{T}} \boldsymbol{X}_h^{\mathrm{T}} \boldsymbol{X}_h \boldsymbol{f}_m - 2(\boldsymbol{x}_i^h)^{\mathrm{T}} \boldsymbol{X}_h \boldsymbol{f}_m + (\boldsymbol{x}_i^h)^{\mathrm{T}} \boldsymbol{x}_i^h}{\boldsymbol{f}_m^{\mathrm{T}} \boldsymbol{f}_m - 1} +$$
$$\sum_{l=1}^{L_h} u_{lm} (\frac{(\boldsymbol{p}_l^h)^{\mathrm{T}} \boldsymbol{f}_m - p_{il}}{\boldsymbol{f}_m^{\mathrm{T}} \boldsymbol{f}_m - 1} - \frac{(\boldsymbol{p}_l^h)^{\mathrm{T}} \boldsymbol{f}_m}{\boldsymbol{f}_m^{\mathrm{T}} \boldsymbol{f}_m}) +$$
$$\sum_{l=1}^{L_h} \frac{\rho}{2} (\frac{(\boldsymbol{p}_l^h)^{\mathrm{T}} \boldsymbol{f}_m}{\boldsymbol{f}_m^{\mathrm{T}} \boldsymbol{f}_m} + \frac{(\boldsymbol{p}_l^h)^{\mathrm{T}} \boldsymbol{f}_m - p_{il}}{\boldsymbol{f}_m^{\mathrm{T}} \boldsymbol{f}_m - 1} - 2\theta_{lm}) (\frac{(\boldsymbol{p}_l^h)^{\mathrm{T}} \boldsymbol{f}_m}{\boldsymbol{f}_m^{\mathrm{T}} \boldsymbol{f}_m} - \frac{(\boldsymbol{p}_l^h)^{\mathrm{T}} \boldsymbol{f}_m - p_{il}}{\boldsymbol{f}_m^{\mathrm{T}} \boldsymbol{f}_m - 1}). \tag{27}$$

- **Situation 2**: $m \neq r$, which means the i-th element of $\boldsymbol{f}_m^{(0)}$ and $\boldsymbol{f}_m$ is 0. As a result, it follows that $\boldsymbol{f}_m^{(0)} = \boldsymbol{f}_m$. We define $\boldsymbol{\delta}_m = \boldsymbol{f}_m^{(m)} - \boldsymbol{f}_m$, where the i-th element of $\boldsymbol{\delta}_m$ equals 1 and all other elements equal 0. Let p_{il} denote the (i,l)-element of matrix $\boldsymbol{P}_h$, then it holds that:

$$\boldsymbol{X}_h \boldsymbol{f}_m^{(m)} = \boldsymbol{X}_h (\boldsymbol{f}_m + \boldsymbol{\delta}_m) = \boldsymbol{X}_h \boldsymbol{f}_m + \boldsymbol{x}_i^h$$
$$(\boldsymbol{f}_m^{(m)})^{\mathrm{T}} \boldsymbol{f}_m^{(m)} = (\boldsymbol{f}_m + \boldsymbol{\delta}_m)^{\mathrm{T}} (\boldsymbol{f}_m + \boldsymbol{\delta}_m) = \boldsymbol{f}_m^{\mathrm{T}} \boldsymbol{f}_m + 1$$
$$(\boldsymbol{p}_l^h)^{\mathrm{T}} \boldsymbol{f}_m^{(m)} = (\boldsymbol{p}_l^h)^{\mathrm{T}} (\boldsymbol{f}_m + \boldsymbol{\delta}_m) = (\boldsymbol{p}_l^h)^{\mathrm{T}} \boldsymbol{f}_m + p_{il}.$$

Substitute the above equations into (26), we have:

$$\overline{\mathcal{L}}(m,i,h) = \frac{\boldsymbol{f}_m^{\mathrm{T}}\boldsymbol{X}_h^{\mathrm{T}}\boldsymbol{X}_h\boldsymbol{f}_m}{\boldsymbol{f}_m^{\mathrm{T}}\boldsymbol{f}_m} - \frac{\boldsymbol{f}_m^{\mathrm{T}}\boldsymbol{X}_h^{\mathrm{T}}\boldsymbol{X}_h\boldsymbol{f}_m + 2(\boldsymbol{x}_i^h)^{\mathrm{T}}\boldsymbol{X}_h\boldsymbol{f}_m + (\boldsymbol{x}_i^h)^{\mathrm{T}}\boldsymbol{x}_i^h}{\boldsymbol{f}_m^{\mathrm{T}}\boldsymbol{f}_m - 1} +$$
$$\sum_{l=1}^{L_h} u_{lm}\left(\frac{(\boldsymbol{p}_l^h)^{\mathrm{T}}\boldsymbol{f}_m}{\boldsymbol{f}_m^{\mathrm{T}}\boldsymbol{f}_m} - \frac{(\boldsymbol{p}_l^h)^{\mathrm{T}}\boldsymbol{f}_m + p_{il}}{\boldsymbol{f}_m^{\mathrm{T}}\boldsymbol{f}_m - 1}\right) +$$
$$\sum_{l=1}^{L_h} \frac{\rho}{2}\left(\frac{(\boldsymbol{p}_l^h)^{\mathrm{T}}\boldsymbol{f}_m}{\boldsymbol{f}_m^{\mathrm{T}}\boldsymbol{f}_m} + \frac{(\boldsymbol{p}_l^h)^{\mathrm{T}}\boldsymbol{f}_m + p_{il}}{\boldsymbol{f}_m^{\mathrm{T}}\boldsymbol{f}_m + 1} - 2\theta_{lm}\right)\left(\frac{(\boldsymbol{p}_l^h)^{\mathrm{T}}\boldsymbol{f}_m + p_{il}}{\boldsymbol{f}_m^{\mathrm{T}}\boldsymbol{f}_m + 1} - \frac{(\boldsymbol{p}_l^h)^{\mathrm{T}}\boldsymbol{f}_m}{\boldsymbol{f}_m^{\mathrm{T}}\boldsymbol{f}_m}\right). \tag{28}$$

To update the i-th row of $\boldsymbol{F}$, we aggregate the objective values computed by (27) and (28) from each client, and incorporate them into our updating formulations:

$$f_{is} = \begin{cases} 1, & s = \arg\min_m \sum_{h=1}^{H} \overline{\mathcal{L}}(m,i,h); \\ 0, & \text{otherwise.} \end{cases} \tag{29}$$

Consider the objective function (27) and (28), which contains five terms: $\boldsymbol{f}_m^{\mathrm{T}}\boldsymbol{f}_m$, $\boldsymbol{f}_m^{\mathrm{T}}\boldsymbol{X}_h^{\mathrm{T}}\boldsymbol{X}_h\boldsymbol{f}_m$, $\boldsymbol{X}_h\boldsymbol{f}_m$, $(\boldsymbol{p}_l^h)^{\mathrm{T}}\boldsymbol{f}_m$, p_{il} and $(\boldsymbol{x}_i^h)^{\mathrm{T}}\boldsymbol{x}_i^h$. It is worth noting that the update formulas for $\boldsymbol{f}_m^{\mathrm{T}}\boldsymbol{f}_m$, $\boldsymbol{f}_m^{\mathrm{T}}\boldsymbol{X}_h^{\mathrm{T}}\boldsymbol{X}_h\boldsymbol{f}_m$, and $\boldsymbol{X}_h\boldsymbol{f}_m$ are provided in (19). Since p_{il} and $(\boldsymbol{x}_i^h)^{\mathrm{T}}\boldsymbol{x}_i^h$ are constants and require no updating, we present the update formula for $(\boldsymbol{p}_l^h)^{\mathrm{T}}\boldsymbol{f}_m$ as follows:

$$(\boldsymbol{p}_l^h)^{\mathrm{T}}\boldsymbol{f}_r \leftarrow (\boldsymbol{p}_l^h)^{\mathrm{T}}\boldsymbol{f}_r - p_{il}; \quad (\boldsymbol{p}_l^h)^{\mathrm{T}}\boldsymbol{f}_s \leftarrow (\boldsymbol{p}_l^h)^{\mathrm{T}}\boldsymbol{f}_s + p_{il}. \tag{30}$$

The computation of $\boldsymbol{F}$, along with other associated terms, has been completed. Moving forward, we intend to present an approach for solving the variables $\boldsymbol{\Theta}$ and $\boldsymbol{U}$.

2) $\boldsymbol{\Theta}$ and $\boldsymbol{U}$-solving procedure. When solving the variable $\boldsymbol{\Theta}$, the other two variables can be regarded as constants. Considering the optimization problem formulated in (25), it can be observed that this problem can be expressed as a quadratic programming (QP) problem with respect to $\boldsymbol{\Theta}$. As a result, the closed solution for $\boldsymbol{\Theta}$ can be obtained readily as follows:

$$\theta_{lj} = \begin{cases} \frac{(\boldsymbol{p}_l^h)^{\mathrm{T}}\boldsymbol{f}_j}{\boldsymbol{f}_j^{\mathrm{T}}\boldsymbol{f}_j} - \frac{u_{lj}}{\rho}, & \beta_l \le \frac{(\boldsymbol{p}_l^h)^{\mathrm{T}}\boldsymbol{f}_j}{\boldsymbol{f}_j^{\mathrm{T}}\boldsymbol{f}_j} \le \alpha_l, \\ \beta_l, & \frac{(\boldsymbol{p}_l^h)^{\mathrm{T}}\boldsymbol{f}_j}{\boldsymbol{f}_j^{\mathrm{T}}\boldsymbol{f}_j} < \beta_l, \\ \alpha_l, & \frac{(\boldsymbol{p}_l^h)^{\mathrm{T}}\boldsymbol{f}_j}{\boldsymbol{f}_j^{\mathrm{T}}\boldsymbol{f}_j} > \alpha_l, \end{cases} \quad l \in [L_h], j \in [k]. \tag{31}$$

To solve the dual variable $\boldsymbol{U}_h$, we make use of the inherent properties of the ADMM framework [13], which enables us to derive an updating formula for $\boldsymbol{U}_h$:

$$u_{lj} \leftarrow u_{lj} + \rho\left(\theta_{lj} - \frac{(\boldsymbol{p}_l^h)^{\mathrm{T}}\boldsymbol{f}_j}{\boldsymbol{f}_j^{\mathrm{T}}\boldsymbol{f}_j}\right), \quad l \in [L_h],\ j \in [k]. \tag{32}$$

Remarks. When a client possesses sensitive attributes, a fairness-constrained subproblem is solved by using the Lagrangian function (27) and (28). Moreover, if a client does not have sensitive attributes, an unconstrained subproblem is solved using the degenerated function (17). Both types of clients contribute their objective values to the VFL process. Furthermore, the distribution of sensitive features among clients has no impact on the final result, as determined by the separable nature of the optimization problem (20).

4.2.3 Guarantee for feasible solutions. Taking into account that the fair k-means problem does not necessarily have a feasible solution (due to inappropriate choices of $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$), we propose several methods to improve the feasibility of the solutions and reduce the number of iterations.

- **Choice of penalty parameter.** The penalty parameter is used in ADMM algorithms to penalize the violation of constraints and control the strength of the penalty. By manipulating the value of the penalty parameter, it becomes possible to achieve a balance between the clustering utility and the level of adherence to fairness constraints. If the penalty parameter is too small, the algorithm may not pay enough attention to the fair constraints, leading to infeasible solutions. Conversely, if the penalty parameter is too large, the algorithm may suffer from numerical instability, preventing the algorithm from converging or outputting infeasible solutions. In practice, the penalty parameter can be gradually increased during iterations by adopting the penalty parameter updating scheme, also known as the penalty parameter increasing method [50]. This method can balance the strength of the penalty and the convergence rate of the algorithm, thereby enhancing the feasibility of obtaining feasible solutions.
- **Stop criterion.** We have reviewed that a fair k-means solution has λ-additive violation if the RD and MP constraints are satisfied up to $\pm\lambda$-violation in Section 3.3. To express these constraints with λ-additive violation in the form of matrix multiplication, we present the following formulation:

$$\beta_l \boldsymbol{f}_j^{\mathrm{T}} \boldsymbol{f}_j - \lambda \leq \boldsymbol{p}_l^{\mathrm{T}} \boldsymbol{f}_j \leq \alpha_l \boldsymbol{f}_j^{\mathrm{T}} \boldsymbol{f}_j + \lambda, j \in [k], l \in [L_h]. \tag{33}$$

  We consider using the inequalities (33) as a stopping criterion for the F^{3}KM iterations. Specifically, when the inequalities (33) are satisfied during the F^{3}KM iterations, we consider the solution to be fair and output the corresponding indicator matrix $\boldsymbol{F}$.

4.3 Complexity Analysis

The high cost of communication and computation in federated learning is a major bottleneck that limits its efficiency. In the following, we present the theoretical communication and computational complexity of the F^{3}KM algorithm, and elaborate in detail why F^{3}KM is an efficient algorithm in terms of communication and computation.

Theorem 1. *Let b denote the number of blocks, and T denote the number of iterations, then the communication complexity of F^{3}KM is given by $O(bT)$.*

Proof. From Algorithm 4 and Figure 6, we can infer that in each iteration of F^{3}KM, communication between the server and clients is required b times for solving $\boldsymbol{F}$, and one communication is needed for solving $\boldsymbol{\Theta_h}$ and $\boldsymbol{U}_h$. Therefore, the total number of communications required in each iteration is $(b+1)$. Let T denote the total number of iterations of F^{3}KM, then the total number of communications required by F^{3}KM is $O(bT)$. □

Proposition 1. *Our F^{3}KM is a communication-efficient federated learning algorithm. This is because in federated learning, the number of communication rounds is often proportional to the size of the dataset. F^{3}KM reduces communication complexity to a constant by employing the BCD method. Specifically, during the computation of the indicator matrix $\boldsymbol{F}$, F^{3}KM updates n_b rows of $\boldsymbol{F}$ at a time instead of a single row. This approach alleviates the communication pressure by leveraging local computation, which is much faster than communication.*

Theorem 2. *The computation complexity of F^{3}KM on each client side is $O(n(d_h + L_h)kT)$.*

Proof. Given the dataset $\boldsymbol{X}_h$ and the color matrix $\boldsymbol{P}_h$, we aim to evaluate the algorithmic complexity of F^{3}KM on the client side (Algorithm 5) by dividing the computation into four parts. Firstly, the computation of $\boldsymbol{X}_h \boldsymbol{f}_m$, $\boldsymbol{f}_m^{\mathrm{T}} \boldsymbol{f}_m$, and $(\boldsymbol{p}_l^h)^{\mathrm{T}} \boldsymbol{f}_m$ $(i \in [n], m \in [k])$, all of which are stored in

Table 2. An overview of the datasets

Ref.	Dataset	n	d	Sensitive attributes	# of groups
[31]	Athlete	271,117	15	sex, season	2, 2
[11]	Bank	4,521	16	marital, default	3, 2
[11]	Census	32,561	15	sex, race	2, 5
[11]	Creditcard	30,000	24	marriage, education	4, 7
[11]	Diabetes	101,766	50	gender, race	2, 6
[3]	Recruitment	4,001	15	gender, ind-degree	2, 3
[7]	Spanish	4,747	21	gender	3
[1]	Student	32,594	12	gender,disability	2, 2
[11]	Census1990	2,458,285	69	iSex, dAge	2, 8
[2]	HMDA	5,986,660	53	ethnicity, race	3, 7

memory, requires nd_hk, nk, and nL_hk additions, respectively. The computation of $(\boldsymbol{x}_i^h)^{\mathrm{T}}\boldsymbol{f}_m$ requires nd_h multiplications, and the computation of $\boldsymbol{f}_m^{\mathrm{T}}\boldsymbol{X}_h^{\mathrm{T}}\boldsymbol{X}_h\boldsymbol{f}_m$ requires d_hk multiplications. Secondly, the computation in step 4 involves $2nk+3nL_hk$ additions, $n(d_h+3L_h)k$ multiplications and $2nL_hk+2nk$ divisions. Thirdly, steps 7 and 8 require $2n(d_h+L_h)$ additions, and finally, steps 10 and 11 require $2L_hk$ additions and L_hk multiplications. Given that the time required for multiplications and divisions is significantly higher than that for additions, we limit our analysis to the number of multiplications and divisions. The total number of multiplications and divisions is thus given by $(nd_h + d_hk + n(d_h + 3L_h)k + 2nL_hk + 2nk + L_hk)$. Therefore, when the algorithm is executed for T iterations, the total algorithmic complexity is $O(n(d_h + L_h)kT)$. □

Proposition 2. *The state-of-the-art method for solving the fair k-means problem, known as Fair-LP [11], employs nk LP variables. However, due to the high complexity of LP [16], this method becomes computationally impractical for large input data sizes. In contrast, our proposed method, F^{3}KM, reduces the complexity to linear in the input data size, enabling efficient and fast solutions to the fair k-means problem. As a result, our method has the potential to advance the field of fair clustering and promote the development of more scalable and practical algorithms.*

5 EXPERIMENTS

Goals. In the forthcoming experiments, we intend to validate the effectiveness and efficiency of F^{3}KM. With respect to effectiveness, our primary focus is to weigh the trade-off between utility and fairness. Regarding the algorithm's efficiency, we take into account the communication rounds and the computation time.

5.1 Settings

Datasets. To evaluate the performance of F^{3}KM, we employ a diverse range of ten real-world datasets, namely Athlete, Bank, Census, Creditcard, Diabetes, Recruitment, Spanish, Student, Census1990, and HMDA. Of these, Census1990 and HMDA represent datasets of a million-scale magnitude, utilized specifically to scrutinize the computational efficiency of F^{3}KM. For each dataset, we subsample several features and select at least one sensitive feature, such as race and sex. A comprehensive overview of the datasets can be obtained from Table 2.

Baselines. We experimentally evaluate the performance of F^{3}KM against three methods, namely, Fair-LP [11], CDKM [41], and Lloyd's heuristic [38]. As explained in our related works, Fair-LP is the state-of-the-art method for k-means with fair constraints. CDKM and Lloyd's heuristic are two methods that perform k-means clustering without fair constraints: CDKM is a clustering method which can usually converge to a better local minimum than Lloyd's heuristic.

Measurements. We employ several metrics to evaluate the trade-off between utility and fairness in our study. The first metric is the sum of squared error (*SSE*), which has been previously introduced in Section 3.1. We also introduce a notion called *relative error*, denoted as $\frac{SSE(S')}{SSE(S)}$, to assess the clustering utility. Here, S' represents the results obtained from either F^{3}KM, Fair-LP, or CDKM, while S represents the results obtained from Lloyd's heuristic. For fairness evaluation, we adopt the *balance* proposed by Chierichetti et al. [15]. Specifically, let $\eta_l := \frac{\boldsymbol{p}_l^{\mathrm{T}}\boldsymbol{p}_l}{n}$ denote the proportion of points in group l over the entire dataset, and let $\eta_l(j) := \frac{\boldsymbol{p}_l^{\mathrm{T}}\boldsymbol{f}_j}{\boldsymbol{f}_j^{\mathrm{T}}\boldsymbol{f}_j}$ denote the proportion of points in group l over the points in cluster j. Then, we define $balance(j)$ as $\min\{\frac{\eta_l}{\eta_l(j)}, \frac{\eta_l(j)}{\eta_l}\}$, where $l \in [L]$ and $j \in [k]$. We parameterize $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ as follows: $\alpha_l = \frac{\eta_l}{1-\delta}$ and $\beta_l = \eta_l(1-\delta)$, where δ is a hyperparameter that adjusts the gap between $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$. For the evaluation of efficiency, we employ both the communication rounds and the running time as metrics.

Implementations. Our algorithms were executed on a computational platform comprising an Intel i7-8750H CPU with 6 cores, 32 GB of RAM, and operating on the Windows 10 environment. The software implementation was realized in Matlab 2020b and open-sourced[2]. In addition, we provide an implementation of F^{3}KM in Scala 2.12 and Spark 3.0, utilizing the MapReduce paradigm.

5.2 Effectiveness Analysis

In this section, our main focus is on examining the trade-off between utility and fairness. To this end, we undertake three distinct sets of experiments, specifically: (i) exploring the *SSE* as a function of varying k; (ii) investigating the variation of relative error as a function of varying δ; and (iii) analyzing the balance of the four methods within each cluster, given fixed k and δ. Ultimately, we present a summary of our experimental findings.

5.2.1 SSE vs. number of clusters. Due to the considerable computational overhead of executing Fair-LP, we opt to downsample each dataset to 1000 data points to conform with the computational constraints of Fair-LP [11]. The hyperparameter δ is set to 0.5, the values of n_b are set to 200, 500, and 1000, and after subjecting each method to 50 repetitions, we obtain the mean *SSE*, enabling the visualization of the *SSE* versus the number of clusters in Figure 7.

Observations. From Figure 7, we can observe that the *SSE* of all methods decreases as the number of clusters increases. Furthermore, CDKM exhibits a consistently lower *SSE* compared to Lloyd's heuristic, providing empirical evidence that CDKM can indeed converge to a superior local minimum. This finding aligns with the conclusion drawn in [41]. In addition, F^{3}KM typically achieves a lower *SSE* than Fair-LP on eight datasets, suggesting that F^{3}KM, an in-processing method, outperforms Fair-LP, a post-processing method, in terms of the trade-off between utility and fairness. Moreover, in certain scenarios, F^{3}KM attains an even lower *SSE* than Lloyd's heuristic, indicating that F^{3}KM inherits the advantage of CDKM in terms of converging to a better local minimum. Additionally, when selecting a smaller value for n_b in F^{3}KM, a smaller *SSE* can be achieved. This is attributed to the fact that a smaller n_b allows for more parameters updates within a single iteration.

[2]https://github.com/zsk66/F3KM-MATLAB

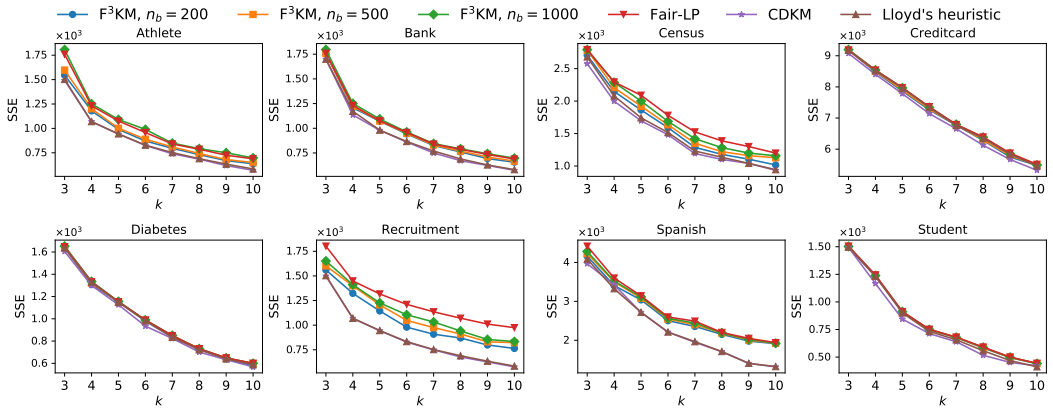

Fig. 7. The *SSE* of multiple methods at varying values of k.

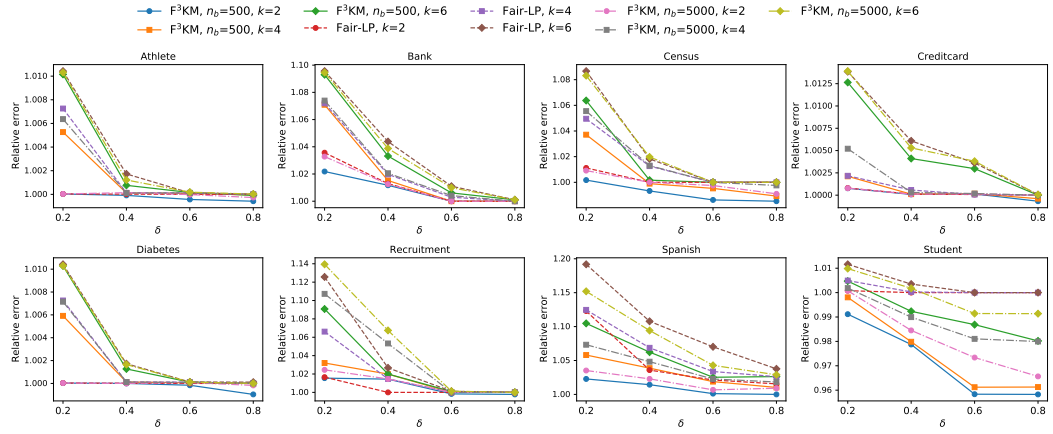

Fig. 8. Comparison between F³KM and Fair-LP in terms of varying values of k and δ.

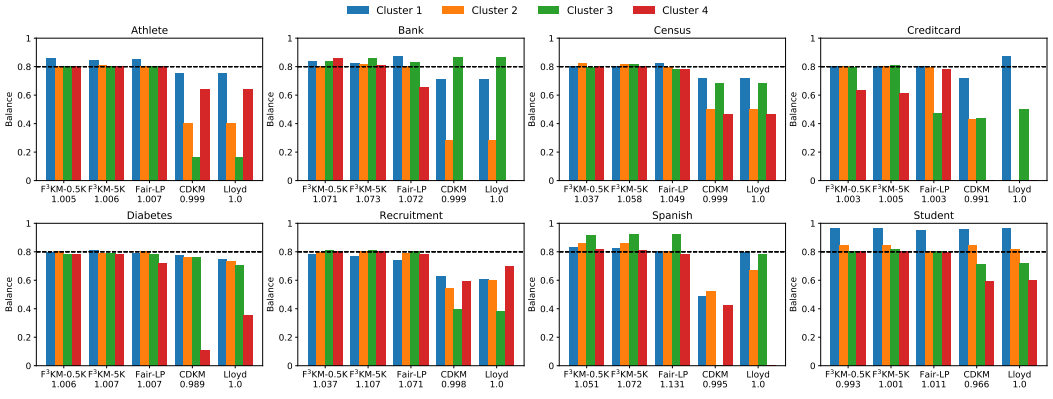

Fig. 9. Evaluation of multiple methods in terms of their balance, where the numbers below the x-axis are relative errors.

5.2.2 Relative error vs. δ. We examine the impact of fairness constraints on the utility of clustering by exploring the changes in relative error resulting from adjusting the gap between $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ in the fairness constraints. Our experiments utilize the entire samples in the datasets and, to reduce time consumption, each algorithm is run 10 times, with the average relative error being computed. Figure 8 presents four distinct values of δ (0.2, 0.4, 0.6, and 0.8), and we observe the changes in relative error of F^{3}KM and Fair-LP at different values of k and n_b as δ varies.

Observations. Based on the observations from Figure 8, it is evident that an increase in δ leads to a decrease in the relative error of both F^{3}KM and Fair-LP at various k values. This can be attributed to the widening gap between $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ with increasing δ, which facilitates the fulfillment of fairness constraints and yields feasible solutions with smaller *SSE* for the fair k-means problem. Additionally, a decrease in k results in a reduction in the relative error due to the associated decrease in the number of fairness constraints, thereby leading to the attainment of feasible solutions with smaller *SSE* for the fair k-means problem. Furthermore, the relative error of F^{3}KM with $n_b = 500$ is often lower than that of Fair-LP at the same k value, thus confirming the superior trade-off between utility and fairness offered by F^{3}KM compared to Fair-LP. However, it should be noted that when $n_b = 5000$, in certain datasets such as Recruitment, the relative error of F^{3}KM exceeds that of Fair-LP.

5.2.3 Balance vs. four methods. In the following analysis, we aim to investigate the balance metric and its corresponding relative error with respect to distinct methods. To ensure a consistent evaluation, we set the value of δ to 0.2, following the widely accepted interpretation of the 80% rule in the DI doctrine [11]. Specifically, we apply this assessment on eight datasets, where the parameter k is fixed at 4. The outcomes of this examination are presented in Figure 9, illustrating the balance values and relative error for each method across the four clusters. The abbreviations F^{3}KM-0.5K and F^{3}KM-5K correspond to F^{3}KM with n_b =500 and 5000, respectively.

Observations. In Figure 9, we present the balance values of several clustering methods for each cluster. Notably, the CDKM and Lloyd's heuristic, which lack fairness constraints, exhibit the lowest balance values and frequently fail to satisfy the threshold of 0.8. Conversely, the balance values of F^{3}KM and Fair-LP are demonstrably higher than those of the unconstrained methods. Nevertheless, our findings indicate that Fair-LP often falls short of achieving a balance value of 0.8 in some clusters, leading to additive violation with a maximum value of 3, as evidenced in the Creditcard dataset. In contrast, F^{3}KM achieves a balance value of 0.8 in most clusters of the datasets, with an additive violation of 2 solely in the fourth cluster of the Creditcard dataset, which is smaller than that of Fair-LP. Additionally, our observations reveal that F^{3}KM-0.5K has a lower relative error and a larger balance value than Fair-LP. However, it is noteworthy that the relative error of F^{3}KM-5K may occasionally exceed that of Fair-LP. Hence, selecting an appropriate n_b in F^{3}KM can better ensure the trade-off between utility and fairness.

5.2.4 Summary of Lessons Learned. We have presented an analysis of the changes in *SSE* with respect to k, the variations of relative error as a function of δ, and a comparison of balance among different methods for given values of k and δ. Our experimental results have led us to draw the following conclusions:

- Selecting an appropriate n_b in F^{3}KM can better ensure the desired trade-off between utility and fairness compared to state-of-the-art methods. This can be attributed to the decrease in the number of parameter updates per iteration as n_b increases.
- Modifying the value of k and δ in fair k-means clustering can impact its utility. Specifically, an increase in k and a decrease in δ result in a reduction in the clustering utility, attributed to the imposition of additional and more stringent fairness constraints.

Table 3. The communication rounds and corresponding relative error at varying n_b.

Dataset	Communication rounds						Relative error					
	1	200	500	1000	5000	10000	1	200	500	1000	5000	10000
Athlete	13.56M	67.80K	27.15K	13.60K	2.75K	1.40K	1.005	1.005	1.005	1.006	1.006	1.006
Bank	0.23M	1.15K	0.50K	0.25K	0.05K	0.05K	1.064	1.069	1.070	1.071	1.073	1.073
Census	1.63M	8.15K	3.30K	1.65K	0.35K	0.20K	1.004	1.008	1.009	1.027	1.055	1.057
Creditcard	1.50M	7.50K	30.00K	1.50K	0.30K	0.15K	1.003	1.003	1.003	1.004	1.005	1.008
Diabetes	5.09M	25.45K	10.20K	5.10K	1.05K	0.55K	1.005	1.006	1.006	1.006	1.007	1.007
Recruitment	0.20M	1.00K	0.40K	0.20K	0.05K	0.05K	1.031	1.031	1.037	1.061	1.107	1.107
Spanish	0.24M	1.20K	0.50K	0.25K	0.05K	0.05K	1.020	1.022	1.031	1.036	1.080	1.080
Student	1.63M	8.15K	3.30K	1.65K	0.35K	0.20K	0.984	0.989	0.991	0.993	0.995	0.996

- When the fairness constraints become excessively stringent, the fair k-means problem may lack feasible solutions, resulting in an additive violation. Our experimental results have substantiated that the additive violation of F^{3}KM is smaller than the current state-of-the-art methods.

5.3 Efficiency Analysis

5.3.1 Number of communications vs. n_b. Table 2 shows the variations in communication rounds and relative error of F^{3}KM as a function of n_b. We have chosen a maximum of 50 iterations for F^{3}KM, with k assigned a value of 4, δ set to 0.2, and n_b is assigned values of 1, 0.2K, 0.5K, 1K, 5K, and 10K. In the event that the value of n_b exceeds the dimension of the dataset, n_b is set equal to the dataset size.

Observations. Based on the findings in Table 3, it is evident that increasing the value of n_b leads to a significant reduction in the number of communications. Specifically, the BCD method demonstrates a substantial reduction in the number of communications, with the reduction reaching nearly 10K times when $n_b = 10$K as compared to the coordinate descent method ($n_b = 1$). However, increasing n_b can also adversely affect the clustering utility as demonstrated by the observed deterioration in relative error. Specifically, in the Recruitment dataset, the maximum relative error increased by approximately 10.7% when $n_b = 10$K as compared to $n_b = 1$, while the Athlete dataset showed a minimal increase of 0.6%. The primary cause of the high relative error in the Recruitment dataset is that $n_b > n$, leading to difficulties in variable $\boldsymbol{F}$ converging to a local minimum during the computing and update process. Based on our extensive experience, we recommend that $n_b = \frac{1}{4}n$ be selected to achieve a significant reduction in communication rounds while simultaneously maintaining a low relative error.

5.3.2 Running time vs. sampling size. Next, we compare the running time of F^{3}KM and Fair-LP. We set the value of k to 3, the value of δ to 0.5, and the value of n_b to 1/4 of the sampling size. Table 4 provides information on how the running time of both algorithms vary with the sampling size.

Observations. Table 4 illustrates the relationship between running time and sampling size for F^{3}KM and Fair-LP on two large-scale datasets, namely Census1990 and HMDA. It is evident that both methods demonstrate an increase in running time as sampling size increases. Notably, when the sampling size is relatively small, such as 50K and 0.1M on the HMDA dataset, Fair-LP exhibits lower running time compared to F^{3}KM. Conversely, as the sampling size increases, F^{3}KM converges faster than Fair-LP. This is determined by the complexity of the two algorithms. Fair-LP can only accomplish computations within the allotted time (i.e., 1 hour) when the sampling size is

Table 4. Comparison between F^{3}KM and Fair-LP in terms of running time (seconds). We abbreviate TLE as the time limit exceeded for 1 hour, and SLE, as the sampling size limit exceeded for the dataset dimension.

Dataset	Method	50K	0.1M	0.2M	0.5M	1M	2M	5M
Census1990	F^{3}KM	30.4	56.9	124.1	422.6	1045.3	1966.7	SLE
	Fair-LP	35.7	93.5	261.8	1722.6	TLE	TLE	SLE
HMDA	F^{3}KM	30.6	48.8	102.5	328.1	775.6	1304.8	3459.4
	Fair-LP	14.7	27.8	161.4	1201.8	TLE	TLE	TLE

approximately 0.5M, while F^{3}KM can still converge within the same time when the sampling size is 5M. Consequently, F^{3}KM is more computationally efficient than Fair-LP in the context of big data.

5.3.3 Summary of Lessons Learned. We have ascertained the efficiency of F^{3}KM via an extensive investigation of both communication and computation complexity. Our experimental findings have led to the following definitive conclusions:

- Our results demonstrate that selecting an appropriate value of n_b can lead to a substantial reduction in the communication rounds of F^{3}KM when using BCD, as compared to CD. Moreover, this reduction in communication is attained with minimal compromise on utility, as evidenced by the low level of the relative error.
- In terms of computation efficiency, F^{3}KM outperforms the state-of-the-art method by enabling fair k-means clustering on datasets with a magnitude of 5M within a time span of 1 hour, whereas Fair-LP is limited to handling datasets with a magnitude of 0.5M within the same time.

6 CONCLUSIONS AND FUTURE WORK

This paper investigated the issue of fair k-means in situations where data cannot be shared among multiple parties. To address this issue, we introduced F^{3}KM, an algorithm designed to effectively solve the fair k-means problem within the VFL framework. We decomposed the fair k-means problem into multiple subproblems and distributed them among the clients. We proposed using the ADMM to solve each subproblem. Notably, during the solution process, the server and clients exchanged only the computed results, without sharing the raw data. Our theoretical analysis shows that F^{3}KM achieves a linear speedup in communication complexity, and reduces the computation complexity to linear with respect to the dataset size. Our experiments verified F^{3}KM outperformed other baseline methods in terms of the trade-off between clustering utility and fairness/efficiency.

In future, we plan to further enhance the applicability and versatility of F^{3}KM, and explore the integration of diverse types of data into our algorithm. For instance, we consider extending F^{3}KM to encompass additional clustering objective functions, such as k-median. Furthermore, we will investigate the incorporation of streaming data, addressing the inclusion of sensitive groups defined by attribute patterns, as well as handling data stored in various formats, such as graph data.

7 ACKNOWLEDGEMENT

Sheng Wang was supported by the National Natural Science Foundation of China (62202338) and Key R&D in Hubei Province (2023BAB081). Jinshan Zeng was supported by the National Natural Science Foundation of China (62376110, 61977038), and Thousand Talents Plan of Jiangxi Province (jxsq2019201124), and the Jiangxi Provincial Natural Science Foundation for Distinguished Young Scholars (20224ACB212004). Besides, we also thank anonymous reviewers for their helpful reports.

REFERENCES

[1] 2015. Open University Learning Analytics dataset. https://analyse.kmi.open.ac.uk/open_dataset.
[2] 2017. The Home Mortgage Disclosure Act. https://ffiec.cfpb.gov/data-browser/.
[3] 2022. Utrecht Fairness Recruitment dataset. https://www.kaggle.com/datasets/ictinstitute/utrecht-fairness-recruitment-dataset.
[4] Sara Ahmadian, Alessandro Epasto, Ravi Kumar, and Mohammad Mahdian. 2019. Clustering without over-representation. In *KDD*. 267–275.
[5] Mihael Ankerst, Markus M Breunig, Hans-Peter Kriegel, and Jörg Sander. 1999. OPTICS: Ordering points to identify the clustering structure. In *SIGMOD*, Vol. 28. 49–60.
[6] Abolfazl Asudeh, HV Jagadish, Julia Stoyanovich, and Gautam Das. 2019. Designing fair ranking schemes. In *SIGMOD*. 1259–1276.
[7] Tsanas Athanasios and Little Max. 2009. UCI Machine Learning Repository: Gender Gap in Spanish WP Data Set. https://archive.ics.uci.edu/ml/datasets/Parkinsons+Telemonitoring/
[8] Olivier Bachem, Mario Lucic, and Andreas Krause. 2018. Scalable k-means clustering via lightweight coresets. In *KDD*. 1119–1127.
[9] Arturs Backurs, Piotr Indyk, Krzysztof Onak, Baruch Schieber, Ali Vakilian, and Tal Wagner. 2019. Scalable fair clustering. In *ICML*. 405–413.
[10] Maria-Florina F Balcan, Steven Ehrlich, and Yingyu Liang. 2013. Distributed k-means and k-median clustering on general topologies. In *NeurIPS*. 1995–2003.
[11] Suman Bera, Deeparnab Chakrabarty, Nicolas Flores, and Maryam Negahbani. 2019. Fair algorithms for clustering. In *NeurIPS*. 4955–4966.
[12] Suman Bera, Syamantak Das, Sainyam Galhotra, and Sagar Sudhir Kale. 2022. Fair k-Center Clustering in MapReduce and Streaming Settings. In *WWW*. 1414–1422.
[13] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, Jonathan Eckstein, et al. 2011. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Found. Trends Mach. Learn* 3, 1 (2011), 1–122.
[14] Badrish Chandramouli, Junyi Xie, and Jun Yang. 2006. On the database/network interface in large-scale publish/subscribe systems. In *SIGMOD*. 587–598.
[15] Flavio Chierichetti, Ravi Kumar, Silvio Lattanzi, and Sergei Vassilvitskii. 2017. Fair clustering through fairlets. In *NeurIPS*. 5029–5037.
[16] Michael B Cohen, Yin Tat Lee, and Zhao Song. 2019. Solving linear programs in the current matrix multiplication time. In *STOC*. 938–942.
[17] Graham Cormode and Shan Muthukrishnan. 2005. An improved data stream summary: the count-min sketch and its applications. *J. Algorithms* 55, 1 (2005), 58–75.
[18] Don Kurian Dennis, Tian Li, and Virginia Smith. 2021. Heterogeneity for the win: One-shot federated clustering. In *ICML*. 2611–2620.
[19] Hu Ding, Yu Liu, Lingxiao Huang, and Jian Li. 2016. k-means clustering with distributed dimensions. In *ICML*. 1339–1348.
[20] Arash Fard, Anh Le, George Larionov, Waqas Dhillon, and Chuck Bear. 2020. Vertica-ml: Distributed machine learning in vertica database. In *SIGMOD*. 755–768.
[21] Tom Farrand, Fatemehsadat Mireshghallah, Sahib Singh, and Andrew Trask. 2020. Neither private nor fair: Impact of data imbalance on utility and fairness in differential privacy. In *CCS*. 15–19.
[22] Michael Feldman, Sorelle A Friedler, John Moeller, Carlos Scheidegger, and Suresh Venkatasubramanian. 2015. Certifying and removing disparate impact. In *KDD*. 259–268.
[23] Fangcheng Fu, Huanran Xue, Yong Cheng, Yangyu Tao, and Bin Cui. 2022. Blindfl: Vertical federated machine learning without peeking into your data. In *SIGMOD*. 1316–1330.
[24] Mehrdad Ghadiri, Samira Samadi, and Santosh Vempala. 2021. Socially fair k-means clustering. In *FAccT*. 438–448.
[25] Avishek Ghosh, Jichan Chung, Dong Yin, and Kannan Ramchandran. 2020. An Efficient Framework for Clustered Federated Learning. In *NeurIPS*. 19586–19597.
[26] Mehmet Gönen and Adam A Margolin. 2014. Localized data fusion for kernel k-means clustering with application to cancer biology. In *NeurIPS*. 1305–1313.
[27] Elfarouk Harb and Ho Shan Lam. 2020. KFC: A Scalable Approximation Algorithm for k- center Fair Clustering. In *NeurIPS*. 14509–14519.
[28] Xi He, Ashwin Machanavajjhala, and Bolin Ding. 2014. Blowfish privacy: Tuning privacy-utility trade-offs using policies. In *SIGMOD*. 1447–1458.
[29] Zhicheng He, Wei Xia, Kai Dong, Huifeng Guo, Ruiming Tang, Dingyin Xia, and Rui Zhang. 2022. Unsupervised Learning Style Classification for Learning Path Generation in Online Education Platforms. In *KDD*. 2997–3006.

[30] Lingxiao Huang, Shaofeng Jiang, and Nisheeth Vishnoi. 2019. Coresets for clustering with fairness constraints. In *NeurIPS*. 7587–7598.
[31] Lingxiao Huang, Zhize Li, Jialin Sun, and Haoyu Zhao. 2022. Coresets for Vertical Federated Learning: Regularized Linear Regression and k-Means Clustering. In *NeurIPS*, Vol. 35. 29566–29581.
[32] Yefan Huang, Xiaoli Wang, Feiyan Liu, and Guofeng Huang. 2022. OVQA: A Clinically Generated Visual Question Answering Dataset. In *SIGIR*. 2924–2938.
[33] Maliha Tashfia Islam, Anna Fariha, Alexandra Meliou, and Babak Salimi. 2022. Through the data management lens: Experimental analysis and evaluation of fair classification. In *SIGMOD*. 232–246.
[34] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, et al. 2021. Advances and open problems in federated learning. *Found. Trends Mach. Learn.* 14, 1–2 (2021), 1–210.
[35] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. 2020. Federated learning: Challenges, methods, and future directions. *IEEE Signal Process. Mag.* 37, 3 (2020), 50–60.
[36] Zitao Li, Tianhao Wang, and Ninghui Li. 2022. Differentially Private Vertical Federated Clustering. *Proc. VLDB Endow.* 16, 6 (2022), 1277 – 1290.
[37] Edo Liberty, Zohar Karnin, Bing Xiang, Laurence Rouesnel, Baris Coskun, Ramesh Nallapati, Julio Delgado, Amir Sadoughi, Yury Astashonok, Piali Das, et al. 2020. Elastic machine learning algorithms in amazon sagemaker. In *SIGMOD*. 731–737.
[38] Stuart Lloyd. 1982. Least squares quantization in PCM. *IEEE Trans. Inf. Theory* 28, 2 (1982), 129–137.
[39] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *AISTATS*. PMLR, 1273–1282.
[40] Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. 2021. A survey on bias and fairness in machine learning. *ACM Comput. Surv.* 54, 6 (2021), 1–35.
[41] Feiping Nie, Jingjing Xue, Danyang Wu, Rong Wang, Hui Li, and Xuelong Li. 2021. Coordinate Descent Method for k-means. *IEEE Trans. Pattern Anal. Mach. Intell.* 44, 5 (2021), 2371–2385.
[42] John Paparrizos and Luis Gravano. 2015. k-shape: Efficient and accurate clustering of time series. In *SIGMOD*. 1855–1870.
[43] Erich Schubert, Alexander Koos, Tobias Emrich, Andreas Züfle, Klaus Arthur Schmid, and Arthur Zimek. 2015. A framework for clustering uncertain data. *Proc. VLDB Endow.* 8, 12 (2015), 1976–1979.
[44] Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet S Talwalkar. 2017. Federated multi-task learning. In *NeurIPS*, Vol. 30. 4424–4434.
[45] Yanchao Tan, Carl Yang, Xiangyu Wei, Chaochao Chen, Weiming Liu, Longfei Li, Jun Zhou, and Xiaolin Zheng. 2022. Metacare++: Meta-learning with hierarchical subtyping for cold-start diagnosis prediction in healthcare data. In *SIGIR*. 449–459.
[46] Suhas Thejaswi, Ameet Gadekar, Bruno Ordozgoiti, and Michal Osadnik. 2022. Clustering with Fair-Center Representation: Parameterized Approximation Algorithms and Heuristics. In *KDD*. 1749–1759.
[47] Sheng Wang, Yuan Sun, and Zhifeng Bao. 2020. On the efficiency of k-means clustering: evaluation, optimization, and algorithm selection. *Proc. VLDB Endow.* 14, 2, 163–175.
[48] Yu Wang, Wotao Yin, and Jinshan Zeng. 2019. Global convergence of ADMM in nonconvex nonsmooth optimization. *J. Sci. Comput.* 78 (2019), 29–63.
[49] Xindong Wu, Vipin Kumar, J Ross Quinlan, Joydeep Ghosh, Qiang Yang, Hiroshi Motoda, Geoffrey J McLachlan, Angus Ng, Bing Liu, Philip S Yu, et al. 2008. Top 10 algorithms in data mining. *Knowl. Inf. Syst.* 14, 1 (2008), 1–37.
[50] Yi Xu, Mingrui Liu, Qihang Lin, and Tianbao Yang. 2017. ADMM without a fixed penalty parameter: Faster convergence with new adaptive penalization. In *NeurIPS*. 1267–1277.
[51] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. 2019. Federated machine learning: Concept and applications. *ACM Trans. Intell. Syst. Technol.* 10, 2 (2019), 1–19.
[52] Zhenkun Yang, Chuanhui Yang, Fusheng Han, Mingqiang Zhuang, Bing Yang, Zhifeng Yang, Xiaojun Cheng, Yuzhong Zhao, Wenhui Shi, Huafeng Xi, et al. 2022. OceanBase: a 707 million tpmC distributed relational database system. *Proc. VLDB Endow.* 15, 12 (2022), 3385–3397.
[53] Jinshan Zeng, Tim Tsz-Kit Lau, Shaobo Lin, and Yuan Yao. 2019. Global convergence of block coordinate descent in deep learning. In *ICML*. 7313–7323.
[54] Juntao Zhang, Sheng Wang, Yuan Sun, and Zhiyong Peng. 2023. Prerequisite-driven Fair Clustering on Heterogeneous Information Networks. *Proc. ACM Manag. Data* 1, 2 (2023), 1–27.
[55] Yang Zhou, Hong Cheng, and Jeffrey Xu Yu. 2009. Graph clustering based on structural/attribute similarities. *Proc. VLDB Endow.* 2, 1 (2009), 718–729.

Received April 2023; revised July 2023; accepted August 2023