
# Highly-Efficient Large-Scale $k$-means with Individual Fairness

Shengkun Zhu[1], Jinshan Zeng[2], Yuan Sun[3], Sheng Wang[1*], Yiming Wang[1], Yushuai Ji[1], Feiping Nie[4], Xiaodong Li[5], Zhiyong Peng[1]
[1]School of Computer Science, Wuhan University, [2]School of Management, Xi'an Jiaotong University, [3]La Trobe Business School, La Trobe University, [4]The School of Artificial Intelligence, Optics and Electronics (iOPEN), Northwestern Polytechnical University, [5]School of Computing Technologies, RMIT University
{whuzsk66,swangcs,yimwang12,yushuai,peng}@whu.edu.cn,jsh.zeng@gmail.com,
yuan.sun@latrobe.edu.au,feipingnie@gmail.com,xiaodong.li@rmit.edu.au

## ABSTRACT

Traditional $k$-means minimizes the sum of squared error (SSE) but may treat data points unequally, as some are assigned to significantly distant centroids. This leads to unfair outcomes in downstream tasks such as facility location planning, where each cluster corresponds to a specific share of limited resources. To address this, we modify the objective of $k$-means via *exponential tilting*, which emphasizes the impact of distant data points and yields a new objective: the *tilted SSE*. We propose `TKM`, which optimizes via coordinate descent and stochastic gradient descent, and improves fairness by shifting centroids toward underrepresented groups. We adopt the within-cluster variance to quantify fairness among individuals within the same group, which provably reduces extreme disparities in outcomes. To improve efficiency, we propose `FastTKM`, which uses stochastic dynamics to estimate the tilted SSE with lower computational cost. We theoretically demonstrate that, under our proposed methods, the variance decreases with $t$, a scaling factor that controls the degree of centroid deviation. Furthermore, our methods exhibit time and space complexities comparable to the classical Lloyd's heuristic. Experimentally, our methods outperform six baselines in terms of clustering utility and fairness across twelve real-world datasets. In terms of efficiency, our methods achieve thousand-fold speedups in running time and reduction in memory usage, with this factor growing as the dataset size increases.

**PVLDB Reference Format:**
Shengkun Zhu, Jinshan Zeng, Yuan Sun, Sheng Wang, Yiming Wang, Yushuai Ji, Feiping Nie, Xiaodong Li, Zhiyong Peng. Highly-Efficient Large-Scale $k$-means with Individual Fairness. PVLDB, 19(5): 808 - 821, 2026.
doi:10.14778/3796195.3796197

**PVLDB Artifact Availability:**
The source code, data, and/or other artifacts have been made available at https://github.com/whu-totemdb/TKM.

## 1 INTRODUCTION

The explosive growth of data across diverse domains [29, 30] has made clustering an essential technique for extracting meaningful insights from large-scale datasets. One important application is facility location [35], which focuses on optimizing the placement of resources to improve accessibility [25]. The classic $k$-means algorithm [44], which measures similarity using Euclidean distance, is well suited for this task due to its simplicity and scalability [71].

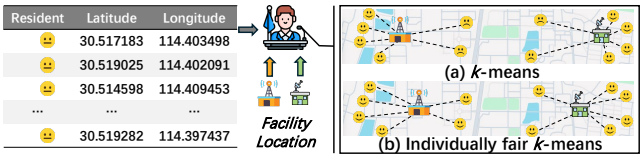


**Figure 1: A comparison between $k$-means and individually fair $k$-means in a facility location planning scenario. $k$-means results in the minority being too far from the centroid, while in individually fair $k$-means, the distance of each point to the centroid is approximately equal.**

However, directly applying $k$-means to facility location often leads to the issue of unfairness [7, 38, 40, 41, 58]. For example, an estimated 40 million Americans live in food deserts [54], which are defined as regions with low access to fresh food (e.g., people with no cars, and no grocery stores within a mile of their home) [27, 37]. Similar disparities are observed in the placement of base stations [33, 60, 62] and COVID-19 testing sites [56, 65]. As shown in Figure 1(a), when deploying public infrastructure such as base stations, $k$-means tends to place them in densely populated areas, leaving users in remote regions with weaker signal strength. This results in inequitable access to resources. Individual fairness is a promising concept that ensures each data point within the same cluster is *treated approximately equally* [16, 47]. Figure 1(b) demonstrates how a clustering method that equalizes the distances between users and base stations leads to a more equitable distribution of signal strength. We refer to this method as individually fair $k$-means.

One of the most widely studied concepts in individually fair $k$-means is the "*service in your neighborhood*" proposed by Jung et al. [38]. This concept ensures that each data point has a centroid within a small constant factor of its neighborhood radius. The neighborhood radius is defined as the minimum radius of a ball centered at each point that includes at least $n/k$ points, where $n$ is the total number of points. The rationale behind $n/k$ is that if each point had an equal chance of being selected as a facility, each point would expect a facility to be located within a neighborhood radius [49]. This concept naturally encourages placing more facilities in dense areas, where the neighborhood radius is smaller [49]. Several studies [45, 49] have improved clustering utility and yielded tighter theoretical bounds based on this individual fairness concept. Mahabadi and Vakilian [45] introduced a local search method that notably surpasses Jung et al. [38] in effectiveness. Negahbani

* Sheng Wang is the corresponding author.

Proceedings of the VLDB Endowment, Vol. 19, No. 5 ISSN 2150-8097.
doi:10.14778/3796195.3796197

and Chakrabarty [49] proposed leveraging linear programming to develop improved algorithms both theoretically and practically.

However, existing fairness definitions may face challenges in certain contexts: they ensure each point has a nearby facility but neglect within-cluster fairness, which refers to whether users assigned to the same facility receive similar service quality. For example, while methods like [38, 45, 49] guarantee facility access within a bounded radius, they permit significant disparities among users sharing the same facility. In base station deployment, users at coverage edges may experience much weaker signal strength than those near the station, despite both being served by the same facility. Existing definitions constrain only maximum distances but do not ensure equitable treatment within each service area.

Moreover, existing individually fair clustering methods suffer from the efficiency issue: their running time heavily depends on the dataset size. The most promising theoretical finding suggests a time complexity of $O(kn^4)$ [49]. Due to the high time complexity of existing algorithms, no individual fair clustering algorithm can effectively perform clustering analysis on large-scale datasets. Moreover, as the data scale increases, existing methods suffer from the issue of memory overflow since they require computation of the pairwise distances, necessitating the storage of an $n \times n$ array in memory. Additionally, in the clustering results obtained by these algorithms, each centroid must be selected from the data points, which is often unreasonable in real-world applications.

To address the within-cluster fairness issue, we propose a criterion based on the variance of squared distances in each cluster. Variance naturally quantifies treatment uniformity: small variance ensures users within a service area experience comparable service quality and reduces maximum distances, promoting balanced geometric structure. As a secondary benefit, we observe that minimizing within-cluster variance yields more balanced global facility distributions, potentially mitigating cross-regional inequalities. Drawing inspiration from *exponential tilting* [43], a technique for shifting probability distributions, we develop tilted $k$-means (`TKM`), which establishes fair $k$-means as an optimization problem using a tilted SSE objective parameterized by a scaling factor $t$. In particular, we show that when $t \to 0$, the tilted SSE reduces to the standard SSE. We use coordinate descent (CD) [73] and stochastic gradient descent (SGD) [12] for optimization, encouraging centroids to move closer to underrepresented points. To improve large-scale data efficiency, we propose `FastTKM`, which uses stochastic dynamics for gradient estimation. We theoretically show that increasing the scaling factor $t$ reduces variance, thus improving fairness. Our methods achieve time and space complexity of $O(ndk)$ and $O(nd + kd)$, comparable to classical `Lloyd's heuristic` [44] and significantly more efficient than existing fairness-aware methods. Our experiments on two non-spherical and twelve diverse application datasets show that our methods outperform SOTA methods in both effectiveness and efficiency, highlighting the broader applicability of our methods beyond facility location. Specifically, our methods achieve improved clustering utility and fairness, along with thousand-fold speedups in running time and reduced memory usage, overcoming the scalability limitations of previous methods. We validate the impact of different hyperparameters on performance and present a suitable combination of hyperparameters for practical applications.

Our contributions are summarized as follows:

**Table 1: Summary of notations**

| Notation | Description |
|---|---|
| $\mathcal{X} \coloneqq \{\boldsymbol{x}_i\}_{i=1}^n$ | The dataset of $n$ points |
| $\mathcal{S} \coloneqq \{\mathcal{S}_j\}_{j=1}^k$ | The set of $k$ clusters |
| $C \coloneqq \{\boldsymbol{c}_j\}_{j=1}^k$ | The set of $k$ centroids |
| $\overline{\psi}, \overline{\phi}$ | The SSE, tilted SSE of all clusters |
| $\psi, \phi$ | The SSE, tilted SSE of each cluster |
| Tm | The tilted mean operator |
| $d$ | The dimensionality of a data point |
| $\eta$ | The learning rate |
| $E$ | The epoch size |

- We propose a variance-based fairness metric focusing on within-cluster distance uniformity, unlike existing metrics that only consider accessibility and neglect equity within service areas.
- We propose a novel objective, tilted SSE, by incorporating exponential tilting into the standard SSE. Based on this, we optimize using CD and SGD, and propose a fair $k$-means algorithm, `TKM`.
- To improve efficiency on large-scale data, we introduce `FastTKM`, a computationally accelerated variant of `TKM`. Both methods achieve linear time and space complexity with respect to dataset size, offering superior scalability over existing methods.
- We provide a theoretical analysis of the fairness of our methods and prove that the variance of squared distances within each cluster decreases as the scaling factor $t$ increases.
- We experimentally demonstrate that our methods outperform SOTA approaches in clustering utility and fairness, achieving thousand-fold improvements in speed and memory usage.

## 2 NOTATIONS

We use different text formatting styles to represent different mathematical concepts: plain letters for scalars, bold letters for vectors, and calligraphic letters for sets. For instance, $k$ represents a scalar, $\boldsymbol{x}$ represents a vector, and $C$ denotes a set. Without loss of generality, all data points in this paper are represented using vectors. We use $[k]$ to represent the set $\{1, 2, ..., k\}$. The symbol $\mathbb{E}$ denotes the expectation of a random variable, and we use ":=" to indicate a definition. We use $\|\cdot\|$ to denote the Euclidean norm of a vector. We use the symbol "log" to denote the natural logarithm with base $e$. Table 1 lists the notations appearing in this paper and their interpretations.

## 3 RELATED WORK

**Fair Facility Location and Fair Clustering.** Inequity in the placement of critical facilities is a well-documented issue [34, 35, 40, 41, 68]. For example, an estimated 40 million Americans live in food deserts [27, 37], where remote populations (e.g., individuals without cars and no grocery stores within a mile) face limited access to fresh food. Similar disparities have also been observed in the placement of base stations [33, 60, 62] and COVID-19 testing sites [56, 65]. Recent studies in fair facility location extend classical $k$-median, $k$-means, and $k$-center formulations by embedding fairness constraints or penalty terms into the optimization objective. Gupta et al. [35] treats the $\ell_p$ objective as a fairness regularizer, where larger $p$ values impose stronger penalties on inequitable access. Wang et al. [68] studies fairness from the facility perspective, ensuring equitable agent allocation across facilities, orthogonal to our focus on equitable treatment of users within each service area.

**Table 2: Comparison of space and time complexity across different individually fair clustering methods.**

| Methods | Space complexity | Time complexity |
|---|---|---|
| **CDCS** [20] | $O((\log n)^2 + d\log n)$ | $O(kd + k^8d^4 + (k\log n)^4)$ |
| **JKL** [38] | $O(n^2 + nd)$ | $O(ndk)$ |
| **MV** [45] | $O(n^2 + nd)$ | $O(k^5n^4)$ |
| **FR** [49] | $O(n^2 + nd)$ | $O(kn^4)$ |
| **Ours** | $O(nd + kd)$ | $O(ndk)$ |

Fair clustering algorithms are typically divided into two categories: *group fairness* and *individual fairness* [14, 16, 26, 47]. Group fairness aims to cluster points with minimal SSE while ensuring balance across protected attributes. As this paper does not focus on group fairness, readers may refer to [11, 19, 21, 24, 74] for details. The concept of *individual fairness* is initially introduced by Dwork et al. [28] in the context of classification, which posits that "*similar individuals should be treated equally*". Several studies have explored this definition in clustering [13, 17]. Another widely used and researched concept of individual fairness is referred to as "*service in your neighborhood*", which is initially proposed by Jung et al. [38]. This concept aims to ensure that each point has a centroid within at most a small constant factor of its neighborhood radius, where the neighborhood radius is the minimum radius of a ball centered at the point $\boldsymbol{x}_i$ that includes at least $n/k$ data points. Subsequently, various methods addressing the individually fair $k$-clustering are based on this paradigm [20, 39, 45, 49], along with numerous improved theoretical upper bounds [36, 66]. Mahabadi and Vakilian [45] propose a local search algorithm for $k$-clustering, which significantly outperforms the method proposed by Jung et al. [38] in terms of clustering utility. Negahbani and Chakrabarty [49] propose leveraging linear programming to develop improved algorithms for individually fair $k$-clustering, both theoretically and practically.

Existing fairness metrics ensure facility accessibility by bounding the distance from each point to its nearest centroid. However, they overlook within-cluster fairness, which does not guarantee equitable treatment among users served by the same facility. Moreover, existing individually fair clustering methods encounter the same issue: they suffer from prohibitively high computational time. Specifically, the time complexity of Mahabadi and Vakilian [45] is $O(k^5n^4)$, and Negahbani and Chakrabarty [49] is $O(kn^4)$. To address this issue, Chhaya et al. [20] proposed a method to reduce the dataset size by constructing a coreset. However, this approach results in diminished clustering utility and fails to mitigate the inherent dependency of the time complexity on dataset size. Furthermore, regarding spatial complexity, these algorithms require calculating the pairwise distance between each point and storing it in memory, which consumes $O(n^2)$ memory. Table 2 compares the time and space complexity of different fair clustering methods, highlighting the efficiency of our methods in both aspects.

**Exponential Tilting.** We elucidate the concept of exponential tilting and explore its applications across various disciplines. Let $\mathcal{P} := \{p_\theta\}$ be a set of parametric distributions; for any $x$, let $I(x,\theta)$ be the information of $x$ under $\theta$ [22], which is defined as

$$I(x,\theta) := -\log p_\theta(x). \tag{1}$$

If we assume that $X$ is a random variable drawn from a distribution $p(\cdot)$, which may not match to $\mathcal{P}$, meaning that the parametric family

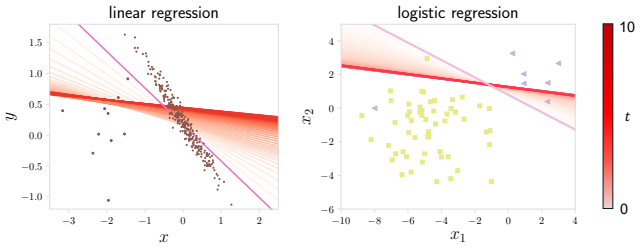


**Figure 2: Two examples of TERM [43]. Increasing the scaled factor $t$ magnifies the impact of the minority on the models.**

could be misspecified, then the cumulant generating function [23] of the information random variable $I(X,\theta)$ can be defined as

$$\mathcal{G}_X(t,\theta) := \log\Big(\mathbb{E}\Big[e^{tI(X,\theta)}\Big]\Big) = \log\sum_x p(x)p_\theta(x)^{-t}, \tag{2}$$

where $\mathbb{E}[e^{tI(X,\theta)}]$ is commonly referred to as an exponential tilting of the information density, and can induce the probability distribution with parameter $\theta$ shifting. Exponential tilting has been applied in numerous fields, such as statistics [15, 61, 64], applied probability [23], information theory [10, 48], and optimization [53, 59]. Interested readers can refer to [43] for a more detailed introduction. Currently, there are relatively few applications of exponential tilting in machine learning [43, 63, 72]. Li et al. [43] proposed tilted empirical risk minimization (TERM), which allows flexible tuning of individual losses, marking a pioneering move in machine learning. TERM offers several examples of supervised learning, including linear regression and logistic regression, as illustrated in Figure 2. Recent research has also concentrated on supervised learning, such as the additive model [72] and semantic segmentation [63].

***Challenges.*** 1) Existing fairness metrics ensure access to nearby facilities but overlook within-cluster fairness, leaving disparities among users in the same cluster. 2) The time and space complexity of existing individually fair $k$-clustering algorithms heavily depend on the dataset size. 3) Existing individually fair clustering algorithms are constrained to select centroids only from the input data points, reducing the flexibility and adaptability in continuous spaces. 4) The current application of exponential tilting is still limited to supervised learning, and it has not been applied in unsupervised learning, especially in clustering.

## 4 PRELIMINARIES

### 4.1 $k$-means

We begin by presenting $k$-means, which is a widely used clustering algorithm designed to partition a dataset into $k$ distinct clusters based on similarities among points. Let $\mathcal{X} := \{\boldsymbol{x}_i\}_{i=1}^n$ be a dataset of $n$ $d$-dimensional points, $k$-means aims to find a set $\mathcal{S} := \{\mathcal{S}_j\}_{j=1}^k$ of $k$ clusters such that the sum of squared error (SSE) is minimized,

$$\min_{\mathcal{S},C}\Big\{\overline{\psi}(\mathcal{S},C) := \sum_{j=1}^k \frac{1}{n}\sum_{\boldsymbol{x}_i\in\mathcal{S}_j} f(\boldsymbol{x}_i,\boldsymbol{c}_j)\Big\}, \tag{3}$$

where $C := \{\boldsymbol{c}_j\}_{j=1}^k$ is a set of centroids, $\boldsymbol{c}_j$ is the centroid of cluster $\mathcal{S}_j$, $f(\boldsymbol{x}_i,\boldsymbol{c}_j) := \|\boldsymbol{x}_i - \boldsymbol{c}_j\|^2$ denotes the square of the Euclidean distance from a data point $\boldsymbol{x}_i$ to the centroid $\boldsymbol{c}_j$. The commonly used method for solving $k$-means is the well-known `Lloyd's heuristic` [44], which iteratively computes the assignment of each data point

**Algorithm 1:** $k$-means++

**Input:** $\mathcal{X}$: dataset, $k$: # clusters.

1 $C \leftarrow$ Sample a point uniformly from $\mathcal{X}$;

2 $C \leftarrow$ Sample the next centroid $\boldsymbol{c}$ with probability $\frac{D(\boldsymbol{c})^2}{\sum_{\boldsymbol{c}\in\mathcal{X}} D(\boldsymbol{c})^2}$;

3 Repeat Step 2 until $k$ centroids are chosen;

/* Coordinate descent. */

4 **while** *not converge* **do**

5 Update $\Delta$ by Equation (5);

6 Update $C$ by Equation (6);

7 **return** $\Delta$ and $C$.

and the refinement of the centroids through coordinate descent (CD). Next, we provide a detailed description of the optimization process of `Lloyd's heuristic`. We begin by presenting the equivalent form of Problem (3) as follows,

$$\min_{\Delta,C}\left\{\overline{\psi}(\Delta, C) := \sum_{j=1}^{k} \psi(\boldsymbol{\delta}_j, \boldsymbol{c}_j) := \sum_{j=1}^{k} \frac{1}{n} \sum_{i=1}^{n} f(\boldsymbol{x}_i, \boldsymbol{c}_j)\delta_{ij}\right\}, \tag{4}$$

where $\delta_{ij}, i \in [n], j \in [k]$ denotes the assignment of each data point, for example, if $\boldsymbol{x}_i \in \mathcal{S}_j$, then $\delta_{ij} = 1$, else $\delta_{ij} = 0$; $\Delta := \{\boldsymbol{\delta}_j\}_{j=1}^{k}$ defines an indicator set, each $\boldsymbol{\delta}_j := (\delta_{1j}, \delta_{2j}, \cdots, \delta_{nj}) \in \mathbb{R}^n$ represents the assignment of data points in the $j$-th cluster, and $\psi(\boldsymbol{\delta}_j, \boldsymbol{c}_j) := \frac{1}{n}\sum_{i=1}^{n} f(\boldsymbol{x}_i, \boldsymbol{c}_j)\delta_{ij}$ denotes the SSE in the cluster $\mathcal{S}_j$. To solve Problem (4), one can iteratively assign each point to its nearest centroid and refine $\boldsymbol{c}_j$ using `Lloyd's heuristic`. Following initialization, with the centroid set $C$ holds constant, the solution for the indicator set $\Delta$ can be obtained as

$$\delta_{ij} = \begin{cases} 1, & j = \arg\min_l f(\boldsymbol{x}_i, \boldsymbol{c}_l), \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

When the indicator set $\Delta$ holds constant, solve for $C$ as follows:

$$\boldsymbol{c}_j = \frac{\sum_{i=1}^{n} \delta_{ij} \cdot \boldsymbol{x}_i}{\sum_{i=1}^{n} \delta_{ij}}, \tag{6}$$

As the initialization plays a critical role in the performance of the $k$-means, next we introduce the well-known $k$-means++ algorithm.

## 4.2 $k$-means++

$k$-means++ [9] is an improved version of $k$-means by providing a more effective strategy for selecting initial centroids, thus enhancing speed and accuracy. We provide the details of $k$-means++ in Algorithm 1. Its process involves selecting the first centroid randomly from the dataset (Step 1 in Algorithm 1). Let $D(\boldsymbol{x}_i)$ be the shortest distance from a data point $\boldsymbol{x}_i$ to its closest centroids that we have already chosen, then the subsequent centroid is chosen from the data points based on their squared distances to the nearest existing centroids, with a probability $\frac{D(\boldsymbol{x}_i)^2}{\sum_{\boldsymbol{x}_i\in\mathcal{X}} D(\boldsymbol{x}_i)^2}$ (Step 2 in Algorithm 1). This iterative process is repeated until $k$ centroids are chosen (Step 3 in Algorithm 1). After selecting $k$ centroids for initialization, the update of $C$ and $\Delta$ is performed through CD, which is identical to that of `Lloyd's heuristic` (Steps 4-6 in Algorithm 1).

## 4.3 Fairness in Clustering

Existing individually fair clustering methods ensure facility accessibility but overlook within-cluster fairness, as points assigned to the same facility may experience significantly unequal centroid distances, leading to inequitable service quality. Next, we provide a formal definition of fairness in this context.

**Algorithm 2:** TKM

**Input:** $\mathcal{X}$: dataset, $k$: # clusters, $E$: # epoch.

1 Initialize $\Delta$ and $C$ by $k$-means++;

2 **while** *not converge* **do**

/* Assignment. */

3 Update $\Delta$ by (5);

/* Refinement. */

4 **for** $e = 1, \cdots, E$ **do**

5 Sample a mini-batch data $\mathcal{B}$ from $\mathcal{X}$;

6 Update $C$ by (11);

7 **return** $\Delta$ and $C$.

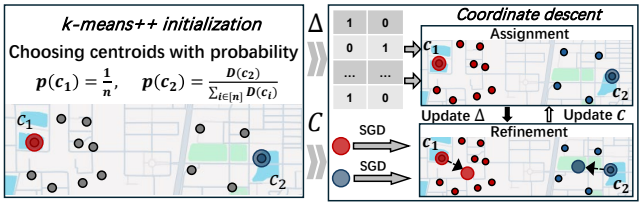


**Figure 3: A running example of TKM includes the stages of initialization, assignment, and refinement.**

**Definition 1** (Fairness). *Let $\mathcal{D} = \{\mathcal{D}_j\}_{j=1}^{k}$ and $\mathcal{D}' = \{\mathcal{D}'_j\}_{j=1}^{k}$ represent sets of squared distances generated by two clustering algorithms $\mathcal{A}$ and $\mathcal{A}'$, where $\mathcal{D}_j$ and $\mathcal{D}'_j$ represent the set of squared distances of the $j$-th cluster. We say $\mathcal{A}$ is fairer than $\mathcal{A}'$ if the distance distribution of $\mathcal{A}$ is more uniform than $\mathcal{A}'$, i.e., $\sum_{j=1}^{k} \mathrm{Var}[\mathcal{D}_j] < \sum_{j=1}^{k} \mathrm{Var}[\mathcal{D}'_j]$, where $\mathrm{Var}[\mathcal{D}_j]$ denotes the variance in the $j$-th cluster.*

Our definition focuses on within-cluster fairness, which is different from the commonly used "service in your neighborhood" criterion [38]. That criterion requires each point $\boldsymbol{x}$ to have a centroid within a radius $r(x)$, ensuring accessibility but not equitable treatment within each service area. In contrast, variance directly quantifies treatment uniformity: points in the same cluster with similar distances to the centroid receive comparable service quality.

One might aim to make all distances uniform by minimizing variance across *all points*, but clustering algorithms focus on improving intra-cluster similarity and maximizing inter-cluster dissimilarity. Minimizing global variance without considering the data's intrinsic structure could undermine this goal, as centroids may fail to reflect the true data structure. Moreover, our definition likely reduces the maximum distance, promoting a balanced geometric structure within each cluster. Let $D_{ij}$ denote the squared distance from a point in the $j$-th cluster to its centroid. A reduction in variance $\mathrm{Var}[\mathcal{D}_j]$ leads to a tighter concentration of distances around $\mathbb{E}[\mathcal{D}_j]$. By Chebyshev's inequality, $\Pr\left\{|D_{ij} - \mathbb{E}[\mathcal{D}_j]| \geq \epsilon\right\} \leq \frac{\mathrm{Var}[\mathcal{D}_j]}{\epsilon^2}$, which implies that the probability of large deviations decreases as $\mathrm{Var}[\mathcal{D}_j]$ becomes smaller. Therefore, although the maximum distance is not directly bounded by variance, a smaller $\mathrm{Var}[\mathcal{D}_j]$ increases the likelihood that the maximum distance is reduced.

# 5 PROPOSED TKM

## 5.1 Problem Formulation

Although $k$-means is widely used, it falls short in scenarios where average performance overlooks the need for equitable treatment

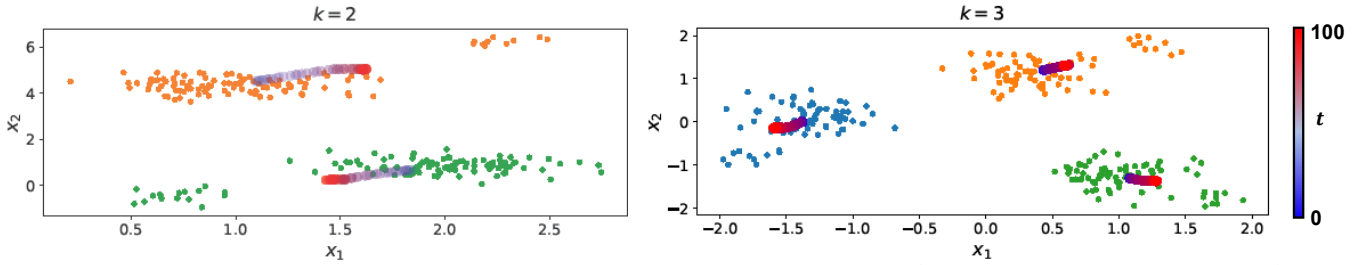


**Figure 4: Visualization of two synthetic 2-dimensional data for $k = 2$ and $k = 3$ by TKM (open-resourced in our repository). Each centroid is obtained by running TKM for 60 different $t$ within the range of $(0, 100)$. It can be observed that a larger value of $t$ can bring the centroids closer to the points with larger distances. The centroids corresponding to different $t$ are represented using a gradient from blue to red. Points within different clusters are distinguished by different colors.**

within clusters. In this section, we present a unified framework called *tilted k-means* (TKM) to address the limitations of standard $k$-means. TKM incorporates exponential tilting into the SSE, resulting in a *tilted SSE* parameterized by a real-valued scale factor $t \in \mathbb{R}^+$. This formulation allows individually fair $k$-means to be formulated as the following optimization problem:

$$\min_{\Delta, C} \left\{ \overline{\phi}(t, \Delta, C) := \sum_{j=1}^{k} \phi(t, \boldsymbol{\delta}_j, \boldsymbol{c}_j) := \sum_{j=1}^{k} \frac{1}{t} \log \frac{1}{n} \sum_{i=1}^{n} e^{t f(\boldsymbol{x}_i, \boldsymbol{c}_j) \delta_{ij}} \right\}, \quad (7)$$

where $\overline{\phi}(\cdot)$ denotes the tilted SSE aggregated over all clusters, and $\phi(\cdot)$ denotes the tilted SSE for each cluster. The tilted SSE provides flexibility in incorporating other constraints to account for other potential complex conditions in fairness clustering scenarios. We now examine the special case when $t = 0$ of the tilted SSE.

**Theorem 1** (Limit case of tilted SSE). *Consider the tilted SSE parameterized by $t \in \mathbb{R}^+$. As $t \to 0$, the following holds:*

$$\lim_{t \to 0} \overline{\phi}(t, \Delta, C) = \frac{1}{n} \sum_{j=1}^{k} \sum_{i=1}^{n} f(\boldsymbol{x}_i, \boldsymbol{c}_j) \delta_{ij}. \quad (8)$$

This result follows directly from L'Hôpital's rule, and implies that, as $t \to 0$, the tilted SSE degenerates into the standard SSE, and TKM degenerates into the standard $k$-means.

## 5.2 TKM Algorithm Design

Since Problem (7) involves a highly non-convex objective function (due to $\Delta$), we consider using CD to solve it. We begin by fixing $C$ to solve $\Delta$. Note that both the logarithmic and exponential functions are monotonically increasing with $t f(\boldsymbol{x}_i, \boldsymbol{c}_j) \geq 0$. Therefore, to minimize the objective, we would select $\delta_{ij} = 1$ for the closest centroid $\boldsymbol{c}_j$ for each data point $\boldsymbol{x}_i$, and set $\delta_{ij} = 0$ for all other centroids (the solution for $\Delta$ is identical to Equation (5)). Next, we consider fixing $\Delta$ to solve $C$. Since the tilted SSE is convex with $\boldsymbol{c}_j$, we can derive the optimality condition for the tilted SSE with $\boldsymbol{c}_j$. We present the first-order gradient of $\overline{\phi}(t, \Delta, C)$ with $\boldsymbol{c}_j$ as

$$\nabla_{\boldsymbol{c}_j} \overline{\phi}(t, \Delta, C) = \nabla_{\boldsymbol{c}_j} \phi(t, \boldsymbol{\delta}_j, \boldsymbol{c}_j) = \frac{\sum_{\boldsymbol{x}_i \in \mathcal{S}_j} e^{t f(\boldsymbol{x}_i, \boldsymbol{c}_j)} \cdot \nabla_{\boldsymbol{c}_j} f(\boldsymbol{x}_i, \boldsymbol{c}_j)}{\sum_{i=1}^{n} e^{t f(\boldsymbol{x}_i, \boldsymbol{c}_j) \delta_{ij}}}, \quad (9)$$

where $\nabla_{\boldsymbol{c}_j} f(\boldsymbol{x}_i, \boldsymbol{c}_j) = 2(\boldsymbol{c}_j - \boldsymbol{x}_i)$ is the first-order gradient of $f(\boldsymbol{x}_i, \boldsymbol{c}_j)$ with respect to $\boldsymbol{c}_j$. Then setting Equation (9) equal to zero yields the optimal condition of $\boldsymbol{c}_j$:

$$\sum_{\boldsymbol{x}_i \in \mathcal{S}_j} e^{t f(\boldsymbol{x}_i, \boldsymbol{c}_j)} (\boldsymbol{c}_j - \boldsymbol{x}_i) = 0. \quad (10)$$

Note that obtaining the closed-form solution for $\boldsymbol{c}_j$ from Equation (10) is nontrivial, therefore, we consider employing the first-order gradient method to solve $\boldsymbol{c}_j$. Let $\mathcal{B}$ be a batch data sampled from $\mathcal{X}$, then $\boldsymbol{c}_j$ is updated via mini-batch gradient descent as follows,

$$\begin{aligned} \boldsymbol{c}_j &\leftarrow \boldsymbol{c}_j - \eta \sum_{\boldsymbol{x}_i \in \mathcal{B}} w_t(\boldsymbol{x}_i) \nabla_{\boldsymbol{c}_j} f(\boldsymbol{x}_i, \boldsymbol{c}_j) \delta_{ij}, \\ w_t(\boldsymbol{x}_i) &:= \frac{e^{t f(\boldsymbol{x}_i, \boldsymbol{c}_j)}}{\sum_{i=1}^{n} e^{t f(\boldsymbol{x}_i, \boldsymbol{c}_j) \delta_{ij}}} = e^{t f(\boldsymbol{x}_i, \boldsymbol{c}_j) - t \phi_j}, \end{aligned} \quad (11)$$

where $\eta$ is a learning rate, $w_t(\boldsymbol{x}_i)$ is referred to a *tilted weight* with respect to the point $\boldsymbol{x}_i$, and $\phi_j$ is an abbreviation for $\phi(t, \boldsymbol{\delta}_j, \boldsymbol{c}_j)$.

**Algorithm Description.** The algorithmic process of TKM can be summarized into three parts: initialization, assignment, and refinement. We provide algorithm details for TKM in Algorithm 2 and an example in Figure 3. Firstly, the centroids set $C$ is initialized using $k$-means++ (Line 1 in Algorithm 2). Subsequently, we employ CD to iteratively solve $\Delta$ (assignment) and $C$ (refinement) (Lines 3-6 in Algorithm 2). We set $E$ epochs for solving $C$, where in each epoch, a batch $\mathcal{B}$ of data is sampled from $\mathcal{X}$, and the data points within $\mathcal{B}$ are used to solve $C$ using Equation (11).

To illustrate how TKM can be applied in fair clustering, we provide two toy examples, illustrated in Figure 4. As $t$ increases, each cluster's centroid shifts toward points with larger distances. This phenomenon can be explained as follows: From (11), the gradient represents a weighted average of the gradients of the original individual losses, where each data point is weighted in exponential proportion to the value of its loss. For any $t \in \mathbb{R}^+$, this exponential weighting increases the influence of points farther from the centroid, encouraging the centroid to move toward regions that reduce within-cluster distance disparities. Consequently, points within the same cluster experience more uniform distances to their centroid, ensuring more equitable service in each facility's coverage area.

**Stopping Criterion.** Since we use SGD as the solver for $C$, and it is an iterative optimization method, the algorithm will run indefinitely and waste computational resources. To address this, we propose the following stopping criterion:

- Setting a maximum number of epochs $E$, after which the algorithm terminates. However, this approach has a disadvantage as it may run for an excessive number of iterations unnecessarily.
- Monitoring the magnitude of parameter updates and stopping when the change becomes negligible, specifically when for any $j \in [k]$, we have $\|\boldsymbol{c}_j^{e+1} - \boldsymbol{c}_j^{e}\| < \gamma$, where $\boldsymbol{c}_j^{e}$ represents the $e$-th epoch value of $\boldsymbol{c}_j$, and $\gamma$ is a small threshold, e.g. $\gamma = 10^{-5}$.

By combining the two conditions mentioned above, we consider the algorithm to have converged when either of them is satisfied.

**Algorithm 3:** FastTKM

**Input:** $\mathcal{X}$: dataset, $k$: # clusters, $E$: # epoch, $\mu$: weighted factor.

1 Initialize $\Delta$ and $C$ by $k$-means++, and initialize $\{\tilde{\phi}_j\}_{j=1}^k$ by $\tilde{\phi}_j \leftarrow \frac{1}{t}\log\frac{1}{n}\sum_{i=1}^n e^{tf(\boldsymbol{x}_i,\boldsymbol{c}_j)z_{ij}}$;

2 **while** *stopping criterion not reached* **do**

/* Refinement. */

3 **for** $e = 1, \cdots, E$ **do**

4 Sample a mini-batch data $\mathcal{B}$ from $\mathcal{X}$;

5 Compute $f(\boldsymbol{x}_i, \boldsymbol{c}_j)$ and $\nabla_{\boldsymbol{c}_j} f(\boldsymbol{x}_i, \boldsymbol{c}_j)$ for all $\boldsymbol{x}_i \in \mathcal{B}$;

6 $\phi_{\mathcal{B},j} \leftarrow t$-tilted SSE on mini-batch $\mathcal{B}$;

7 $\tilde{\phi}_j \leftarrow \frac{1}{t}\log\Big((1-\mu)e^{t\tilde{\phi}_j} + \mu e^{t\phi_{\mathcal{B},j}}\Big)$;

8 $w_t(\boldsymbol{x}_i) \leftarrow e^{tf(\boldsymbol{x}_i,\boldsymbol{c}_j) - t\tilde{\phi}_j}$;

9 $\boldsymbol{c}_j \leftarrow \boldsymbol{c}_j - \frac{\eta}{|\mathcal{B}|}\sum_{\boldsymbol{x}_i\in\mathcal{B}} w_t(\boldsymbol{x}_i)\nabla_{\boldsymbol{c}_j} f(\boldsymbol{x}_i, \boldsymbol{c}_j)\delta_{ij}$;

/* Assignment. */

10 Update $\Delta$ by Equation (5);

11 **return** $\Delta$ and $C$.

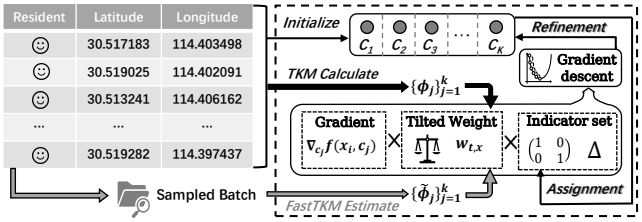


**Figure 5: Comparison of TKM and FastTKM workflows: TKM computes the exact tilted SSE, while FastTKM estimates it.**

## 5.3 Improved Version of TKM

In Algorithm 2, obtaining unbiased stochastic gradients requires calculating the tilted weights for each sample, which can be computationally expensive for large datasets. This bottleneck arises from calculating the tilted SSE, involving: 1) distance computations between points and centroids, and 2) iterative exponential/logarithmic operations. To mitigate this, we propose FastTKM, a faster variant of TKM (detailed in Algorithm 3). Specifically, we use $\{\tilde{\phi}_j\}_{j=1}^k$, which integrates stochastic dynamics $\phi_{\mathcal{B},j} := \frac{1}{t}\log\frac{1}{n}\sum_{\boldsymbol{x}_i\in\mathcal{B}} e^{tf(\boldsymbol{x}_i,\boldsymbol{c}_j)\delta_{ij}}$, to estimate the tilted SSE (Line 6 in Algorithm 3). This $\tilde{\phi}_j$ is then applied to calculate the tilted weights, as described in (11). We apply a tilted averaging of the current estimate $e^{t\phi_{\mathcal{B},j}}$ and past values $e^{t\tilde{\phi}_j}$ with a weighted factor $\mu$ to update $\phi_j$ to ensure an unbiased estimator (Line 7 in Algorithm 3). The objective of TKM can be interpreted as a function composition, specifically of $\frac{1}{n}\sum_{i=1}^n e^{tf(\boldsymbol{x}_i,\boldsymbol{c}_j)\delta_{ij}}$ and $\frac{1}{t}\log(\cdot)$. This objective can be optimized by leveraging established stochastic compositional optimization methods [32, 43, 55, 69, 70]. We maintain two sequences during the optimization process: one for the centroids $C$, and the other for the tilted SSE estimates $\{\tilde{\phi}_j\}_{j=1}^k$. This strategy is used to ensure both convergence and efficiency.

Figure 5 compares TKM and FastTKM in computing tilted weights. While TKM calculates the exact tilted SSE, FastTKM estimates it. Although one might expect FastTKM to slow convergence, Section 6 shows that FastTKM converges more rapidly. This improvement is due to the fact that adding randomness in non-convex optimization helps escape local optima and find better minima [12]. Thus, FastTKM outperforms TKM in both effectiveness and efficiency.

## 5.4 Complexity Analysis

Both TKM and FastTKM exhibit time and space complexities similar to mini-batch gradient descent, ensuring scalability for large-scale datasets. The assignment step computes pairwise distances between $n$ $d$-dimensional points and $k$ centroids, yielding $O(nkd)$ complexity. For refinement, without loss of generality, we assume that SGD iterates over each data point to update the $k$ centroids, resulting in $O(nkd)$ complexity. While FastTKM uses stochastic dynamics for gradient estimation, it retains this complexity. Space complexity is dominated by storing the $n\times d$ data matrix, $n\times k$ sparse assignment matrix $\Delta$, $k\times d$ centroid matrix $C$, a gradient matrix (scaled by batch size), and the tilted SSE scalar, leading to $O(nd + kd)$ complexity, similar to that of `Lloyd's heuristic` [44].

Moreover, FastTKM helps address the numerical issues caused by the exponential tilting operator. For example, when calculating $\sum_{i=1}^n e^{tf(\boldsymbol{x}_i,\boldsymbol{c}_j)\delta_{ij}}$, it is common to encounter situations where the result exceeds the representable range of floating-point numbers. Using the estimated tilted SSE can help avoid this issue. In addition, FastTKM is amenable to further acceleration through vectorized operations and GPU parallelization. Since the algorithm retains the additive structure of standard clustering objectives and only modifies the loss weights, it can be easily implemented within existing deep learning frameworks such as PyTorch [51].

## 5.5 Fairness Analysis

In this section, we aim to demonstrate that the clustering results generated by TKM and FastTKM reduce variance and ensure fairness. We begin by providing the definitions of tilted empirical mean and tilted empirical variance that will be used in our theory.

**Definition 2** (Tilted empirical mean and variance)**.** *Let $\{\mathcal{S}_j\}_{j=1}^k$ and $\{\boldsymbol{c}_j\}_{j=1}^k$ be the outcomes generated by Algorithm 2 or 3, let $\mathrm{f}(\mathcal{S}_j, \boldsymbol{c}_j) := \{f(\boldsymbol{x}_i, \boldsymbol{c}_j) | \boldsymbol{x}_i \in \mathcal{S}_j\}$ be a set of squared Euclidean distances of points in $\mathcal{S}_j$ to the centroid $\boldsymbol{c}_j$, then the tilted empirical mean and tilted empirical variance in cluster $\mathcal{S}_j$ are defined as*

$$\mathbb{E}_t\big(\mathrm{f}(\mathcal{S}_j, \boldsymbol{c}_j)\big) := \sum_{\boldsymbol{x}_i\in\mathcal{S}_j} w_t(\boldsymbol{x}_i)\cdot f(\boldsymbol{x}_i, \boldsymbol{c}_j), \tag{12}$$

$$\mathrm{Var}_t\big(\mathrm{f}(\mathcal{S}_j, \boldsymbol{c}_j)\big) := \mathbb{E}_t\Big(f(\boldsymbol{x}_i, \boldsymbol{c}_j) - \mathbb{E}_t\big(\mathrm{f}(\mathcal{S}_j, \boldsymbol{c}_j)\big)\Big)^2. \tag{13}$$

Note that when $t = 0$, tilted empirical mean and variance generalize to the standard mean and variance in statistics. Next, we consider the monotonicity of the tilted empirical variance with $t$.

**Theorem 2.** *Suppose the dataset is normalized. Let $\{\mathcal{S}_j\}_{j=1}^k$ be the outcomes generated by Algorithm 2 or 3, and $\{\boldsymbol{c}_j(t) = \mathrm{Tm}(t, \mathcal{S}_j)\}_{j=1}^k$ be the corresponding centroid set, where $\boldsymbol{c}_j = \mathrm{Tm}(t, \mathcal{S}_j)$ means the values of $\boldsymbol{c}_j$ satisfy (10), then for any $t \geq 0$, it holds that*

$$\frac{\partial}{\partial t}\Big\{\mathrm{Var}_\tau\Big(\mathrm{f}(\mathcal{S}_j, \boldsymbol{c}_j(t))\Big)\Big\} < 0, \tag{14}$$

*where $\tau$ is a constant in the calculation of tilted empirical variance and contributes to the tilted weight adjustment.*

The proof of Theorem 2 is provided in Section 7. Theorem 2 states that the $\tau$-tilted empirical variance will decrease with an increase in $t$. Therefore, our methods can achieve desirable clustering utility and fairness flexibly. Although Theorem 2 assumes normalized data and an exact solution for $C$ (a condition can be met given the tilted SSE's strong convexity with $C$ for fixed $\Delta$), we observe favorable

Table 3: Overview of datasets: summary of attributes and points for various datasets, with corresponding references.

| Datasets | Iris | Abalone | MNIST | Seeds | 3D-spatial | CNAD | Health | Mobile | Singapore | Census1990 | HMDA | Argoverse |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Reference** | [1] | [1] | [42] | [1] | [2] | [6] | [4] | [8] | [5] | [46] | [3] | [18] |
| **# attributes** | 4 | 8 | 784 | 8 | 4 | 12 | 8 | 31 | 9 | 69 | 53 | 2 |
| **# points** | 150 | 4177 | 60,000 | 210 | 434,874 | 16,829 | 3,756 | 38,529 | 25,293 | 2,458,285 | 5,986,660 | 12,000,000 |

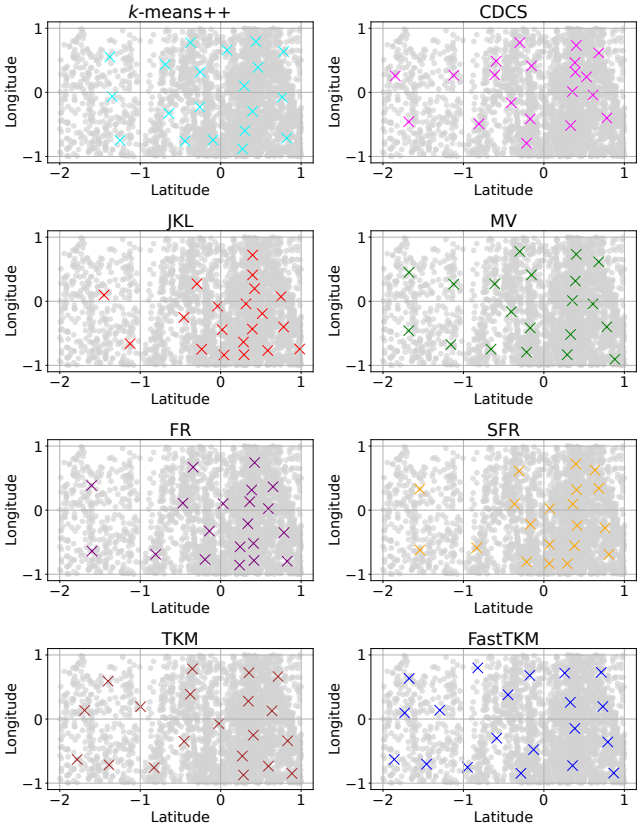


**Figure 6: Visualization of clustering results of different fairness-aware methods. `TKM` and `FastTKM` adjust centroid positions to reduce within-cluster distance variance, promoting equitable service quality for users assigned to the same facility and yielding more balanced global facility distributions.**

numerical results in Section 6, motivating the extension of these results beyond the cases that are theoretically studied.

## 6 EXPERIMENTS

**Goals.** In this section, we aim to: 1) examine the within-cluster fairness of our proposed methods; 2) evaluate the effectiveness in balancing clustering utility and fairness; 3) assess the efficiency advantages over existing approaches; and 4) analyze the influence of different hyperparameters.

### 6.1 Settings

**Datasets.** We use 12 real-world datasets to validate the performance of `TKM` and `FastTKM`. To compare the effectiveness, fairness, and convergence of our proposed methods, we conduct a comprehensive comparison with various existing methods and hyperparameter settings. The evaluation is performed using 4 labeled datasets: Iris [1], Abalone [1], MNIST [42], and Seeds [1], as well as 5 location-based

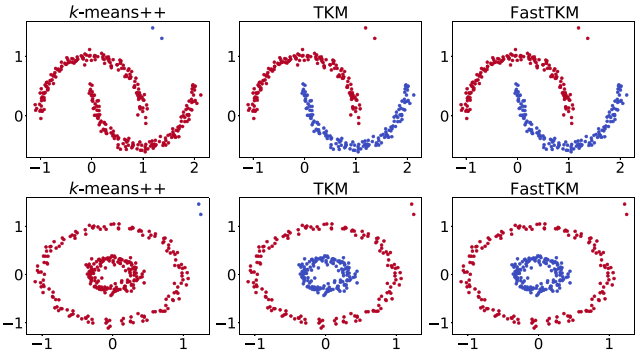


**Figure 7: Clustering results of $k$-`means++`, `TKM`, and `FastTKM` on the two-moon (top row) and three-ring (bottom row) datasets, with extra points added at cluster boundaries. $k$-`means++` incorrectly groups different moons and rings into the same cluster, while `TKM` and `FastTKM` correctly separate them by accounting for within-cluster distance disparities.**

datasets: 3D-spatial [2], CNAD [6], Health [4], Mobile [8], and Singapore [5]. To compare efficiency, we employ 3 large-scale datasets: Census1990 [46], HMDA [3], and Argoverse [18]. We extracted selected features from these datasets (detailed in our repository) and standardized them by subtracting the global mean and dividing by the global standard deviation. This preprocessing step preserves cluster assignments compared to using raw data for clustering. An overview of the datasets is shown in Table 3.

**Baselines.** We experimentally evaluate the performance of `TKM` and `FastTKM` against six baselines, namely, $k$-`means++` [9], `JKL` [38], `MV` [45], `FR` [49], `SFR` [49], and `CDCS` [20]. As explained in our related work, `JKL` first introduced the concept of individually fair $k$-means. `MV`, `FR` and `SFR` are three SOTA methods for individually fair $k$-means, where `SFR` is a sparse version of `FR`. `CDCS` is a method that constructs a coreset over the entire dataset before applying `MV`. $k$-`means++` is one of the most classical clustering methods that does not take individual fairness into account.

**Measurements.** We employ several metrics to evaluate the performance of clustering algorithms. To evaluate utility, we use the *SSE*, where lower values indicate superior clustering performance, and the *Adjusted Rand Index (ARI)* [31, 57], which assesses the agreement between predicted and true cluster assignments. ARI scores range from 0 (random agreement) to 1 (perfect match), with higher values indicating greater concordance with the ground truth. To measure fairness among different clustering algorithms, we consider using three metrics. The first is the *variance* of each point's squared distance to the centroid within each cluster. A smaller variance indicates a fairer algorithm. The second metric is the *maximum distance (Max_D)* from each point in a cluster to the centroid, where a smaller maximum distance signifies greater fairness. Finally, we adopt the fairness criterion from [45], which defines, for any centroid set $C$, a violation score $\theta_C(\boldsymbol{x}) := \frac{d(\boldsymbol{x},C)}{r(\boldsymbol{x})}$, where

**Table 4: Comparative performance of various methods across four labeled datasets. The best and the second-best performance for each metric is highlighted. Statistically significant differences with the best method are denoted by $^{**}$ for $p < 0.01$ and $^{*}$ for $p < 0.05$. Arrow annotations indicate the desired performance trend: ↑ means higher is better, and ↓ means lower is better.**

| Datasets | Methods | SSE ↓ | Variance ↓ | ARI ↑ | Max_V ↓ | Max_D ↓ |
|---|---|---|---|---|---|---|
| **Iris** | JKL | 0.1653 ± 0.0001 $^{**}$ | 0.0253 ± 0.0001 $^{**}$ | 0.3599 ± 0.0001 $^{**}$ | 0.9133 ± 0.0002 $^{**}$ | 0.8695 ± 0.0000 $^{**}$ |
| | MV | 0.1458 ± 0.0018 $^{**}$ | 0.0210 ± 0.0043 $^{*}$ | 0.3671 ± 0.0267 $^{**}$ | 0.9809 ± 0.0449 $^{**}$ | 0.7489 ± 0.0401 $^{**}$ |
| | FR | 0.1436 ± 0.0000 $^{**}$ | 0.0237 ± 0.0001 $^{**}$ | 0.4056 ± 0.0002 $^{**}$ | 0.9346 ± 0.0001 $^{**}$ | 0.7747 ± 0.0000 $^{**}$ |
| | SFR | 0.1653 ± 0.0001 $^{**}$ | 0.0253 ± 0.0001 $^{**}$ | 0.3599 ± 0.0001 $^{**}$ | 0.9065 ± 0.0004 $^{*}$ | 0.8695 ± 0.0000 $^{**}$ |
| | CDCS | 0.1519 ± 0.0011 $^{**}$ | 0.0231 ± 0.0024 $^{**}$ | 0.3455 ± 0.0103 $^{**}$ | 0.9972 ± 0.0341 $^{**}$ | 0.8011 ± 0.0121 $^{**}$ |
| | TKM | 0.1352 ± 0.0001 $^{**}$ | 0.0204 ± 0.0005 | 0.7177 ± 0.0042 $^{*}$ | 0.9050 ± 0.0079 | 0.6425 ± 0.0066 |
| | FastTKM | **0.1273 ± 0.0001** | **0.0201 ± 0.0006** | **0.7219 ± 0.0068** | **0.9016 ± 0.0030** | **0.6418 ± 0.0098** |
| **Abalone** | JKL | 0.4798 ± 0.0429 $^{**}$ | 0.1312 ± 0.0169 $^{**}$ | 0.0060 ± 0.0050 $^{**}$ | 0.9983 ± 0.0011 $^{*}$ | 1.7714 ± 0.2246 $^{**}$ |
| | MV | 0.2239 ± 0.0152 $^{**}$ | 0.0531 ± 0.0095 $^{**}$ | 0.0539 ± 0.0151 $^{**}$ | 1.0843 ± 0.0170 $^{**}$ | 1.4502 ± 0.2310 $^{**}$ |
| | FR | 0.2201 ± 0.0136 $^{**}$ | 0.0494 ± 0.0051 $^{**}$ | 0.0546 ± 0.0108 $^{**}$ | 0.9933 ± 0.0077 $^{*}$ | 1.4200 ± 0.1845 $^{**}$ |
| | SFR | 0.2364 ± 0.0233 $^{**}$ | 0.0569 ± 0.0136 $^{**}$ | 0.0554 ± 0.0170 $^{**}$ | **0.9876 ± 0.0154** | 1.5081 ± 0.2502 $^{**}$ |
| | CDCS | 0.2511 ± 0.0217 $^{**}$ | 0.0544 ± 0.0120 $^{**}$ | 0.0504 ± 0.0128 $^{**}$ | 1.0109 ± 0.0239 $^{**}$ | 1.6925 ± 0.1441 $^{**}$ |
| | TKM | 0.2275 ± 0.0191 $^{**}$ | 0.0313 ± 0.0135 $^{**}$ | **0.1245 ± 0.0137** | 1.0047 ± 0.0104 $^{**}$ | **1.2314 ± 0.2052** |
| | FastTKM | **0.2092 ± 0.0118** | **0.0274 ± 0.0181** | 0.1238 ± 0.0122 $^{*}$ | 1.0868 ± 0.0115 $^{**}$ | 1.2463 ± 0.3019 $^{*}$ |
| **MNIST** | JKL | 639.34 ± 12.408 $^{**}$ | 19.679 ± 2.8506 $^{**}$ | 0.0592 ± 0.0095 $^{**}$ | 1.0473 ± 0.0087 $^{**}$ | 34.4850 ± 2.0306 $^{**}$ |
| | MV | 640.47 ± 9.3020 $^{**}$ | 21.790 ± 4.3753 $^{**}$ | 0.0658 ± 0.0187 $^{**}$ | 1.0768 ± 0.0162 $^{**}$ | 34.6256 ± 0.6777 $^{**}$ |
| | FR | 597.22 ± 7.5347 $^{**}$ | 25.644 ± 2.9417 $^{**}$ | 0.0806 ± 0.0127 $^{**}$ | 1.0630 ± 0.0132 $^{**}$ | 34.3497 ± 0.6301 $^{**}$ |
| | SFR | 597.22 ± 7.1273 $^{**}$ | 25.726 ± 2.9957 $^{**}$ | 0.0788 ± 0.0132 $^{**}$ | 1.0587 ± 0.0140 $^{**}$ | 34.4780 ± 0.6598 $^{**}$ |
| | CDCS | 652.63 ± 12.106 $^{**}$ | 20.360 ± 3.1949 $^{**}$ | 0.0544 ± 0.0176 $^{**}$ | 1.0741 ± 0.0115 $^{**}$ | 34.3144 ± 0.6483 $^{**}$ |
| | TKM | 496.16 ± 5.9496 $^{**}$ | **1.1715 ± 0.1617** | 0.2468 ± 0.0252 $^{**}$ | 1.0143 ± 0.0309 $^{**}$ | **24.6511 ± 0.2523** |
| | FastTKM | **400.33 ± 4.6801** | 2.6745 ± 0.9723 $^{**}$ | **0.3521 ± 0.0407** | **0.8575 ± 0.0078** | 27.3265 ± 0.7360 $^{**}$ |
| **Seeds** | JKL | 0.1294 ± 0.0001 $^{**}$ | 0.0180 ± 0.0001 $^{**}$ | 0.1870 ± 0.0001 $^{**}$ | 0.9983 ± 0.0000 $^{**}$ | 0.6443 ± 0.0001 $^{**}$ |
| | MV | 0.1000 ± 0.0012 $^{**}$ | 0.0173 ± 0.0008 $^{**}$ | 0.3002 ± 0.0196 $^{**}$ | 0.9697 ± 0.0164 $^{**}$ | 0.6136 ± 0.0382 $^{**}$ |
| | FR | 0.0971 ± 0.0002 $^{*}$ | 0.0171 ± 0.0001 $^{**}$ | 0.3199 ± 0.0001 $^{**}$ | 0.9729 ± 0.0001 $^{**}$ | 0.5834 ± 0.0001 $^{**}$ |
| | SFR | 0.1002 ± 0.0001 $^{**}$ | 0.0171 ± 0.0001 $^{**}$ | 0.2912 ± 0.0002 $^{**}$ | 1.0217 ± 0.0001 $^{**}$ | 0.6218 ± 0.0001 $^{**}$ |
| | CDCS | 0.1365 ± 0.0047 $^{**}$ | 0.0182 ± 0.0026 $^{**}$ | 0.2115 ± 0.0351 $^{**}$ | 0.9754 ± 0.0233 $^{**}$ | 0.6243 ± 0.0276 $^{**}$ |
| | TKM | **0.0963 ± 0.0001** | **0.0161 ± 0.0008** | 0.7361 ± 0.0083 | **0.9374 ± 0.0080** | **0.5479 ± 0.0090** |
| | FastTKM | 0.0964 ± 0.0001 | 0.0163 ± 0.0002 | **0.7403 ± 0.0000** | 0.9420 ± 0.0114 $^{*}$ | 0.5481 ± 0.0069 |

$d(\boldsymbol{x}, C)$ denotes the distance from point $\boldsymbol{x}$ to its assigned centroid, and $r(\boldsymbol{x})$ is the smallest radius of a ball centered at $\boldsymbol{x}$ containing at least $n/k$ points. The *maximum violation (Max_V)*, defined as $\max_{\boldsymbol{x}\in\mathcal{X}} \theta_C(\boldsymbol{x})$, serves as the third metric. For efficiency evaluation, we assess the *running time* and *memory usage* of each algorithm. To verify convergence, we use *tilted SSE* as the metric.

**Implementations.** All algorithms were executed on a CentOS 7 platform with an Intel i9-14900KF CPU and 64 GB RAM. The implementations, including our methods and the baselines (provided by [49] and [20]), were realized in Python 3.7 and open-sourced.

## 6.2 Comparison among Various Methods

*6.2.1 Visualization of Clustering.* We visually demonstrate the effectiveness of our methods on both spherical and non-spherical datasets. Figure 6 presents a visualization of the clustering results on the standardized CNAD dataset, comparing TKM and FastTKM with baselines $k$-means++, CDCS, JKL, MV, FR, and SFR. We sampled 5,000 points from the CNAD dataset, selecting longitude and latitude as features. Specifically, 30% of the points are sampled from the latitude range $[-2, -0.5]$ as sparse areas, and 70% from $[-0.5, 1]$ as dense areas. The number of clusters is set to $k = 20$. For TKM and FastTKM, the hyperparameters are configured as $t = 10$, 5 epochs, 500 iterations, and a batch size of 50. The hyperparameters for JKL, MV, FR, and SFR are set to their default values in their papers. Specifically, for JKL, the initial parameters are set to $low = 1$ and $high = 2$. For MV and CDCS, we set $dilation = 3$ and $ratio = 2$; for FR and SFR, we set $dilation = 2$ and $ratio = 2$, where $dilation$ and $ratio$ represent algorithm parameters for filter and radius calculation, respectively, initialized according to the settings in their original papers. For CDCS, the coreset sampling size is set to 10%, following the configuration to balance clustering utility, fairness, and efficiency. For SFR, the sparsification parameter is set to $\delta = 0.3$.

Figure 7 compares clustering results on the non-spherical datasets: two-moon and three-ring datasets, generated using `scikit-learn` [52] with additional points added to simulate sparse regions. Each dataset is transformed by first computing a radial basis function (RBF) kernel similarity matrix, then constructing the normalized Laplacian matrix [67]. Spectral embedding is then performed by extracting the first two eigenvectors of the Laplacian and normalizing each row to obtain a lower-dimensional representation [50]. Clustering is then applied using $k$-means++, TKM, and FastTKM. Both datasets consist of 300 data points, with the number of clusters set to $k = 2$. For TKM and FastTKM, hyperparameters are configured as $t = 10$, 5 epochs, 500 iterations, and a batch size of 20.

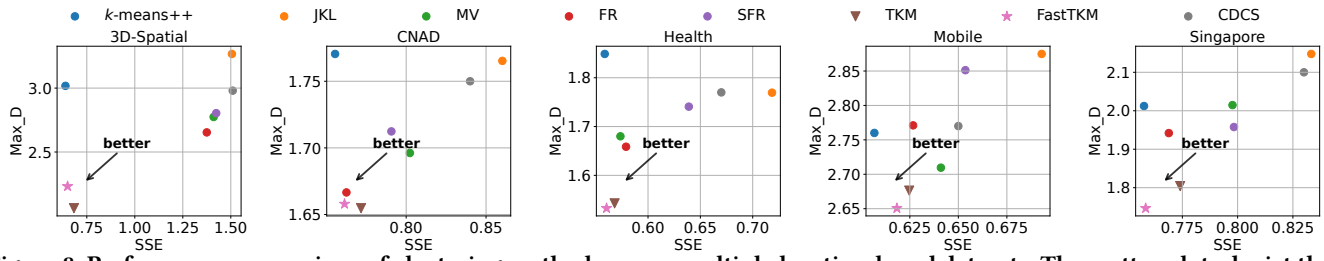


**Figure 8: Performance comparison of clustering methods across multiple location-based datasets. The scatter plots depict the relationship between SSE and Max_D for various clustering algorithms. The direction of the arrow indicates the optimal region.**

***Observations from spherical datasets.*** As shown in Figure 6, our methods adjust centroid positions to reduce variance. Existing methods (JKL, MV, FR, SFR, CDCS) ensure facility accessibility but permit distance disparities among points within the same cluster. In contrast, TKM and FastTKM optimize for within-cluster uniformity, ensuring more equitable service quality for users assigned to the same facility. This within-cluster optimization also yields more balanced global facility distributions, as shown by the more spatially dispersed centroids in TKM and FastTKM compared to other methods.
***Observations from non-spherical datasets.*** Figure 7 shows the results on non-spherical datasets containing several boundary points. $k$-means++ mispartitions the data, merging distinct moons or rings and isolating boundary points. In contrast, TKM and FastTKM leverage exponential tilting to reduce within-cluster variance, amplifying the influence of boundary points and preventing misassignments. Both methods accurately recover the true cluster structure. These findings highlight that optimizing within-cluster fairness enhances robustness to boundary cases and complex geometries.

*6.2.2 Effectiveness Analysis.* Table 4 presents an overall performance comparison of various methods across four labeled datasets, evaluating metrics such as SSE, variance, ARI, Max_V, and Max_D. For variance and Max_D, we report the average value across clusters. Figure 8 presents a trade-off comparison across five location-based datasets, examining the relationship between SSE and Max_D. SSE is a key indicator of utility in a facility location scenario, with lower values indicating smaller distances between all users and the facility. In contrast, Max_D measures fairness, with lower values indicating a smaller maximum distance between users and the facility. Due to the computational complexity of the baselines, we conduct experiments on sampled datasets for scalability. For Abalone, MNIST, 3D-spatial, CNAD, Health, Mobile, and Singapore, we generate 20 subsets, each containing 1,000 points, and report the averaged metrics. For Iris and Seeds, we use the full data but create 20 shuffled variants to ensure a robust evaluation. For TKM and FastTKM, hyperparameters are configured as $t = 0.1$, 5 epochs, 500 iterations, and a batch size of 50. The hyperparameters for CDCS, JKL, MV, FR, and SFR follow the same configuration as in Section 6.2.1.
***Observations from labeled datasets.*** TKM and FastTKM consistently show strong performance, particularly in clustering utility metrics such as SSE and ARI. In the Iris dataset, FastTKM achieves the lowest SSE (0.1273 ± 0.0001) and highest ARI (0.7219 ± 0.0068), while TKM follows closely with SSE of 0.1352 ± 0.0001 and ARI of 0.7177 ± 0.0042. Both methods outperform the others in these clustering-related metrics, indicating their superior suitability for clustering tasks. Furthermore, they demonstrate competitive results in fairness-related metrics, with both TKM and FastTKM exhibiting lower variance, Max_V, and Max_D compared to other methods. For example, in the Iris dataset, FastTKM achieves the lowest variance of 0.0201 ± 0.0006, the lowest Max_V of 0.9016 ± 0.0030, and the lowest Max_D (0.6418 ± 0.0098), showing the advantages in fairness. In the Abalone dataset, FR and SFR achieve the smallest Max_V. However, except for this metric, TKM and FastTKM show a significant advantage in all other metrics. Specifically, FastTKM and TKM overall achieve top-2 performances in terms of SSE, variance, ARI, and Max_D. For the MNIST and Seeds datasets, both TKM and FastTKM achieve the top-2 performance in all metrics. The performance differences in the SSE and ARI are notably significant, demonstrating that both methods excel in clustering utility. Moreover, in fairness-related metrics, TKM and FastTKM stand out with considerably lower variance, Max_V, and Max_D values compared to other methods. By comparing TKM and FastTKM, we observe that FastTKM achieves better performance in most cases, which is an interesting phenomenon. This can be attributed to the fact that the randomness introduced in the gradient computation helps the algorithm escape local optima, thus leading to better performance.
***Observations from location-based datasets.*** Among the methods evaluated, $k$-means++ exhibits the best clustering utility, achieving the lowest SSE values across all datasets. However, this advantage comes at the cost of fairness, as $k$-means++ shows high Max_D values, indicating poor fairness. In contrast, TKM and FastTKM manage to strike a balance between both metrics. Our methods demonstrate excellent clustering utility while minimizing Max_D, achieving both low SSE and Max_D values across the datasets. By comparing FastTKM and TKM, we observe that FastTKM demonstrates a better trade-off between SSE and Max_D in most cases. This is due to FastTKM 's use of stochastic dynamics in gradient estimation, which introduces randomness that helps escaping the local optima.
*6.2.3 Efficiency Analysis.* Figure 9 provides a comparative analysis of the running time and memory usage between TKM, FastTKM, and 5 baselines: $k$-means++, MV, CDCS, FR, and SFR (Due to the poor performance of JKL, we do not consider this method in the efficiency comparison). For fair comparison with $k$-means++, both TKM and FastTKM employ random initialization. We sample Census1990, HMDA, and Argoverse with sizes $n_s$ of 1K, 2K, 5K, 10K, 20K, 50K, 90K, 200K, 2M, 5M, and 10M, where the choice of sample sizes is based on our testing, which showed that these specific sample sizes correspond to the point at which certain algorithms exceed the preset time limits or experience memory overflow. We set the number of iterations for TKM and FastTKM to 500, the batch size to $\frac{1}{50}n_s$, the epoch size to 5, and the learning rate to 0.05. The hyperparameters of CDCS, MV, FR, and SFR are configured same with Section 6.2.1.

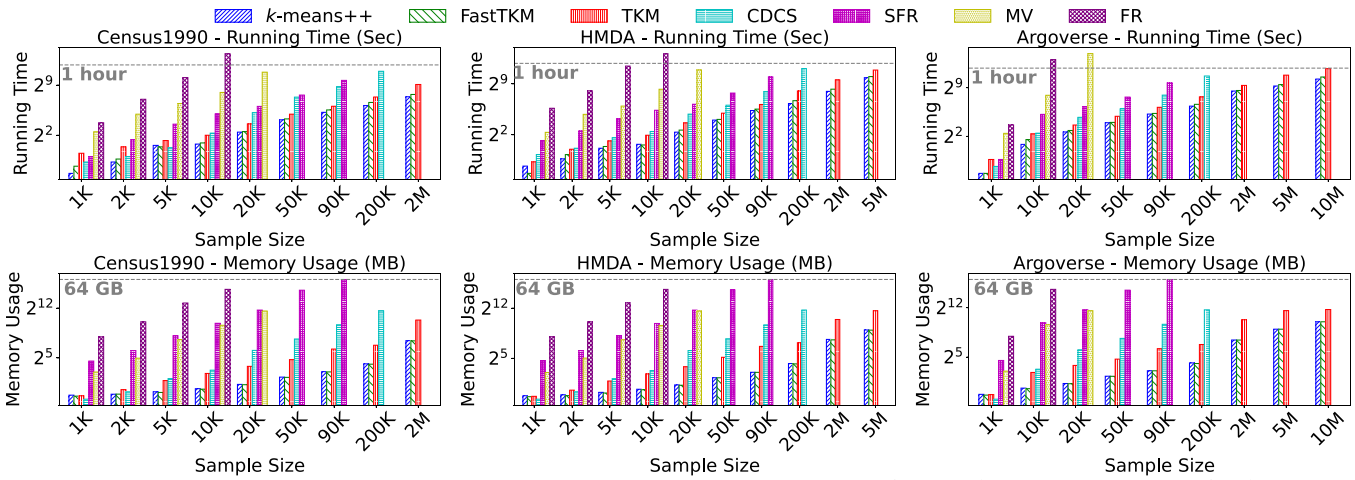


**Figure 9: Comparison among different methods in terms of running time (seconds) and memory usage (MB).**

***Observations***. As shown in Figure 9, regardless of the number of data points sampled, the running time of TKM and FastTKM is consistently shorter than that of MV, FR, and SFR. Due to the use of coreset in CDCS, it exhibits shorter running time when the dataset size is small (less than 5,000). However, as the dataset size increases to 10,000, the required time for CDCS becomes considerably longer than that of TKM and FastTKM. $k$-means++ does not consider individual fairness, which results in a shorter running time compared to TKM. However, FastTKM and $k$-means++ exhibit nearly identical running time, highlighting the remarkable efficiency advantage of FastTKM. When comparing FastTKM and TKM, FastTKM consistently requires less running time than TKM. All algorithms were implemented in Python 3.7, with TKM and FastTKM demonstrating significantly higher scalability than MV, FR, SFR, and CDCS by efficiently processing million-scale datasets per hour. Moreover, as the dataset size increases, the running time and memory efficiency of TKM and FastTKM become hundreds to thousands of times superior to those of MV and FR. While CDCS and SFR run significantly faster and consume less memory than MV and FR, both TKM and FastTKM achieve substantial acceleration in terms of both running time and memory usage over SFR and CDCS as the sample size increases. Furthermore, when the sample size reaches 90,000, the algorithmic characteristic of SFR, which requires computing distances between each sample point, can lead to a memory overflow issue, causing the algorithm to terminate. This issue also arises in MV and FR.

*6.2.4 Summary of Lessons Learned.* We present visualizations of clustering results, compare the effectiveness and efficiency across multiple methods. Our results lead to the following insights:

- TKM and FastTKM effectively improve within-cluster fairness by reducing distance variance within each cluster through exponential tilting, which amplifies the influence of points farther from centroids. This optimization potentially yields more balanced global facility distributions and demonstrates robustness in handling boundary cases in challenging non-spherical datasets.
- TKM and FastTKM outperform SOTA methods in effectiveness, demonstrating superior performance across multiple clustering utility and fairness metrics on both labeled and location-based datasets. Notably, FastTKM achieves better results in most cases than TKM due to the use of stochastic dynamics in gradient estimation, which introduces randomness to help escape local optima.
- TKM and FastTKM surpass SOTA methods in terms of efficiency. Specifically, TKM and FastTKM can cluster more data points in a shorter running time, and as the sample size increases, this acceleration effect becomes even more pronounced. Moreover, our methods can overcome the memory overflow issue that existing methods encounter when dealing with large-scale data.

## 6.3 Comparison among Various Parameters

*6.3.1 Tilted SSE vs. $t$.* Figure 10 illustrates the convergence of TKM and FastTKM for tilted SSE across iterations at $t$ values of 0.01, 0.05, and 0.1. We use five location-based datasets: 3D-spatial, Health, Mobile, Singapore, and CNAD. We randomly select 1,000 data points for each dataset and repeat this process 20 times. We conduct experiments on these 20 subsampled datasets, calculating the average of the resulting tilted SSE. For other hyperparameters, we set the learning rate to 0.05, the number of iterations to 500, the batch size to 50, the epoch size to 5, and $k = 3$.

***Observations***. First, we observe that despite using SGD to update the centroids, the tilted SSE of both TKM and FastTKM decreases steadily with iterations, which confirms the convergence of our methods. Secondly, we observe that FastTKM exhibits better convergence than TKM. This is because FastTKM estimates the gradient using stochastic dynamics, which introduces more randomness compared to TKM. Since the objective of TKM is highly non-convex, this increased randomness helps the algorithm more effectively escape local optima and find better local minima. Third, as $t$ increases, the tilted SSE also increases monotonically. Readers can refer to [43] for the theoretical analysis of this monotonicity.

*6.3.2 Tilted SSE vs. Epoch.* Figure 11 illustrates the convergence of tilted SSE with different epoch sizes during iterations. The data preprocessing here follows the same procedures outlined in Section 6.3.1. To visualize the curve of tilted SSE over iterations intuitively, we set $t = 0.02$, $k = 3$, learning rate to 0.05, number of iterations to 500, batch size to 50, and epoch size to 1, 3, 5.

***Observations***. From Figure 11, it can be observed that as the number of iterations increases, the tilted SSE of TKM and FastTKM decreases and tends to stabilize after reaching a certain value on all datasets. With an increase in the epoch size, the convergence speed of TKM and FastTKM accelerates, and its convergence performance improves. This is because increasing the epoch size allows

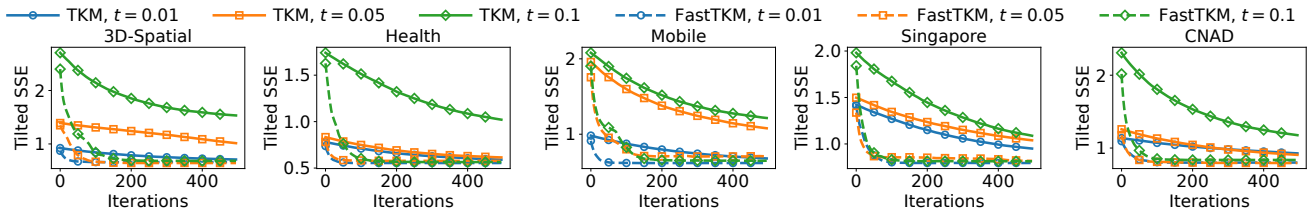


**Figure 10: Effect of the scaled factor $t$ on the convergence of TKM and FastTKM.**

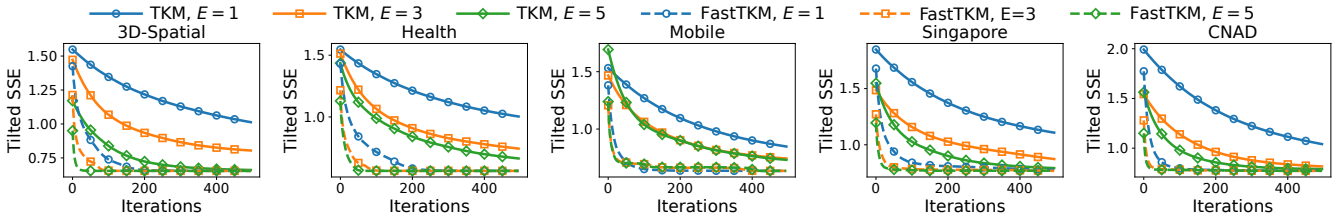


**Figure 11: Effect of the epoch on the convergence of TKM and FastTKM.**

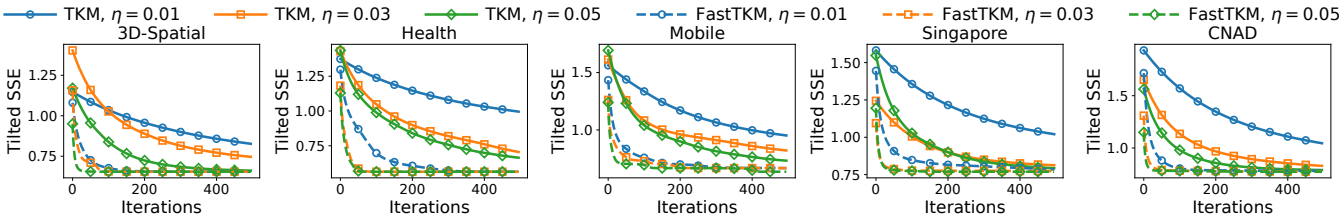


**Figure 12: Effect of the learning rate on the convergence of TKM and FastTKM.**

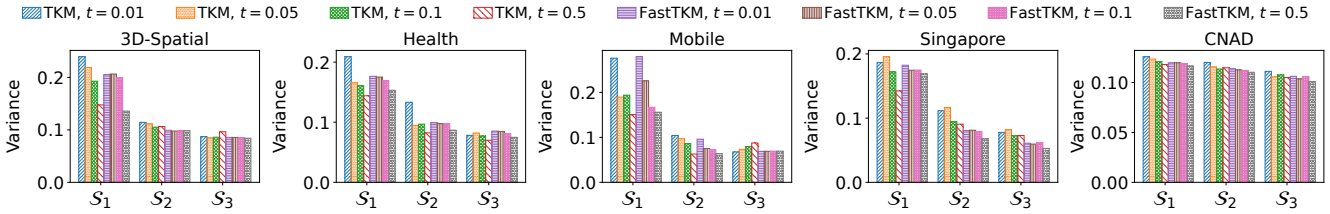


**Figure 13: Effect of the scaled factor $t$ on the variance in each cluster.**

for higher precision in the solution obtained through SGD during each iteration, as more data can be used to update $C$. However, in some datasets, we found that increasing the epoch size does not necessarily improve convergence. For example, in Mobile, an epoch size of 3 or 5 has little impact on convergence. This can be attributed to the fact that with an epoch size of 3, the solution has already converged to a local optimum, and additional epochs do not further improve the accuracy of the solution.

*6.3.3 Tilted SSE vs. Learning Rate.* Figure 12 illustrates the convergence of the tilted SSE across iterations with different learning rates. The data preprocessing steps are the same as those described in Section 6.3.1. For the hyperparameter settings, we set $t = 0.02$, the epoch size to 5, the batch size to 50, the number of iterations to 500, and the learning rates to 0.01, 0.03, and 0.05.

***Observations***. From Figure 12, we can see that, across all datasets, the convergence speed generally increases with the increase in learning rate. However, when the learning rate increases to a certain extent, the increase in convergence speed becomes slower. For example, when $\eta = 0.03, 0.05$, the converged tilted SSE value on the Singapore is almost indistinguishable. Therefore, selecting $\eta = 0.05$ is a suitable choice for most datasets.

*6.3.4 Variance vs. t.* Figure 13 illustrates the variance within each cluster for TKM and FastTKM across five location-based dataset when $t = 0.01, 0.05, 0.1, 0.5$. The variances within clusters 1-3 are sorted in descending order. The data processing and hyperparameter configurations are consistent with those in Section 6.2.2.

***Observations***. As $t$ increases, both TKM and FastTKM exhibit a decrease in variance, confirming the theoretical findings presented in Theorem 2. This indicates that even when the dataset does not satisfy the assumptions of normalization and the existence of an exact solution for $C$, the numerical results motivate the extension of these theoretical findings to broader, more practical scenarios. In most cases, FastTKM consistently outperforms TKM, showing smaller variance values. This confirms that the stochastic dynamics introduced in FastTKM contribute to better fairness.

*6.3.5 Summary of Lessons Learned.* We have evaluated the convergence and fairness of TKM and FastTKM under varying hyperparameters. The results lead to the following insights:

- TKM and FastTKM are convergent algorithms, and the tilted SSE increases monotonically with $t$. Specifically, for different values of $t$, the tilted SSE steadily decreases to a stable value. Moreover, as $t$ increases, the tilted SSE increases.

- The convergence of TKM and FastTKM is influenced by the epoch size and learning rate. Specifically, selecting an appropriate epoch size and learning rate can lead to faster and better convergence. However, choosing excessively large epoch sizes or learning rates does not necessarily improve performance.
- Although the experimental setting extends beyond the assumptions made in the theory, both TKM and FastTKM exhibit a decreasing variance with increasing $t$, aligning well with the theoretical results. This suggests that TKM and FastTKM generalize effectively beyond the conditions formally studied.

## 7 PROOF OF THEOREM 2

We begin by defining the tilted weight, tilted empirical mean, and tilted empirical variance when all data points are normalized.

**Definition 3** (Tilted weight)**.** *Suppose the dataset is normalized, then the tilted weight is defined as follows,*

$$w_i(t, \mathcal{S}_j, \boldsymbol{c}_j) := \frac{e^{t\|\boldsymbol{x}_i - \boldsymbol{c}_j\|^2}}{\sum_{\boldsymbol{x}_i \in \mathcal{S}_j} e^{t\|\boldsymbol{x}_i - \boldsymbol{c}_j\|^2}} = \frac{1}{|\mathcal{S}_j|} e^{-2t\boldsymbol{c}_j^\top \boldsymbol{x}_i - \Gamma(t, \mathcal{S}_j, \boldsymbol{c}_j)},$$

*where* $\Gamma(t, \mathcal{S}_j, \boldsymbol{c}_j) := \log \frac{1}{|\mathcal{S}_j|} \sum_{\boldsymbol{x}_i \in \mathcal{S}_j} e^{-2t\boldsymbol{c}_j^\top \boldsymbol{x}_i}$.

**Definition 4** (Tilted empirical mean and variance)**.** *Suppose the dataset is normalized, the tilted empirical mean and tilted empirical variance in each cluster are defined as*

$$\mathbb{E}_t\Big(\mathrm{f}(\mathcal{S}_j, \boldsymbol{c}_j)\Big) := \|\boldsymbol{c}_j\|^2 + \sum_{\boldsymbol{x}_i \in \mathcal{S}_j} w_i(t, \mathcal{S}_j, \boldsymbol{c}_j)\|\boldsymbol{x}_i\|^2 - \boldsymbol{c}_j^\top M(t, \mathcal{S}_j, \boldsymbol{c}_j),$$

$$\mathrm{Var}_t\Big(\mathrm{f}(\mathcal{S}_j, \boldsymbol{c}_j)\Big) := \mathbb{E}_t\Big(\boldsymbol{c}_j^\top\big(-2\boldsymbol{x}_i - M(t, \mathcal{S}_j, \boldsymbol{c}_j)\big)\Big)^2 = \boldsymbol{c}_j^\top V(t, \mathcal{S}_j, \boldsymbol{c}_j)\boldsymbol{c}_j,$$

*where* $M(t, \mathcal{S}_j, \boldsymbol{c}_j) := \sum_{\boldsymbol{x}_i \in \mathcal{S}_j} 2w_i(t, \mathcal{S}_j, \boldsymbol{c}_j)\boldsymbol{x}_i$*, and*

$$V(t, \mathcal{S}_j, \boldsymbol{c}_j) := \mathbb{E}_t\big(-2\boldsymbol{x}_i - M(t, \mathcal{S}_j, \boldsymbol{c}_j)\big)^\top\big(-2\boldsymbol{x}_i - M(t, \mathcal{S}_j, \boldsymbol{c}_j)\big)$$
$$= \sum_{\boldsymbol{x}_i \in \mathcal{S}_j} w_i(t, \mathcal{S}_j, \boldsymbol{c}_j)\big(-2\boldsymbol{x}_i - M(t, \mathcal{S}_j, \boldsymbol{c}_j)\big)^\top\big(-2\boldsymbol{x}_i - M(t, \mathcal{S}_j, \boldsymbol{c}_j)\big).$$

To establish the results in Theorem 2, we need the partial derivatives of $M(t, \mathcal{S}_j, \boldsymbol{c}_j)$ and $\Gamma(t, \mathcal{S}_j, \boldsymbol{c}_j)$ with respect to $t$ and $\boldsymbol{c}_j$.

**Lemma 1** (Partial derivatives of $M(t, \mathcal{S}_j, \boldsymbol{c}_j)$)**.** *For any* $t \in \mathbb{R}^+$*, and any* $\boldsymbol{c}_j \in \mathbb{R}^d$*, the following results hold:*

$$\frac{\partial}{\partial t} M(t, \mathcal{S}_j, \boldsymbol{c}_j) = -V(t, \mathcal{S}_j, \boldsymbol{c}_j)\boldsymbol{c}_j, \tag{15}$$
$$\nabla_{\boldsymbol{c}_j} M(t, \mathcal{S}_j, \boldsymbol{c}_j) = -tV(t, \mathcal{S}_j, \boldsymbol{c}_j), \tag{16}$$

Lemma 1 can be trivially established through differentiation. Building on the above, we proceed to prove Theorem 2.

Proof of Theorem 2. Let $\boldsymbol{c}_j(t) := \mathrm{Tm}(t, \mathcal{S}_j)$ be the solution of (7), then substituting $t$, $\mathcal{S}$ and $\boldsymbol{c}_j(t)$ into the tilted weight denoted as $\hat{w}_i := w_i(t, \mathcal{S}_j, \boldsymbol{c}_j(t))$, we can obtain the tilted empirical mean and variance for each cluster as follows,

$$\mathbb{E}_t\Big(\mathrm{f}(\mathcal{S}_j, \boldsymbol{c}_j)\Big) = \sum_{\boldsymbol{x}_i \in \mathcal{S}_j} \hat{w}_i \cdot f(\boldsymbol{x}_i, \boldsymbol{c}_j) = \|\boldsymbol{c}_j\|^2 + \sum_{\boldsymbol{x}_i \in \mathcal{S}_j} \hat{w}_i\|\boldsymbol{x}_i\|^2 - \boldsymbol{c}_j^\top M_t$$
$$\mathrm{Var}_t\Big(\mathrm{f}(\mathcal{S}_j, \boldsymbol{c}_j)\Big) = \mathbb{E}_t\Big(\boldsymbol{c}_j^\top\big(-2\boldsymbol{x}_i - M_t\big)\Big)^2 = \boldsymbol{c}_j^\top V_t \boldsymbol{c}_j,$$

where $M_t := 2\sum_{\boldsymbol{x}_i \in \mathcal{S}_j} \hat{w}_i \cdot \boldsymbol{x}_i$ and $V_t := \sum_{\boldsymbol{x}_i \in \mathcal{S}_j} \hat{w}_i\big(-2\boldsymbol{x}_i - M_t\big)^\top\big(-2\boldsymbol{x}_i - M_t\big)$ are constants. Then, by taking derivative of $\mathrm{Var}_\tau\Big(\mathrm{f}\big(\mathcal{S}_j, \boldsymbol{c}_j(t)\big)\Big)$ with respect to $t$, we have

$$\frac{\partial}{\partial t}\Big\{\mathrm{Var}_\tau\Big(\mathrm{f}\big(\mathcal{S}_j, \boldsymbol{c}_j(t)\big)\Big)\Big\}$$
$$= \Big(\frac{\partial}{\partial t}\boldsymbol{c}_j(t)\Big)^\top \cdot \nabla_{\boldsymbol{c}_j}\Big\{\mathrm{Var}_\tau\Big(\mathrm{f}\big(\mathcal{S}_j, \boldsymbol{c}_j(t)\big)\Big)\Big\}$$
$$= 2\Big(\frac{\partial}{\partial t}\boldsymbol{c}_j(t)\Big)^\top V_\tau \boldsymbol{c}_j(t). \tag{17}$$

Based on the optimal condition with $\boldsymbol{c}_j$, we have

$$0 = \sum_{\boldsymbol{x}_i \in \mathcal{S}_j} e^{t\|\boldsymbol{x}_i - \boldsymbol{c}_j(t)\|^2}(\boldsymbol{x}_i - \boldsymbol{c}_j(t)). \tag{18}$$

Divide both sides of (18) by $-\frac{1}{2}\sum_{\boldsymbol{x}_i \in \mathcal{S}_j} e^{t\|\boldsymbol{x}_i - \boldsymbol{c}_j(t)\|^2}$, and differentiate with respect to $t$ yields

$$0 = \frac{\partial}{\partial t}\Big\{\sum_{\boldsymbol{x}_i \in \mathcal{S}_j} w_i(t, \mathcal{S}_j, \boldsymbol{c}_j(t)) \cdot 2(\boldsymbol{c}_j(t) - \boldsymbol{x}_i)\Big\}$$
$$= \frac{\partial}{\partial t}\Big\{2\boldsymbol{c}_j(t) - M(t, \mathcal{S}_j, \boldsymbol{c}_j(t))\Big\}$$
$$= \frac{\partial \boldsymbol{c}_j(t)}{\partial t}\Big(2 - \nabla_{\boldsymbol{c}_j} M(t, \mathcal{S}_j, \boldsymbol{c}_j(t))\Big) - \frac{\partial}{\partial \tau} M(\tau, \mathcal{S}_j, \boldsymbol{c}_j(t))\Big|_{\tau=t} \tag{19}$$
$$= \frac{\partial \boldsymbol{c}_j(t)}{\partial t}\Big(2 + tV(t, \mathcal{S}_j, \boldsymbol{c}_j(t))\Big) + V(t, \mathcal{S}_j, \boldsymbol{c}_j(t))\boldsymbol{c}_j(t), \tag{20}$$

where (19) follows from the chain rule, and (20) follows from Lemma 1. Then we can infer from (20) that

$$\frac{\partial \boldsymbol{c}_j(t)}{\partial t} = -V(t, \mathcal{S}_j, \boldsymbol{c}_j(t))\boldsymbol{c}_j(t) \cdot \frac{1}{2 + tV(t, \mathcal{S}_j, \boldsymbol{c}_j(t))}. \tag{21}$$

Substituting (21) into (17), we obtain

$$\frac{\partial}{\partial t}\Big\{\mathrm{Var}_\tau\Big(\mathrm{f}\big(\mathcal{S}_j, \boldsymbol{c}_j(t)\big)\Big)\Big\} = 2\Big(\frac{\partial}{\partial t}\boldsymbol{c}_j(t)\Big)^\top V_\tau \boldsymbol{c}_j(t)$$
$$= \underbrace{-\frac{\boldsymbol{c}_j(t)^\top V(t, \mathcal{S}_j, \boldsymbol{c}_j(t))V_\tau \boldsymbol{c}_j(t)}{2 + tV(t, \mathcal{S}_j, \boldsymbol{c}_j(t))}}_{<0},$$

which completes the proof. □

## 8 CONCLUSIONS

We proposed TKM and FastTKM to address within-cluster fairness in individually fair clustering. By introducing a variance-based fairness metric and an exponentially tilted SSE objective, our methods reduce intra-cluster distance disparities while encouraging more balanced facility distributions. Extensive experiments show that both algorithms consistently outperform state-of-the-art methods in effectiveness and efficiency, and scale robustly to large datasets without memory overflow. The adjustable variance parameter further enhances their flexibility in practical applications.


## ACKNOWLEDGMENT

This work was supported by the National Key R&D Program of China (2023YFB4503600), National Natural Science Foundation of China (62202338), Key R&D Program of Hubei Province (2023BAB081). This work of Jinshan Zeng is supported in part by the National Natural Science Foundation of China (62376110) and the Major Key Project of PCL (PCL2024A06). This work of Feiping Nie is supported in part by the National Natural Science Foundation of China (62576277). This work of Zhiyong Peng is supported in part by the National Natural Science Foundation of China (62372337).

# REFERENCES


[1] 1988. UC Irvine Machine Learning Repository. https://archive.ics.uci.edu/.
[2] 2017. 3D Road Network (North Jutland, Denmark). https://archive.ics.uci.edu/dataset/246/3d+road+network+north+jutland+denmark.
[3] 2017. The Home Mortgage Disclosure Act. https://ffiec.cfpb.gov/data-browser/.
[4] 2018. Ghana Health Facilities. https://www.kaggle.com/datasets/citizen-ds-ghana/health-facilities-gh.
[5] 2018. Singapore HDB Postal Code Mapper. https://www.kaggle.com/datasets/mylee2009/singapore-postal-code-mapper.
[6] 2022. Cellular Network Analysis Dataset. https://www.kaggle.com/datasets/suraj520/cellular-network-analysis-dataset.
[7] 2022. A Tutorial and Resources for Fair Clustering. https://www.fairclustering.com/.
[8] 2023. Mobile Home Park Inventory. https://www.kaggle.com/datasets/thedevastator/mobile-home-park-inventory.
[9] David Arthur and Sergei Vassilvitskii. 2007. K-means++ the advantages of careful seeding. In *SODA*. 1027–1035.
[10] Ahmad Beirami, A. Robert Calderbank, Mark M. Christiansen, Ken R. Duffy, and Muriel Médard. 2019. A Characterization of Guesswork on Swiftly Tilting Curves. *IEEE Trans. Inf. Theory* 65, 5 (2019), 2850–2871.
[11] Suman Kalyan Bera, Deeparnab Chakrabarty, Nicolas Flores, and Maryam Negahbani. 2019. Fair Algorithms for Clustering. In *NeurIPS*. 4955–4966.
[12] Léon Bottou, Frank E Curtis, and Jorge Nocedal. 2018. Optimization methods for large-scale machine learning. *SIAM Rev.* 60, 2 (2018), 223–311.
[13] Brian Brubach, Darshan Chakrabarti, John P. Dickerson, Samir Khuller, Aravind Srinivasan, and Leonidas Tsepenekas. 2020. A Pairwise Fair and Community-preserving Approach to k-Center Clustering. In *ICML*. 1178–1189.
[14] Francois Buet-Golfouse and Islam Utyagulov. 2022. Towards Fair Unsupervised Learning. In *FAccT*. 1399–1409.
[15] Ronald W Butler. 2007. *Saddlepoint approximations with applications*. Cambridge University Press.
[16] Simon Caton and Christian Haas. 2023. Fairness in Machine Learning: A Survey. *ACM Comput. Surv.* (2023).
[17] Darshan Chakrabarti, John P. Dickerson, Seyed A. Esmaeili, Aravind Srinivasan, and Leonidas Tsepenekas. 2022. A New Notion of Individually Fair Clustering: $\alpha$-Equitable k-Center. In *AISTATS*, Vol. 151. 6387–6408.
[18] Ming-Fang Chang, John W Lambert, Patsorn Sangkloy, Jagjeet Singh, Slawomir Bak, Andrew Hartnett, De Wang, Peter Carr, Simon Lucey, Deva Ramanan, and James Hays. 2019. Argoverse: 3D Tracking and Forecasting with Rich Maps. In *CVPR*.
[19] Xingyu Chen, Brandon Fain, Liang Lyu, and Kamesh Munagala. 2019. Proportionally Fair Clustering. In *ICML*, Vol. 97. 1032–1041.
[20] Rachit Chhaya, Anirban Dasgupta, Jayesh Choudhari, and Supratim Shit. 2022. On Coresets for Fair Regression and Individually Fair Clustering. In *AISTATS*, Vol. 151. 9603–9625.
[21] Flavio Chierichetti, Ravi Kumar, Silvio Lattanzi, and Sergei Vassilvitskii. 2017. Fair Clustering Through Fairlets. In *NeurIPS*. 5029–5037.
[22] Thomas M Cover. 1999. *Elements of information theory*. John Wiley & Sons.
[23] Amir Dembo and Ofer Zeitouni. 2009. *Large deviations techniques and applications*. Springer Science & Business Media.
[24] John P. Dickerson, Seyed A. Esmaeili, Jamie H. Morgenstern, and Claire Jie Zhang. 2024. Doubly Constrained Fair Clustering. In *NeurIPS*.
[25] Tan Do-Duy, Long Dinh Nguyen, Trung Q. Duong, Saeed R. Khosravirad, and Holger Claussen. 2021. Joint Optimisation of Real-Time Deployment and Resource Allocation for UAV-Aided Disaster Emergency Communications. *IEEE J. Sel. Areas Commun.* 39, 11 (2021), 3411–3424.
[26] Yushun Dong, Jing Ma, Song Wang, Chen Chen, and Jundong Li. 2023. Fairness in Graph Mining: A Survey. *IEEE Trans. Knowl. Data Eng.* 35, 10 (2023), 10583–10602.
[27] Paula Dutko, Michele Ver Ploeg, and Tracey Farrigan. 2012. Characteristics and Influential Factors of Food Deserts. (2012).
[28] Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard S. Zemel. 2012. Fairness through awareness. In *ITCS*. 214–226.
[29] Pavan Edara and Mosha Pasumansky. 2021. Big Metadata : When Metadata is Big Data. *Proc. VLDB Endow.* 14, 12 (2021), 3083–3095.
[30] Wenfei Fan. 2022. Big Graphs: Challenges and Opportunities. *Proc. VLDB Endow.* 15, 12 (2022), 3782–3797.
[31] Alexander J. Gates and Yong-Yeol Ahn. 2017. The Impact of Random Models on Clustering Similarity. *J. Mach. Learn. Res.* 18 (2017), 87:1–87:28.
[32] Saeed Ghadimi, Andrzej Ruszczynski, and Mengdi Wang. 2020. A Single Timescale Stochastic Approximation Method for Nested Stochastic Optimization. *SIAM J. Optim.* 30, 1 (2020), 960–979.
[33] Rozhina Ghanavi, Maryam Sabbaghian, and Halim Yanikomeroglu. 2019. Q-Learning Based Aerial Base Station Placement for Fairness Enhancement in Mobile Networks. In *GlobalSIP*. 1–5.
[34] Swati Gupta, Akhil Jalan, Gireeja Ranade, Helen Yang, and Simon Zhuang. 2020. Too Many Fairness Metrics: Is There a Solution?. In *EDSC*.
[35] Swati Gupta, Jai Moondra, and Mohit Singh. 2023. Which Lp norm is the fairest? Approximations for fair facility location across all "p". In *EC*. 817.
[36] Mohammad Hossein, Bateni, Vincent Cohen-Addad, Alessandro Epasto, and Silvio Lattanzi. 2024. A Scalable Algorithm for Individually Fair K-means Clustering. *arXiv:2402.06730* (2024).
[37] Sara John, Megan R Winkler, Ravneet Kaur, Julia DeAngelo, Alex B Hill, Samantha M Sundermeir, Uriyoan Colon-Ramos, Lucia A Leone, Rachael D Dombrowski, Emma C Lewis, et al. 2022. Balancing mission and margins: what makes healthy community food stores successful. *Int. J. Environ. Res. Public Health* 19, 14 (2022), 8470.
[38] Christopher Jung, Sampath Kannan, and Neil Lutz. 2020. Service in Your Neighborhood: Fairness in Center Location. In *FORC*, Vol. 156. 5:1–5:15.
[39] Leon Kellerhals and Jannik Peters. 2024. Proportional fairness in clustering: A social choice perspective. *NeurIPS* 37 (2024), 111299–111317.
[40] Amit Kumar and Jon M. Kleinberg. 2000. Fairness Measures for Resource Allocation. In *FOCS*. 75–85.
[41] Amit Kumar and Jon M. Kleinberg. 2006. Fairness Measures for Resource Allocation. *SIAM J. Comput.* 36, 3 (2006), 657–680.
[42] Yann LeCun. 1998. The MNIST database of handwritten digits. http://yann.lecun.com/exdb/mnist/.
[43] Tian Li, Ahmad Beirami, Maziar Sanjabi, and Virginia Smith. 2023. On Tilted Losses in Machine Learning: Theory and Applications. *J. Mach. Learn. Res.* 24 (2023), 142:1–142:79.
[44] Stuart P. Lloyd. 1982. Least squares quantization in PCM. *IEEE Trans. Inf. Theory* 28, 2 (1982), 129–136.
[45] Sepideh Mahabadi and Ali Vakilian. 2020. Individual Fairness for k-Clustering. In *ICML*, Vol. 119. 6586–6596.
[46] Christopher Meek, Bo Thiesson, and David Heckerman. 2002. The Learning-Curve Sampling Method Applied to Model-Based Clustering. *J. Mach. Learn. Res.* 2 (2002), 397–418.
[47] Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. 2022. A Survey on Bias and Fairness in Machine Learning. *ACM Comput. Surv.* 54, 6 (2022), 115:1–115:35.
[48] Neri Merhav. 2014. List Decoding - Random Coding Exponents and Expurgated Exponents. *IEEE Trans. Inf. Theory* 60, 11 (2014), 6749–6759.
[49] Maryam Negahbani and Deeparnab Chakrabarty. 2021. Better Algorithms for Individually Fair $k$-Clustering. In *NeurIPS*. 13340–13351.
[50] Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. 2001. On Spectral Clustering: Analysis and an algorithm. In *NeurIPS*. 849–856.
[51] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch. (2017).
[52] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake VanderPlas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Edouard Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.* 12 (2011), 2825–2830.
[53] E. Y. Pee and Johannes O. Royset. 2011. On Solving Large-Scale Finite Minimax Problems Using Exponential Smoothing. *J. Optim. Theory Appl.* 148, 2 (2011), 390–421.
[54] Lisa M Powell, Sandy Slater, Donka Mirtcheva, Yanjun Bao, and Frank J Chaloupka. 2007. Food store availability and neighborhood characteristics in the United States. *Prev. Med.* 44, 3 (2007), 189–195.
[55] Qi Qi, Yi Xu, Wotao Yin, Rong Jin, and Tianbao Yang. 2023. Attentional-Biased Stochastic Gradient Descent. *Trans. Mach. Learn. Res.* (2023).
[56] Benjamin Rader, Christina M Astley, Kara Sewalk, Paul L Delamater, Kathryn Cordiano, Laura Wronski, Jessica Malaty Rivera, Kai Hallberg, Megan F Pera, Jonathan Cantor, et al. 2022. Spatial modeling of vaccine deserts as barriers to controlling SARS-CoV-2. *Commun. Med.* 2, 1 (2022), 141.
[57] William M Rand. 1971. Objective criteria for the evaluation of clustering methods. *J. Am. Stat. Assoc.* 66, 336 (1971), 846–850.
[58] Sina Shaham, Gabriel Ghinita, and Cyrus Shahabi. 2022. Models and Mechanisms for Spatial Data Fairness. *Proc. VLDB Endow.* 16, 2 (2022), 167–179.
[59] Chunhua Shen and Hanxi Li. 2010. On the Dual Formulation of Boosting Algorithms. *IEEE Trans. Pattern Anal. Mach. Intell.* 32, 12 (2010), 2216–2231.
[60] Mojtaba Shojaei, Ali Hosseini, and Alireza Shafieinejad. 2025. Fairness-aware placement of multiple aerial base stations in wireless networks. *Wirel. Networks* 31, 2 (2025), 1037–1052.
[61] David Siegmund. 1976. Importance sampling in the Monte Carlo study of sequential tests. *Ann. Stat.* (1976), 673–684.
[62] Ruoyu Sun, Mingyi Hong, and Zhi-Quan Luo. 2015. Joint Downlink Base Station Association and Power Control for Max-Min Fairness: Computation and Complexity. *IEEE J. Sel. Areas Commun.* 33, 6 (2015), 1040–1054.
[63] Attila Szabó, Hadi Jamali Rad, and Siva-Datta Mannava. 2021. Tilted Cross-Entropy (TCE): Promoting Fairness in Semantic Segmentation. In *CVPR*. 2305–2310.
[64] Rong Tang and Yun Yang. 2022. Bayesian Inference for Risk Minimization via Exponentially Tilted Empirical Likelihood. *J. R. Stat. Soc. B.* 84, 4 (2022), 1257–1286.

[65] Ran Tao, Joni Downs, Theresa M Beckie, Yuzhou Chen, and Warren McNelley. 2020. Examining spatial accessibility to COVID-19 testing sites in Florida. *Annals of GIS* 26, 4 (2020), 319–327.
[66] Ali Vakilian and Mustafa Yalçiner. 2022. Improved Approximation Algorithms for Individually Fair Clustering. In *AISTATS*, Vol. 151. 8758–8779.
[67] Ulrike von Luxburg. 2007. A tutorial on spectral clustering. *Stat. Comput.* 17, 4 (2007), 395–416.
[68] Chenhao Wang, Xiaoying Wu, Minming Li, and Hau Chan. 2021. Facility's perspective to fair facility location problems. In *AAAI*, Vol. 35. 5734–5741.
[69] Mengdi Wang, Ethan X. Fang, and Han Liu. 2017. Stochastic compositional gradient descent: algorithms for minimizing compositions of expected-value functions. *Math. Program.* 161, 1-2 (2017), 419–449.
[70] Mengdi Wang, Ji Liu, and Ethan X. Fang. 2017. Accelerating Stochastic Composition Optimization. *J. Mach. Learn. Res.* 18 (2017), 105:1–105:23.
[71] 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 (2020), 163–175.
[72] Yingjie Wang, Hong Chen, Weifeng Liu, Fengxiang He, Tieliang Gong, Youcheng Fu, and Dacheng Tao. 2023. Tilted Sparse Additive Models. In *ICML*, Vol. 202. 35579–35604.
[73] Stephen J Wright. 2015. Coordinate descent algorithms. *Math. Program.* 151, 1 (2015), 3–34.
[74] Shengkun Zhu, Quanqing Xu, Jinshan Zeng, Sheng Wang, Yuan Sun, Zhifeng Yang, Chuanhui Yang, and Zhiyong Peng. 2023. F3KM: Federated, Fair, and Fast k-means. *Proc. ACM Manag. Data* 1, 4 (2023), 241:1–241:25.