Fair-Count-Min: Frequency Estimation under Equal Group-wise Approximation Factor

NIMA SHAHBAZI, Department of Computer Science, University of Illinois Chicago, USA
STAVROS SINTOS, Department of Computer Science, University of Illinois Chicago, USA
ABOLFAZL ASUDEH, Department of Computer Science, University of Illinois Chicago, USA

Frequency estimation in streaming data often relies on sketches like Count-Min to provide approximate answers with sublinear space. However, Count-Min sketches introduce additive errors that disproportionately impact the unpopular groups, creating fairness concerns. To address these concerns, we introduce Fair-Count-Min, a frequency estimation sketch that guarantees equal expected approximation factors across various groups. We propose a column partitioning approach with group-aware semi-uniform hashing to eliminate collisions between elements from different groups. We provide theoretical guarantees for fairness, analyze its associated cost, and validate our findings through extensive experiments on real-world datasets in comparison with representative state-of-the-art baselines. Our experimental results demonstrate that Fair-Count-Min achieves fairness with generally small additional error while maintaining efficiency comparable to that of the Count-Min sketch.

CCS Concepts: • **Information systems** → **Data structures**.

Additional Key Words and Phrases: Approximate Query Processing, Cardinality Estimation, Algorithmic Fairness

ACM Reference Format:
Nima Shahbazi, Stavros Sintos, and Abolfazl Asudeh. 2026. Fair-Count-Min: Frequency Estimation under Equal Group-wise Approximation Factor. *Proc. ACM Manag. Data* 4, 3 (SIGMOD), Article 179 (June 2026), 28 pages. https://doi.org/10.1145/3802056

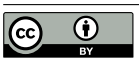
1 Introduction

1.1 Motivation

Estimating element frequencies in data streams is a fundamental problem in computer science, where, given a universe of elements, the objective is to count the appearance of each within the stream. However, due to memory limitations and processing time in streaming settings, exact counting is often impractical. Consequently, approximating the counts using techniques such as the Count-Min (CM) sketch has become widely adopted [31]. The CM estimates the frequency of elements by randomly hashing the elements into a smaller set of buckets, while adding up the frequencies of all elements in each bucket. The frequency count of each bucket is then returned as an upper bound of the frequency of elements it contains.

The CM sketches offer space-efficient frequency estimations with small error guarantees. Nevertheless, *the error guarantees are additive*. Such errors may be negligible for popular elements with a high frequency in the data stream. However, as further elaborated in Example 1, the unpopular

This paper is based upon work supported in part by the National Science Foundation under Award No. 2348919.
Authors' Contact Information: Nima Shahbazi, Department of Computer Science, University of Illinois Chicago, Chicago, Illinois, USA, nshahb3@uic.edu; Stavros Sintos, Department of Computer Science, University of Illinois Chicago, Chicago, Illinois, USA, stavros@uic.edu; Abolfazl Asudeh, Department of Computer Science, University of Illinois Chicago, Chicago, Illinois, USA, asudeh@uic.edu.



ACM 2836-6573/2026/6-ART179
https://doi.org/10.1145/3802056

groups with low frequencies in the stream are disproportionately impacted under such guarantees. That is because even small additive errors can cause a significant relative increase in the count estimation of unpopular (low-frequency) elements. This imbalance gives rise to a fairness issue – an aspect overlooked in the existing work. Let us further illustrate the implications of this issue by presenting the following example:

Example 1: *Urban traffic sensors log vehicle IDs or license plate hashes as cars pass intersections. Buses, taxis, and delivery trucks show up hundreds of times per day, while private vehicles may only appear a few times. Suppose on average the public transportation cars pass the intersections 100 times, while this number is 5 for the private vehicles. In this setting, an additive error of 10 is tolerable for public transportation cars, but it increases the expected estimation for private vehicles by 200%. This distortion matters: traffic management systems could mistakenly flag private vehicles as suspicious, or misallocate toll discounts and congestion pricing.*[1]

Cases similar to Example 1 can be found across a wide range of applications, from wild-life monitoring to recommendation and shingle-frequency estimation.

To address this issue, in this paper, we introduce **Fair-Count-Min** (FCM), a frequency estimation sketch with the equal group-level approximation factor guarantee, ensuring that the relative count-estimation increase is equal for various groups, irrespective of their popularity in the data stream. To the best of our knowledge, FCM is the first frequency estimation sketch with provable *multiplicative error* guarantees in its estimation.

Our key idea in the design of the FCM is the partitioning of the sketch buckets based on a group-aware, semi-uniform hashing scheme that prevents collisions between elements of different groups. Then, we prove that, carefully selecting the number of buckets allocated to each group, our FCM sketch guarantees an equal expected approximation factor across the groups, i.e., equal ratios of the true frequency to the estimated one. The FCM sketch applies to binary and non-binary groups of elements.

Since *our approach is restricted to the same memory capacity as CM, FCM does not introduce any additional memory overhead.* Furthermore, FCM and CM have the same time complexities for update and query operations. In other words, the time taken to look up the frequency estimation of an element and the time to increase the frequency of an element in the data stream are the same in CM and FCM.

In a general setting, where multiple hash functions are used in the FCM, identifying the proper number of buckets allocated to each group requires solving an equation, for which we propose efficient exact and approximation algorithms.

Furthermore, we theoretically analyze the price of fairness (PoF) of FCM. In general, PoF denotes the accuracy loss incurred when fairness constraints are imposed. It arises because accuracy-maximizing outcomes often disproportionately benefit certain groups, and enforcing fairness necessitates deviations from this optimum. Interestingly, we show that in this case achieving fairness may even reduce the total additive error for a specific case when the CM sketch uses a single random hash function, while experimentally demonstrating that the PoF is small for the general settings.

Beyond the theoretical analysis, we conduct extensive experiments on multiple real-world datasets, replicating our motivating examples to evaluate the practical performance of the proposed approach. In summary, our experiments verify (a) while CM is usually not fair, FCM achieves fairness in all experiments, (b) the PoF is generally small, and (c) FCM has the same memory and time efficiency as CM.

1.2 Applications

Example 1 highlights an application, where using traditional CM sketches can cause unfairness in an urban traffic setting. Such examples are not limited to this setting, and can be found across a wide range of domains. In this section[2], we further highlight additional application examples across diverse domains where additive error guarantees are not satisfactory, motivating the need for a (multiplicative) approximation factor and fair count min.

1) Wild Life Monitoring: Automatic monitoring of wildlife in conservation areas is essential for collecting statistics needed to protect and manage species across conservation categories (e.g., *endangered*, *threatened*, and *least concern*). Because manual monitoring is costly and difficult to scale, conservation efforts increasingly rely on automated data collection using resource-constrained sensors such as camera traps. These devices often have limited memory, energy, and communication capacity, making it impractical to store or transmit all detections. To address this, each sensor can maintain approximate detection counts using a traditional Count-Min sketch, where element types correspond to animal species and each detection is an event. While the CM Sketch provides accurate estimates for frequent, least-concern species (e.g., reporting the 1528 detection of crow as 1607), its *additive error guarantees can severely inflate counts for rare endangered species* (e.g., increasing the 12 detection of Chetahs to 109), leading to misleading results. In contrast, an FCM offers multiplicative approximation guarantees, enabling accurate frequency estimation across all species groups, including endangered ones.

2) k-Shingle Frequency in a Document repository: A common task in large-scale document repositories is to estimate the frequency of k-shingles (a sequence of k consecutive words) across a massive collection of text documents. Since the number of possible shingles grows rapidly with vocabulary size and k, and storing exact counts is often infeasible, memory-efficient streaming summaries such as the Count-Min Sketch are widely used for approximate frequency estimation. In practice, CM sketches provide reliable estimates for stop words, that are highly frequent. However, its additive error guarantees can significantly distort the estimated frequencies of technical terms or domain-specific terminology, making the results less useful for downstream tasks such as topic discovery, keyword extraction, or technical content analysis. This limitation motivates the use of Fair Count-Min (FCM), which offers multiplicative approximation guarantees and thus enables accurate estimation across both frequent and rare shingles.

3) Misleading Recommendations: In targeted advertising, recommendations are often driven by users' historical purchase behavior, such as the frequency of purchases within specific item categories (e.g., sports clothing). However, due to privacy constraints and limited storage, it may be infeasible to maintain complete transaction histories for all users. This motivates the use of compact streaming summaries, such as Count-Min Sketches, to approximately track purchase frequencies over time. While CM sketches can efficiently estimate counts for frequent buyers, their additive error guarantees may substantially inflate frequency estimates for low-activity users. This can lead to misleading recommendations and incorrect targeting, resulting in wasted advertising budget and reduced campaign effectiveness.

1.3 Technical Contributions

- We propose a novel notion of *group fairness* in the frequency estimation context, defined by the requirement that the approximation factor (the *multiplicative* overestimation error) remains equal across different groups. This contrasts with traditional approaches based on *additive* error bounds, which tend to disproportionately impact unpopular groups. Using this notion of fairness,

[2]Extended application examples are provided in the technical report [87].

we introduce *Fair-Count-Min* (FCM), the frequency estimation sketch that guarantees group fairness.
- We introduce a *column-partitioning* technique based on semi-uniform hash functions to develop FCM. This method assigns groups to disjoint sets of hash buckets, thereby eliminating inter-group collisions and promoting fairness in estimation. Our proposed method, FCM, is agnostic to both the grouping strategy and the underlying distribution of the data elements. Furthermore, FCM does not increase the memory and time requirements of CM.
- We theoretically prove that FCM is fair, i.e., it ensures an equal expected approximation factor across groups. Furthermore, we analyze the *price of fairness* and show that it is typically small—and in some cases, can even be negative.
- We design both exact and approximate algorithms to efficiently compute the optimal allocation of buckets per group.
- We empirically evaluate FCM on several real-world datasets and compare it with representative state-of-the-art baselines for frequency estimation such as learning-based approach [56] and CM with conservative update [43]. The results show that FCM is the only approach that successfully achieves group fairness while incurring only a generally small additional additive error, all while preserving the time and space efficiency characteristics of CM. Interestingly, CM with conservative update–though effective in minimizing additive error–exhibits the poorest performance with respect to multiplicative error. The learned approach is constrained by its capacity on identifying heavy hitters from the data sample and incurs construction and query times several orders of magnitude higher than non-learned methods.

Remark: Before concluding this section, we would like to note that our paper explicitly targets applications where systematic overestimation of minority groups is harmful. Indeed, for applications without societal impact, and settings where additive overestimation is acceptable, fairness is irrelevant. Examples of such settings include detecting heavy-hitters in market data, or detecting traffic-network anomalies. Such applications are out of the scope of study, and using a standard CM sketch is entirely sufficient. Rather, we address a distinct and well-motivated scenario: in societal applications, where disproportionate inflation of errors for unpopular groups is harmful, FCM provide guarantees that avoid systematically disadvantaging them.

2 Preliminary

2.1 Data Model

Let $\mathcal{D}$ be a data stream consisting of elements (or events), each belonging to a type from a finite universe $\mathcal{U} = \{e_1, e_2, \ldots, e_n\}$ of n element types. Each element type e be associated with a group $\mathrm{g} \in \mathcal{G}$, where $\mathcal{G} = \{\mathrm{g}_1, \mathrm{g}_2, \ldots, \mathrm{g}_\ell\}$, and n_g denotes the number of element types belonging to group g. Let N be the number of elements in the data stream $\mathcal{D}$. We use the function $f : \mathcal{U} \to [N]$ to refer to the frequency of each element type in $\mathcal{D}$. That is, $f(e)$ is the number of times an element of type e appears in $\mathcal{D}$. Formally,

$$f(e) = \sum_{x \in \mathcal{D}} \mathbb{1}(x = e),$$

where $\mathbb{1}(x = e)$ is the indicator function that equals 1 if x is of type e and 0 otherwise.

2.2 Count-Min Sketch

Frequency estimation is a fundamental problem in streaming data processing. Given a data stream $\mathcal{D}$, the goal is to efficiently estimate the frequency $f(e)$ of each element type e. It is easy to see that finding an exact solution to the above problem requires $\Theta(n)$ space, as it involves maintaining n

counters for each element type. To overcome this limitation, frequency estimation data structures (a.k.a sketches) have been designed to provide approximate answers to such queries while using sub-linear space. In addition to being space-efficient, these sketches provide probabilistic error guarantees through tunable parameters and enable constant-time update and query operations.

CM is among the most widely used sketches for frequency estimation. It represents the frequency estimates using a two-dimensional array CM of d rows and w columns. The sketch employs d independent hash functions $\mathsf{h}_1, \mathsf{h}_2, \ldots, \mathsf{h}_d$, where each $\mathsf{h}_i : \mathcal{U} \to [w]$ maps an element type to a column index (called *bucket* or *bin* in the rest of the paper) in row i. These hash functions distribute element types across different buckets to reduce the impact of hash collisions. When a new element of type e arrives in the stream, the sketch updates its counts as follows:

$$CM[i, \mathsf{h}_i(e)] \leftarrow CM[i, \mathsf{h}_i(e)] + 1, \quad \forall i \in [d].$$

Each row records a count for e, though potential overestimations may occur due to hash collisions. To estimate the frequency of an element type[3] e, the sketch returns:

$$\hat{f}(e) = \min_{i=1}^{d} CM[i, \mathsf{h}_i(e)].$$

Taking the minimum across multiple rows mitigates over-counting errors. As previously mentioned, the estimation error in CM occurs due to hash collisions. For each element type e the estimated frequency $\hat{f}(e)$ is always an upper bound on the true frequency $f(e)$:

$$\hat{f}(e) = f(e) + \varepsilon_A(e), \tag{1}$$

where the additive error $\varepsilon_A(e)$ corresponds to the combined frequency of all other elements that hash into the same bucket as e. In expectation, the additive error $\varepsilon_A(e)$ is bounded by $\frac{N}{w}$. Formally:

$$\mathbb{E}[\hat{f}(e)] = f(e) + \frac{\sum_{\forall e' \neq e,\, \mathsf{h}(e') = \mathsf{h}(e)} f(e')}{w} \leq f(e) + \frac{N}{w} \tag{2}$$

Several variations of the CM exist, including the CM with conservative updates [43], the Count-Mean-Min sketch [35], and other related frequency estimation structures such as the Count sketch [24] and Spectral Bloom Filters [29]. While using CM as the foundation for the development of our sketch, and deriving our theoretical analysis, *we compare our work against other baselines in experiments.*

2.3 Fairness: Equal Expected Approximation Factor

Existing CM sketches provide an additive upper-bound error ε_A on the frequency estimation of the elements. However, while ε_A may be a negligible error for the popular groups with frequent elements, it can be intolerable for the unpopular ones, with elements that appear infrequently in the sequence. In other words, the significance of the frequency estimation error is relative to the frequency value. To further clarify this, let us consider Example 2.

[3]In the rest of the paper, we use "element e" and "element type e" interchangeability to refer to elements of type e.

Example 2: *Continuing with Example 1, we consider the NYC Bus [91] dataset with a universe of element types* $\mathcal{U} = \{e_1, e_2, \ldots, e_{61,906}\}$*, where each element type* e *represents a bus route uniquely identified by the tuple* (PublishedLineName, VehicleRef)*. The element types* $\{e_1, \ldots, e_{10,462}\}$ *correspond to routes operated by the MTA Bus Company (MTABC), while the remaining element types* $\{e_{10,463}, \ldots, e_{61,906}\}$ *correspond to routes operated by New York City Transit (NYCT). Now consider a CM sketch with a hash function* h *that maps element types into* $w = 10{,}000$ *bins. Let* $N = 26{,}520{,}716$ *denote the total number of elements in the stream. The resulting additive error of the sketch is therefore bounded by* $\varepsilon_A \leq 2{,}652$*. Assume that the expected frequencies of element types from the MTABC and NYCT groups are* 11 *and* 513*, respectively. Consequently, the expected frequency estimate for an element type from NYCT is bounded by* 3,165*, which corresponds to a relative error of approximately* 515%*. In contrast, the relative error for an element type from the MTABC group can be as large as* 26,570%*.*

From Example 2, it is clear that the additive error does not provide a strong guarantee independent of the element frequencies. Specifically, providing large estimation errors relative to small frequency values, it is *unfair* for unpopular groups. This motivates us to instead promote *approximation factor* as a stronger frequency estimation guarantee.

Definition 1 (Approximation Factor). *Let* $\hat{f}(e)$ *be the frequency estimation of the element type* e *by a count-min sketch CM. The approximation factor of CM for* e *is* $\alpha(e) = \frac{f(e)}{\hat{f}(e)}$.

Ideally, one would like the sketch to provide an equal approximation factor for all individual element types i.e. to be individually fair. This, however, may not be practical because providing such a guarantee would require knowing (at least approximately) the frequencies of all elements apriori and, in some settings, would require $O(n)$ cells, which is not reasonable. Therefore, following the bulk of the research on algorithmic fairness [17], we turn our objective to the middle-ground of **group fairness**.

Using Definition 1, we define the notion of *group fairness* that equalizes the expected approximation factor across various groups. Specifically, a CM is called fair if it satisfies Definition 2.

Definition 2 (Group-Fair Count-Min (FCM)). *A count-min sketch is group-fair iff:*

$$\forall \mathbf{g}_i, \mathbf{g}_j \in \mathcal{G}, \quad \mathbb{E}_{e \in \mathbf{g}_i}\left[\alpha(e)\right] = \mathbb{E}_{e' \in \mathbf{g}_j}\left[\alpha(e')\right]$$

Following our motivation examples, Definition 2 ensures the sketch performs equally good across various groups, by guaranteeing an equal multiplicative error across them, which eliminates disproportionately large overestimation, particularly for the unpopular (low-frequency) groups. This aligns with the general definition of **performance parity** fairness [17]. Disproportionate overestimation for the unpopular group can disparately harm those groups and lead to discrimination. In our motivation examples, for instance, private vehicles can disproportionately get falsely flagged as suspicious, or additional monetary cost for small-businesses advertisement. An FCM sketch ensures such performance disparities and potential disproportionate harms do not happen.

We note that, instead of considering all groups, one could define fairness over a subset of groups of interests for the users (i.e., the groups the users actually query). In such settings, it is enough to combine all other groups as one group of "others", and simply apply Definition 2 on the groups of interest. Without loss of generality and to ease the explanations, we use the binary groups $\mathbf{g}_1$ and $\mathbf{g}_2$ for explaining our sketches and drawing the analyses. Our results readily hold for non-binary and *arbitrary grouping* of element types, as we shall further discuss in Section 3.4.

2.4 Problem Formulation

With the necessary terms and notations defined, we now formally define our problem of interest:

Definition 3. *Given a data stream of elements $\mathcal{D}$ drawn from a universe of element types $\mathcal{U}$ where each element type belongs to a group $g \in \mathcal{G}$, design a group-fair count-min sketch, whose overall additive error increase compared to CM (price of fairness) is small.*

2.5 Solution Overview

A main issue that causes the unbounded multiplicative error in CM is that high-frequency elements can collide with low-frequency ones, adding a major overestimation to their counts. Similarly, a sketch in our case might not be group-fair if elements in $\mathbf{g}_1$ collide with elements in $\mathbf{g}_2$.

One way to reduce the likelihood of such collisions is to substantially increase the number of bins w and/or the number of independent hash functions d used in the estimation, i.e., the number of rows in the sketch. However, this approach has a fundamental drawback: to ensure that an element $e \in \mathbf{g}_1$ is unlikely to collide with any element in $\mathbf{g}_2$ across at least one of the d rows, the sketch size must scale linearly with n, the number of distinct element types—defeating the purpose of sketching.

Instead, to design group-fair CM, our goal is to *make it impossible for elements in different groups to collide*. Specifically, we ensure our goal by designing group-aware "semi-uniform" hash schemes that isolate elements of $\mathbf{g}_1$ and $\mathbf{g}_2$ by strategically partitioning and assigning the columns of the sketch to $\mathbf{g}_1$ and $\mathbf{g}_2$ (Section 3). Proving that our sketch satisfies Definition 2, we study its price of fairness (Section 5).

3 Fair-Count-Min using Group-Aware Semi-Uniform Hashing

Our idea for generating group-fair CM (FCM) is to utilize "semi-uniform" hash functions that separate element types of various groups. Specifically, reserving $w_{\mathbf{g}_1}$ bins for the group $\mathbf{g}_1$and $w_{\mathbf{g}_2} = w - w_{\mathbf{g}_1}$ for the group $\mathbf{g}_2$, the semi-uniform hash function $\mathsf{h}(.)$ assigns each element in $\mathbf{g}_1$to the first $w_{\mathbf{g}_1}$ bins and the others to the remaining $w_{\mathbf{g}_2}$ bins (Figure 1):

$$\mathsf{h} : \begin{cases} \mathbf{g}_1 \rightarrow [w_{\mathbf{g}_1}] \\ \mathbf{g}_2 \rightarrow w_{\mathbf{g}_1} + [w_{\mathbf{g}_2}] \end{cases}$$

The key question we shall answer in the rest of this section is whether there exists a value $w_{\mathbf{g}_1}$ (hence $w_{\mathbf{g}_2} = w - w_{\mathbf{g}_1}$) for which a CM based on the semi-uniform hash scheme becomes group-fair.

In the following, first, we focus on the case where $d = 1$, i.e., the sketch is based on only one hash function h. Next, we extend our analysis to the general values of d.

3.1 $d = 1$

First, let us derive the expected approximation factor for an element e in group $\mathbf{g}_1$. For the case of $d = 1$, we employ the stronger guarantee of the average approximation factor instead of the expected one. For clarity, we use the expectation notation $\mathbb{E}$ to also denote the average. Let $n_{\mathbf{g}_1}$ be the number of element types in $\mathbf{g}_1$.

In order to satisfy group fairness (Definition 2), we need to ensure that the expected approximation factor for the two groups is the same. That is, $\mathbb{E}\left[\alpha_{\mathbf{g}_1}\right] = \mathbb{E}\left[\alpha_{\mathbf{g}_2}\right]$.

Lemma 1. $\mathbb{E}\left[\alpha_{\mathbf{g}_1}\right] = \frac{w_{\mathbf{g}_1}}{n_{\mathbf{g}_1}}$ *and* $\mathbb{E}\left[\alpha_{\mathbf{g}_2}\right] = \frac{w - w_{\mathbf{g}_1}}{n - n_{\mathbf{g}_1}}$.

Proof of Lemma 1. Let C_i be the total frequency counts in each low-frequency bucket $i \in [w_{\mathbf{g}_1}]$. That is, $C_i = \sum_{e_j : \mathsf{h}(e_j) = i} f(e_j)$.

For every element $e_j \in \mathbf{g}_1$, the value of the bucket it hashed to is returned as its frequency estimation. That is, $\hat{f}(e_j) = C_{\mathrm{h}(e_j)}$. Therefore, following the approximation factor definition, $\hat{f}(e_j) = \frac{f(e_j)}{\alpha(e_j)} = C_{\mathrm{h}(e_j)}$. Hence,

$$\alpha(e_j) = \frac{f(e_j)}{C_{\mathrm{h}(e_j)}} \tag{3}$$

Therefore,

$$\begin{aligned}
\mathbb{E}\left[\alpha_{\mathbf{g}_1}\right] &= \frac{1}{n_{\mathbf{g}_1}} \sum_{e_j \in \mathbf{g}_1} \alpha(e_j) = \frac{1}{n_{\mathbf{g}_1}} \sum_{e_j \in \mathbf{g}_1} \frac{f(e_j)}{C_{\mathrm{h}(e_j)}} \\
&= \frac{1}{n_{\mathbf{g}_1}} \sum_{i=1}^{w_{\mathbf{g}_1}} \sum_{e_j : \mathrm{h}(e_j)=i} \frac{f(e_j)}{C_i} \qquad \texttt{//breaking the calculation per cell} \\
&= \frac{1}{n_{\mathbf{g}_1}} \sum_{i=1}^{w_{\mathbf{g}_1}} \frac{1}{C_i} \sum_{e_j : \mathrm{h}(e_j)=i} f(e_j) \qquad \texttt{//factoring out the common term} \\
&= \frac{1}{n_{\mathbf{g}_1}} \sum_{i=1}^{w_{\mathbf{g}_1}} \frac{1}{C_i} C_i \qquad \texttt{//replacing sum of frequencies with } C_i \\
&= \frac{1}{n_{\mathbf{g}_1}} \sum_{i=1}^{w_{\mathbf{g}_1}} 1 = \frac{w_{\mathbf{g}_1}}{n_{\mathbf{g}_1}}
\end{aligned} \tag{4}$$

Similarly, the expected approximation factor for the group $\mathbf{g}_2$can be computed as

$$\mathbb{E}\left[\alpha_{\mathbf{g}_2}\right] = \frac{w_{\mathbf{g}_2}}{n_{\mathbf{g}_2}} = \frac{w - w_{\mathbf{g}_1}}{n - n_{\mathbf{g}_1}}$$

□

By Lemma 1, to ensure fairness, $w_{\mathbf{g}_1}$ is chosen as follows:

$$\mathbb{E}\left[\alpha_{\mathbf{g}_1}\right] = \mathbb{E}\left[\alpha_{\mathbf{g}_2}\right] \Leftrightarrow \frac{w_{\mathbf{g}_1}}{n_{\mathbf{g}_1}} = \frac{w - w_{\mathbf{g}_1}}{n - n_{\mathbf{g}_1}} \Leftrightarrow w_{\mathbf{g}_1} = \frac{n_{\mathbf{g}_1}}{n} w \tag{5}$$

In other words, according to Equation 5, if the number of bins allocated to each group is proportional to their size, i.e., $\frac{n_{\mathbf{g}_1}}{n}$, is sufficient to ensure fairness. Formally,

Theorem 1. *A CM sketch with a group-aware semi-uniform hash function* $\mathrm{h}(\cdot)$ *is group-fair, if the number of bins* $w_{\mathbf{g}}$ *allocated to each group is proportional to the ratio of element types from that group. That is,* $w_{\mathbf{g}} = \frac{n_{\mathbf{g}}}{n} w, \forall \mathbf{g} \in \mathcal{G}$.

Before moving to the larger values of d, i.e., CM with more number of hash functions, let us take a closer look at Theorem 1 and what it indicates. A standard CM hashes the n element types uniformly across the $[w]$ bins (irrespective of which group they belong to). As a result, the expected number of element types in each bucket is $\frac{n}{w}$. According to Theorem 1, $w_{\mathbf{g}} = \frac{n_{\mathbf{g}}}{n} w, \forall \mathbf{g} \in \mathcal{G}$. As a result, the number of elements in each bucket for each group $\mathbf{g}$ is:

$$\frac{n_{\mathbf{g}}}{w_{\mathbf{g}}} = \frac{n_{\mathbf{g}}}{\frac{n_{\mathbf{g}}}{n} w} = \frac{n}{w} \tag{6}$$

In other words, to ensure fairness, we need to *allocate bins to the group in a way that the expected number of element types hashed to each bucket remains unchanged*, i.e., $\frac{n}{w}$. This helps us extend d to more than one row by *equalizing the expected number of element types hashed in each bin* across various groups to ensure fairness.

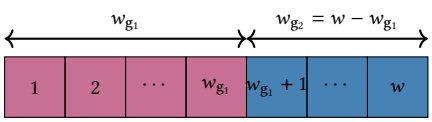


Fig. 1. Column-partitioning based group-fair min-count with one row.

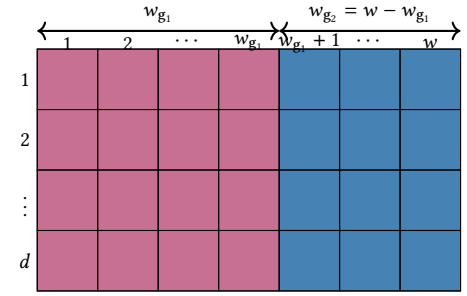


Fig. 2. Column-partitioning based group-fair CM with d rows.

3.2 General Value of d

Having established that a CM with a single group-aware semi-fairness hash function ($d = 1$) ensures group fairness, we now generalize this result to the case of d hash functions (Figure 2).

Similar to $d = 1$, our goal is to partition the w columns across various groups such that the group fairness requirement is satisfied, i.e., all groups have the same expected approximation factor.

We recall from Theorem 1 that achieving group fairness requires equalizing the expected number of element types hashed into the minimum-count bucket across d rows for all groups.

When $d = 1$, the estimated frequency of an element is the frequency count of the bucket it is hashed into. On the other hand, when $d > 1$, a frequency estimation $\hat{f}(e)$ is the *minimum* of the frequency counts of the buckets e is hashed to at each row.

Let us define the size of a bucket as the number of element types hashed to it. For an element $e_j \in \mathbf{g}$ and a row i, let $l = \mathsf{h}_i(e_j)$. Let the random integer variable X_i be the size of the bucket of the element e at row i , i.e., $|\{e \in \mathbf{g} | \mathsf{h}_i(e) = l\}|$. The elements are hashed uniformly at random and independently into the bins. As a result, the probability that a random element is hashed into the bucket l is $\frac{1}{w_\mathbf{g}}$. As a result, X_i follows the binomial distribution $X_i \sim Bin\left(n_\mathbf{g}, \frac{1}{w_\mathbf{g}}\right)$:

$$\Pr(X_i = x) = \binom{n_\mathbf{g}}{x} \cdot \left(\frac{1}{w_\mathbf{g}}\right)^x \cdot \left(\frac{w_\mathbf{g} - 1}{w_\mathbf{g}}\right)^{n_\mathbf{g} - x}$$

Let $\mu_\mathbf{g}$ be the expected frequency for group $\mathbf{g}$. Then, the expected frequency count of each bucket is $\mu_\mathbf{g}$ times the size of the bucket. Following this argument, we simplify our analysis by equalizing the expected number of element types hashed into the *minimum-size* bucket across d rows for all groups.

Fix an element e_j in the group $\mathbf{g}$. Let $X_1, \cdots, X_d$ be the random variables reflecting the sizes of the d buckets e_j is hashed to across various rows, while each X_i following the Binomial distribution

$$X_1, \cdots, X_d \sim Bin\left(n_\mathbf{g}, \frac{1}{w_\mathbf{g}}\right)$$

Define the random variable Y_j as the minimum of X_1 to X_d. That is,

$$Y \sim \min(X_1, \cdots, X_d).$$

In order to satisfy group fairness (Definition 2), we need to ensure that the expected approximation factor for every group is the same. Given a group $\mathbf{g}$, in the next lemma we show a formula to compute $\mathbf{g}$'s expected approximation factor. The proof is shown in our technical report [87].

LEMMA 2. $\mathbb{E}_{e \in \mathbf{g}}[Y] = \sum_{x=1}^{n_\mathbf{g}} \left[\sum_{i=x}^{n_\mathbf{g}} \binom{n_\mathbf{g}}{i} \left(\frac{1}{w_\mathbf{g}}\right)^i \left(\frac{w_\mathbf{g}-1}{w_\mathbf{g}}\right)^{n_\mathbf{g}-i} \right]^d$.

For two groups $\mathbf{g}_1$ and $\mathbf{g}_2$, we allocate $w_{\mathbf{g}_1}$ columns to $\mathbf{g}_1$ and $w_{\mathbf{g}_2} = w - w_{\mathbf{g}_1}$ columns to $\mathbf{g}_2$ by enforcing that the two groups have the same expected approximation factor. Formally, using Lemma 2, we solve the following equation (recall that $n_{\mathbf{g}_2} = n - n_{\mathbf{g}_1}$):

$$\begin{aligned} &\mathbb{E}_{e\in\mathbf{g}_1}[Y] = \mathbb{E}_{e\in\mathbf{g}_2}[Y] \Rightarrow \\ &\sum_{x=1}^{n_{\mathbf{g}_1}} \left[\sum_{i=x}^{n_{\mathbf{g}_1}} \binom{n_{\mathbf{g}_1}}{i} \left(\frac{1}{w_{\mathbf{g}_1}}\right)^i \left(\frac{w_{\mathbf{g}_1}-1}{w_{\mathbf{g}_1}}\right)^{n_{\mathbf{g}_1}-i} \right]^d \\ &\quad = \sum_{x=1}^{n-n_{\mathbf{g}_1}} \left[\sum_{i=x}^{n-n_{\mathbf{g}_1}} \binom{n-n_{\mathbf{g}_1}}{i} \left(\frac{1}{w-w_{\mathbf{g}_1}}\right)^i \left(\frac{w-w_{\mathbf{g}_1}-1}{w-w_{\mathbf{g}_1}}\right)^{n-n_{\mathbf{g}_1}-i} \right]^d \end{aligned} \tag{7}$$

Later in Section 4, we shall provide an efficient algorithm for solving Equation 7.

3.3 Negative Result: Row Partitioning Baseline

FCM follows a column partitioning approach, using the group-aware semi-uniform hashing schemes. Alternatively, rows can be divided by allocating a specific number of exclusive rows to each group, where the allocation of rows to each group is determined by the number of element types within that group. Following Lemma 2, the expected size of the minimum-size bucket across the d' rows, w' columns, and a group with n' elements can be computed as,

$$\mathcal{E}(n', d', w') = \sum_{x=1}^{n'} \left[\sum_{i=x}^{n'} \binom{n'}{i} \left(\frac{1}{w'}\right)^i \left(\frac{w'-1}{w'}\right)^{n'-i} \right]^{d'} \tag{8}$$

Let $d_{\mathbf{g}_1}$ and $d_{\mathbf{g}_2} = d - d_{\mathbf{g}_1}$ be the number of rows allocated to the groups $\mathbf{g}_1$ and $\mathbf{g}_2$. Then, we can find the proper values of $d_{\mathbf{g}_1}$ and $d_{\mathbf{g}_2}$ that achieve fairness by solving the following equation:

$$\mathcal{E}(n_{\mathbf{g}_1}, d_{\mathbf{g}_1}, w) = \mathcal{E}(n_{\mathbf{g}_2}, d_{\mathbf{g}_2}, w) \tag{9}$$

However, there is a major issue that makes this approach unlikely to work: unlike w, which is a relatively large number, d is a small constant (usually on the order of tens). As a result, there are only a small, constant number of choices for $d_{\mathbf{g}_1} \in [d]$. In practice, this makes it unlikely that d is divisible such that Equation 9 is satisfied. We shall verify this negative result in our experiments.

3.4 Generalization to Multiple Arbitrary Groups

So far, we have explained our techniques for the binary grouping of the elements based on their frequencies.

Our approach works for any (arbitrary) grouping of the elements since none of our definitions, results, and analyses are based on any assumption about how elements are grouped. This is also observed in our experiments in Section 6, where we, for example, use demographic groups to group the elements.

Extending our results to the non-binary grouping of the element types is straightforward. Let $\mathcal{G} = \{\mathbf{g}_1, \mathbf{g}_2, \cdots, \mathbf{g}_\ell\}$ be the non-binary set of groups. For every group $\mathbf{g} \in \mathcal{G}$, transform the problem into the binary grouping by merging all other groups as the super-group "others". Then, for $d = 1$, following Theorem 1, $w_{\mathbf{g}} = \frac{n_{\mathbf{g}}}{n} w$. Similarly, for $d > 1$, the value of $w_{\mathbf{g}}$, $\forall \mathbf{g} \in \mathcal{G}$, can be computed by solving Equation 7. We propose an efficient algorithm for finding $w_{\mathbf{g}}$ values when $d > 1$ in Section 4.

3.5 Estimating Group Sizes

Our approach does not make any assumptions about the stream length N or the frequencies, including the group frequencies $N_{\mathbf{g}}$. We only assume a fixed universe model, where the universe of element types $\mathcal{U}$ is known. We further know which group each element belongs to.

While this is standard in many settings, including our motivation examples, in some streaming settings, the universe $\mathcal{U}$ may not be known a priori. In such cases, sampling can be employed to estimate the number of elements in each group ($n_{\mathbf{g}}$). Such approaches are standard in learned frequency-estimation sketches, such as [2, 56, 78, 96, 99], where a classifier is trained on a small sample to infer these parameters.

To obtain the required estimates, we divide the process into a warm-up phase followed by a steady-state phase. Let $d \cdot w$ denote the memory allocated to the CM sketch. During the warm-up phase, each memory cell is used to store the *exact* count of a single element type. As long as the number of distinct element types observed in the stream remains below $d \cdot w$, we can maintain exact counts for all such elements.

Consequently, at the end of the warm-up phase, we observe $d \cdot w$ distinct samples of element types. These samples are used to estimate the group sizes, using Lemma 3, and to construct the FCM accordingly.

Lemma 3. *Consider a universe $\mathcal{U}$ of n element types, where $n_{\mathbf{g}}$ of them belong to a group $\mathbf{g}$. Let m be the total number of elements observed during the warm-up phase. That is $m = \sum_{i=1}^{d.w} C_i$. Similarly, $x_{\mathbf{g}} = m = \sum_{e_{S_j} \in \mathbf{g}} C_j$ be the number of element types of group $\mathbf{g}$ observed during the warm-up phase. Viewing the observed sequence as a random set of $\mathcal{U}$ with replacement, define $\widehat{n}_{\mathbf{g}} = \frac{x_{\mathbf{g}}\, n}{d\, w}$. Then, $\widehat{n}_{\mathbf{g}}$ is an unbiased estimator of $n_{\mathbf{g}}$. Furthermore, for any $\epsilon > 0$,*[4]

$$\Pr\left(|\widehat{n}_{\mathbf{g}} - n_{\mathbf{g}}| \geq n\epsilon\right) \leq 2\exp(-2m\epsilon^2).$$

At the beginning of the steady-state phase, the sketch is initialized using the exact counts collected during the warm-up phase.

Bounding the tail probability, Lemma 3 shows that, for a large enough value of m, the estimation of group sizes are close to the actual values. Still, as a consequences of estimating these values, the FCM sketch may introduce a small amount of unfairness. Nevertheless, while these estimates only approximate $n_{\mathbf{g}}$, our experimental evaluations show that the resulting impact on both unfairness and the price of fairness is generally small, although it remains dependent on the underlying stream distribution. Lastly, we note that group fairness guarantees are susceptible to compromise under adversarial conditions. This occurs primarily when the sampling process fails to accurately estimate the cardinality of element types, as seen in cases of extreme distributional drift. Section 8.3 details a mitigation strategy for these instances, which requires a larger memory footprint.

4 Computing The Column Widths

Our algorithm is straightforward; we should compute the integer value of $w_{\mathbf{g}_1}$ such that Equation (7) holds. We note that Equation (7) may not hold for an integer value of $w_{\mathbf{g}_1}$. In this case the goal is to compute the integer value of $w_{\mathbf{g}_1}$ such that the left side of Equation (7) is as close as possible to the right side of Equation (7). A naive implementation is to try every value of $w_{\mathbf{g}_1}$ and calculate the value of the left and right side of Equation (7) by simply evaluate every term in the sum. Such an implementation would take $\Omega(n^3)$ time because we should try $O(n)$ values of $w_{\mathbf{g}_1}$, and for each such value the calculation of each side of Equation (7) takes $\Omega(n^2)$ time to evaluate the summations.

[4]Proofs are provided in our technical report [87].

We first show an exact near-linear time algorithm to compute the value of $w_{\mathbf{g}_1}$ that solves Equation 7 and then we briefly discuss an approximation algorithm that is used in our experiments to efficiently evaluate the sums of the binomial function.

Efficient exact computation. We first assume that we have two groups, the unpopular elements $\mathbf{g}_1$and the popular elements $\mathbf{g}_2$. Then we extend our method to multiple groups $\mathcal{G} = \{\mathbf{g}_1, \ldots, \mathbf{g}_\ell\}$. Similarly, to the previous section, let $\mathcal{E}(n_{\mathbf{g}_1}, d, w_{\mathbf{g}_1})$ be the function that represents the left side of Equation 7. Equivalently, the right side of Equation (7) is captured by $\mathcal{E}(n_{\mathbf{g}_2}, d, w - w_{\mathbf{g}_1})$. By definition, the function $\mathcal{E}$ computes the expected error of group $\mathbf{g}_1$ (resp. $\mathbf{g}_2$) elements. The values $n_{\mathbf{g}_1}$, $n_{\mathbf{g}_2}$, and d are fixed, so the more bins allocated to group $\mathbf{g}_1$ (respectively, $\mathbf{g}_2$) elements, the smaller the expected error. Hence, the function $\mathcal{E}(n_{\mathbf{g}_1}, d, w_{\mathbf{g}_1})$ is monotone decreasing with respect to $w_{\mathbf{g}_1}$, while $\mathcal{E}(n_{\mathbf{g}_2}, d, w - w_{\mathbf{g}_1})$ is monotone increasing with respect to $w_{\mathbf{g}_1}$. We formally establish the monotonicity of $\mathcal{E}$ in the following proposition. The proof is shown in our technical report [87].

Proposition 1. *For fixed n and d, the function $\mathcal{E}(n, d, w)$ is decreasing with respect to w.*

This property leads to the observation that instead of trying all values of $w_{\mathbf{g}_1}$, we can run a binary search in the range $[1, \ldots, n]$ and compute the best value of $w_{\mathbf{g}_1}$. The overall idea of our algorithm is the following. For every value of $w_{\mathbf{g}_1}$ in the binary search, we evaluate $\mathcal{E}(n_{\mathbf{g}_1}, d, w_{\mathbf{g}_1})$ and $\mathcal{E}(n_{\mathbf{g}_2}, d, w - w_{\mathbf{g}_1})$. If $\mathcal{E}(n_{\mathbf{g}_1}, d, w_{\mathbf{g}_1}) < \mathcal{E}(n_{\mathbf{g}_2}, d, w - w_{\mathbf{g}_1})$ then we try larger values of $w_{\mathbf{g}_1}$ in the binary search. Otherwise, we continue the binary search with smaller values. In the end, we return the value of $w_{\mathbf{g}_1}$ found by the binary search such that $|\mathcal{E}(n_{\mathbf{g}_1}, d, w_{\mathbf{g}_1}) - \mathcal{E}(n_{\mathbf{g}_2}, d, w - w_{\mathbf{g}_1})|$ is minimized.

Even if someone naively applies binary search, the running time would be $\Omega(n^2)$ to evaluate the sums. Given a value of $w_{\mathbf{g}_1}$ we show how to compute $\mathcal{E}(n_{\mathbf{g}_1}, d, w_{\mathbf{g}_1})$ in near-linear time with respect to n. The computation of $\mathcal{E}(n_{\mathbf{g}_2}, d, w - w_{\mathbf{g}_1})$ is equivalent. First, for $x = 1$ we compute the sum $\sum_{i=1}^{n_{\mathbf{g}_1}} \binom{n_{\mathbf{g}_1}}{i} \left(\frac{1}{w_{\mathbf{g}_1}}\right)^i \left(\frac{w_{\mathbf{g}_1}-1}{w_{\mathbf{g}_1}}\right)^{n_{\mathbf{g}_1}-i}$ as follows. For $i = 1$, we compute $b_1 = \binom{n_{\mathbf{g}_1}}{1}$, $t_1 = \frac{1}{w_{\mathbf{g}_1}}$, and $s_1 = \left(\frac{w_{\mathbf{g}_1}-1}{w_{\mathbf{g}_1}}\right)^{n_{\mathbf{g}_1}-1}$. We store $\alpha_1 = b_1 \cdot t_1 \cdot s_1$. Then for $i \in \{2, \ldots, n_{\mathbf{g}_1}\}$, we observe that $b_i = b_{i-1} \cdot \frac{n_{\mathbf{g}_1}+1-i}{i}$, $t_i = t_{i-1} \cdot \frac{1}{w_{\mathbf{g}_1}}$, $s_i = s_{i-1} \cdot \frac{w_{\mathbf{g}_1}}{w_{\mathbf{g}_1}-1}$, and we set $\alpha_i = b_i \cdot t_i \cdot s_i$. Then, we set $\beta_{n_{\mathbf{g}_1}} = \alpha_{n_{\mathbf{g}_1}}$ and for every $i = n_{\mathbf{g}_1} - 1, \ldots, 1$, we compute $\beta_i = \alpha_i + \beta_{i+1}$. The main observation of our algorithm is that $\mathcal{E}(n_{\mathbf{g}_1}, d, w_{\mathbf{g}_1}) = \sum_{x=1}^{n_{\mathbf{g}_1}} \beta_x^d$.

In the technical report [87], we show the next lemma.

Lemma 4. *The time complexity of our algorithm for column-width computation for 2 groups is bounded by $O(n \log^2 n)$.*

We can extend our method to multiple groups $\mathcal{G} = \{\mathbf{g}_1, \ldots, \mathbf{g}_\ell\}$. For each group $\mathbf{g}_j \in \mathcal{G}$, the function $\mathcal{E}(n_j, d, w_j)$ represents the expected error of the elements in group $\mathbf{g}_j$ as computed by lemma 2. The goal is to compute w_j for every $\mathbf{g}_j \in \mathcal{G}$, given that $\sum_{j \in [\ell]} w_j = w$. We solve this problem, by recursively execute our algorithm for two groups ($\mathbf{g}_1$ and $\mathbf{g}_2$) $\ell - 1$ times. Intuitively, if we have ℓ groups, we construct an instance with two colors: one that consists of the elements in the first group, and one that consists of the elements from all groups excluding the first group. For a value $j \in \{1, \ldots, \ell - 1\}$, let W_j be the number of bins that should be assigned to groups $\mathbf{g}_j, \mathbf{g}_{j+1} \ldots, \mathbf{g}_\ell$. Initially, notice that $W_1 = w$. We set $n'_j = \sum_{b=j+1}^{\ell} n_b$. We assume that there exist two groups, one with n_j elements and the other one with n'_j elements, while the total number of bins is W_j. We solve this instance using our efficient implementation for two groups. Let w_j be the number we compute. We set $W_{j+1} \leftarrow W_j - w_j$ and we continue recursively with $j \leftarrow j + 1$. The overall time of our algorithm for groups $\mathcal{G} = \{\mathbf{g}_1, \ldots, \mathbf{g}_\ell\}$ is $O(\ell \cdot n \cdot \log^2 n)$.

Approximate practical implementation. While in theory our previous algorithm operates in the real-RAM model of computation, a direct numerical evaluation of the binomial probability mass function can be problematic in practice. Although the binomial coefficient $\binom{n}{i}$ can be represented exactly using arbitrary-precision integers, its magnitude grows exponentially in n, and converting it to floating-point arithmetic—as required when multiplying by probability terms—quickly leads to numerical overflow. In our experiments, this overflow occurs for moderate values of i once n is on the order of a few hundred, rendering a naive implementation numerically unstable. In practice, while we run the binary search on w_{g_1}, instead of computing the binomial probability mass functions exactly, we use Stirling's approximation, i.e. for every positive integer k, $k! \approx \left(\frac{k}{\mathbf{e}}\right)^k \cdot \sqrt{2\pi k}$, where $\mathbf{e}$ is the Euler's number. It is known that the Gamma function $\Gamma(z) = \int_0^\infty t^{z-1}\mathbf{e}^{-t}dt$ satisfies $\Gamma(k) = k!$. Taking logarithms converts the products in the definition of the binomial coefficient into sums of logs which are easier to handle and avoid overflow. Then we approximate each $\log\Gamma(\cdot)$ term with Stirling's expansion, $\log\Gamma(z) \approx z\log z - z + \frac{1}{2}\log(2\pi) + \frac{1}{12z} + O(z^{-3})$. In our practical implementation, we use the logarithm of Γ function to estimate the values of $\mathcal{E}(n_{\mathrm{g}_1}, d, w_{\mathrm{g}_1})$ and $\mathcal{E}(n_{\mathrm{g}_2}, d, w - w_{\mathrm{g}_1})$ for each value w_{g_1} in the binary search.

As a final note, we would like to reiterate that computing the column widths is performed *only once* during the preprocessing step and is not part of the update path. That being said, the preprocessing time complexity, which is quasi-linear, is comparable to that of common operations such as sorting and is generally acceptable in preprocessing settings. We further evaluate the cost of solving this optimization problem under various parameter settings in Section 6, showing that this cost is generally negligible.

5 Price of Fairness Analysis

Improving fairness usually comes at a cost, known as the "price of fairness" (aka cost of fairness) – the reduction in the overall performance as a result of resolving unfairness [17]. After introducing FCM and its details, in this section, we study its price of fairness. Following Equation 1, the overall performance of a CM sketch can be measured in terms of the sum of its expected additive error across all element types $\mathcal{U} = \{e_1, \cdots, e_n\}$. That is,

$$\mathcal{L}_{CM} = \sum_{i=1}^{n} \varepsilon_A(e_i) \tag{10}$$

Subsequently, we define the price of fairness (PoF) of FCM as the increase in its total expected additive error, compared to CM. That is,

$$PoF = \mathcal{L}_{FCM} - \mathcal{L}_{CM} \tag{11}$$

5.1 $d = 1$

In the following, we provide our price of fairness analysis for $d = 1$, proving that fairness even improves the expected performance, i.e., $PoF < 0$, when $d = 1$.

Lemma 5. *When $d = 1$, $PoF = \mathcal{L}_{FCM} - \mathcal{L}_{CM} < 0$.*

Proof of Lemma 5. Let us begin by computing $\mathcal{L}_{CM}$, the total additive error for the standard CM sketch. By Chapter 3 of [32], the expected additive error of an element type $e_j \in \mathcal{U}$ is

$$\varepsilon_A(e_j) = \frac{N - f(e_j)}{w}.$$

Then, the total additive error of the standard CM sketch is computed as

$$\mathcal{L}_{CM} = \sum_{j=1}^{n} \varepsilon_A(e_j) = \sum_{j=1}^{n} \frac{N - f(e_j)}{w} = n\frac{N}{w} - \frac{N}{w} = (n-1)\frac{N}{w} \tag{12}$$

Now, let us compute $\mathcal{L}_{FCM}$, the total additive error for the FCM sketch.

$$\mathcal{L}_{FCM} = \sum_{j=1}^{n} \varepsilon_A(e_j) = \sum_{e_j \in \mathrm{g}_1} \varepsilon_A(e_j) + \cdots + \sum_{e_j \in \mathrm{g}_\ell} \varepsilon_A(e_j)$$

Following the same calculation as in Equation (12), for every group $\mathrm{g} \in \mathcal{G}$,

$$\sum_{e_j \in \mathrm{g}} \varepsilon_A(e_j) = (n_{\mathrm{g}} - 1)\frac{N_{\mathrm{g}}}{w_{\mathrm{g}}}$$

Hence,

$$\mathcal{L}_{FCM} = \sum_{e_j \in \mathrm{g}_1} \varepsilon_A(e_j) + \cdots + \sum_{e_j \in \mathrm{g}_\ell} \varepsilon_A(e_j) = \sum_{\mathrm{g} \in \mathcal{G}} (n_{\mathrm{g}} - 1)\frac{N_{\mathrm{g}}}{w_{\mathrm{g}}}$$

From Theorem 1, we know, $w_{\mathrm{g}} = \frac{n_{\mathrm{g}}}{n} w, \forall \mathrm{g} \in \mathcal{G}$, while $\sum_{\mathrm{g} \in \mathcal{G}} N_{\mathrm{g}} = N$. Therefore,

$$\begin{aligned}
\mathcal{L}_{FCM} &= \sum_{\mathrm{g} \in \mathcal{G}} (n_{\mathrm{g}} - 1)\frac{N_{\mathrm{g}}}{w_{\mathrm{g}}} = \sum_{\mathrm{g} \in \mathcal{G}} (n_{\mathrm{g}} - 1)\frac{N_{\mathrm{g}}}{\frac{n_{\mathrm{g}}}{n} w} = \sum_{\mathrm{g} \in \mathcal{G}} n\left(1 - \frac{1}{n_{\mathrm{g}}}\right)\frac{N_{\mathrm{g}}}{w} \\
&= n \sum_{\mathrm{g} \in \mathcal{G}} \frac{N_{\mathrm{g}}}{w} - \sum_{\mathrm{g} \in \mathcal{G}} \frac{n}{n_{\mathrm{g}}} \cdot \frac{N_{\mathrm{g}}}{w} = n\frac{N}{w} - \sum_{\mathrm{g} \in \mathcal{G}} \frac{n}{n_{\mathrm{g}}} \cdot \frac{N_{\mathrm{g}}}{w}
\end{aligned} \tag{13}$$

Interestingly, combining Equations 11, 12, and 13, we show that $PoF < 0$, for a CM with random hashing scheme:

$$\begin{aligned}
PoF = \mathcal{L}_{FCM} - \mathcal{L}_{CM} &= n\frac{N}{w} - \sum_{\mathrm{g} \in \mathcal{G}} \left(\frac{n}{n_{\mathrm{g}}} \cdot \frac{N_{\mathrm{g}}}{w}\right) - (n-1)\frac{N}{w} \\
&= \frac{N}{w} - \sum_{\mathrm{g} \in \mathcal{G}} \frac{n}{n_{\mathrm{g}}} \cdot \frac{N_{\mathrm{g}}}{w} \\
&< \frac{N}{w} - \sum_{\mathrm{g} \in \mathcal{G}} \frac{N_{\mathrm{g}}}{w} \qquad \texttt{//since } \frac{n}{n_{\mathrm{g}}} > 1, \forall \mathrm{g} \in \mathcal{G} \\
&= \frac{N}{w} - \frac{N}{w} = 0
\end{aligned}$$

□

The negative PoF sounds counterintuitive, but as we shall further elaborate in our technical report [87], it can be explained by the difference between the random distribution of the elements versus their uniform distribution to the buckets. We further show that the PoF of FCM is zero for $d = 1$ when uniform hashing is used.

5.2 General Value of d

Next, we consider the price of fairness for multiple rows $d > 1$. The error of an element type $e_j \in \mathcal{U}$ is

$$\min_{p \in [d]} \sum_{e \neq e_j : \mathrm{h}_p(e) = \mathrm{h}_p(e_j)} f(e),$$

so the expected additive error is

$$\varepsilon_A(e_j) = \mathbb{E}\left[\min_{p\in[d]} \sum_{e\neq e_j : \mathrm{h}_p(e)=\mathrm{h}_p(e_j)} f(e)\right]. \tag{14}$$

For the standard CM sketch the total additive error is computed as $\mathcal{L}_{CM} = \sum_{j=1}^{n} \varepsilon_A(e_j)$ using Equation (14) to compute the sum considering w buckets. For the FCM sketch the total additive error is computed as $\mathcal{L}_{FCM} = \sum_{\mathrm{g}\in\mathcal{G}} \sum_{e_j\in\mathrm{g}} \varepsilon_A(e_j)$, where each sum $\sum_{e_j\in\mathrm{g}} \varepsilon_A(e_j)$ is computed using Equation (14) for w_{g} buckets. Since we do not have a closed form for the w_{g} values (for $d > 1$) from Equation (7), we cannot directly compute $PoF = \mathcal{L}_{CM} - \mathcal{L}_{FCM}$. Instead, we compute the PoF experimentally for real datasets in Section 6.6. Particularly, our experiments, including Figures 11 to 17 confirm the small PoF for $d > 1$.

6 Experiments

In addition to the theoretical analysis, we perform extensive experiments across a range of settings to validate the fairness as well as the efficiency of our proposed sketch. Consistent with the theoretical guarantees established in previous sections, the experimental results confirm the effectiveness and efficiency of our sketch in practical scenarios.

6.1 Evaluation Plan

We evaluate our proposed sketch based on three metrics 1) unfairness, 2) price of fairness and 3) efficiency. For each metric, we examine the impact of varying four parameters: the width of the sketch w, the depth of the sketch d, and the number of groups ℓ and the number of element types in the unpopular group n_{g_1}. Due to space constraints and the confirmed similarity of results across different datasets, for certain experiments, we present results for a single dataset per setting, with the full results available in the technical report [87]. Finally, we repeat each experiment five times and report the average values for each setting.

6.2 Datasets

For evaluation, we used five real-world datasets. To assess the scalability of our proposed sketch in large-scale scenarios, we selected five benchmark datasets (NYC Bus [91], DDoS [76], Reddit-Twitter [60], Census [66], and Google N-Grams [68]) that are sufficiently large to reflect real-world streaming environments. The dataset details are provided in the technical report [87].

6.3 Baselines

Although our theoretical analysis is based on comparisons with CM, we evaluate our method against four baseline approaches, including *representative state-of-the-art methods from both traditional and learning-based frequency estimation*, to comprehensively demonstrate its effectiveness:

CM [31]: The first baseline is the standard CM sketch described in Section 2.2, which we use across all of our experiments.

Row-Partitioning: The second baseline is the row-partitioning approach discussed in Section 3.3, which, as expected, does not consistently achieve equal group-wise approximation factors in practice. We use this baseline in a subset of our experiments to illustrate the negative result of row-partitioning.

Learned-CM [56]: The third baseline, learned-CM, enhance traditional CM by integrating ML to predict element frequencies. This allows more accurate estimates for rare or critical elements while preserving efficiency for common ones, reducing overall error in high-speed data streams.

A random sample comprising 30% of the stream is utilized to train a neural network designed to identify heavy hitters. We use this baseline across all of our experiments.

CM-CU [43]: The fourth baseline is a CM sketch with conservative update. It estimates element frequencies in a data stream by incrementing only the minimum counter for each element, which reduces overestimation caused by hash collisions. CM-CU has been shown to be **the most accurate** among traditional frequency estimation sketches [49] w.r.t. mean relative error. We use this baseline across all of our experiments.

We extend existing baseline implementations from the original papers or libraries and adapt them for our experiments.

6.4 Experiments Settings

6.4.1 Setup. The experiments were conducted on an Apple M2 Pro processor, 16 GB memory, running macOS. The algorithms were implemented in Python 3.12.[5]

6.4.2 Parameter Settings. In all experiments, the default sketch depth was fixed at $d = 5$. For experiments examining the impact of d, we vary its value from 2 to 10. The default sketch width w was selected on a per-dataset basis: $w = 64$ for the Census and DDoS datasets, $w = 512$ for the NYC Bus and Reddit–Twitter datasets, and $w = 65{,}536$ for the Google Ngram dataset. For experiments investigating the impact of w, we vary its value from 8 to 256 for the NYC Bus dataset, from 8 to 2,048 for the DDoS and Reddit–Twitter datasets, and from 1,024 to 1,048,576 for the Google Ngram dataset. The default number of groups is set to 2 across all experiments. In experiments examining the effect of the number of groups, this parameter is varied from 2 to 10. For ones evaluating the impact of the unpopular group size, the number of element types is increased from 310 K to 1.2 M.

Summary of results: In summary, FCM is the only approach observed to maintain near-zero unfairness–measured as the mean approximation-factor difference between groups–regardless of the sketch parameters or the underlying frequency distribution. This parity is typically achieved at a price that is generally small, though not universally so. All baseline methods generally exhibit some degree of bias in favor of the popular group. Moreover, FCM matches or exceeds state-of-the-art approaches in terms of construction and query times.

6.5 Validation: Unfairness Evaluation

In this experiment, we measure the impact of varying certain parameters on the unfairness of the sketch. Unfairness is measured as the difference between the expected approximation factor of the unpopular and popular groups, i.e., $\mathbb{E}[\alpha_{\mathrm{g}_1}] - \mathbb{E}[\alpha_{\mathrm{g}_2}]$. In the multi-group setting, unfairness is measured as the difference between the maximum and minimum mean approximation factors across all groups, i.e., $\min_{\forall \mathrm{g}_i \in \mathcal{G}}(\mathbb{E}[\alpha_{\mathrm{g}_i}]) - \max_{\forall \mathrm{g}_j \in \mathcal{G}}(\mathbb{E}[\alpha_{\mathrm{g}_j}])$. A sketch is considered group-fair if the unfairness value is zero.

Varying the number of columns w: We begin with the experiment that examines the impact of varying the sketch width w on unfairness. Looking at Figures 3, 4 and 5, we first observe that FCM achieves the smallest approximation-factor difference between the two groups, maintaining a value close to zero as w increases. Because n_{g_1} is relatively large across all datasets, the approximation factors for all groups and methods start out close to zero when w is small. For the standard CM baseline, as w increases, the approximation factor for the popular group gradually approaches 1, while remaining comparatively low for the unpopular group. Once w surpasses a certain threshold, it becomes large enough for the unpopular group's mean approximation factor to also approach 1.

[5]Code available at: https://github.com/neemashahbazi/FCM

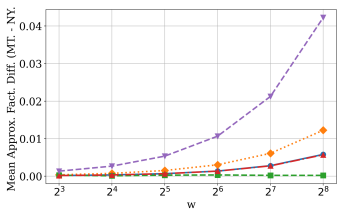


Fig. 3. effect of varying w on unfairness, NYC BUS, $d = 5$.

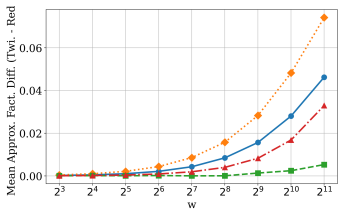


Fig. 4. effect of varying w on unfairness, REDDIT-TWITTER, $d = 5$.

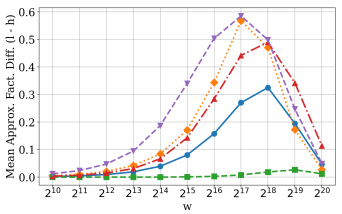


Fig. 5. effect of varying w on unfairness, GOOGLE NGRAM, $d = 5$.

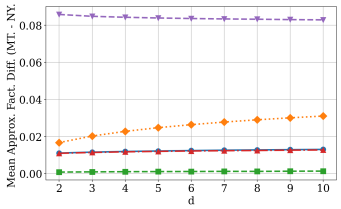


Fig. 6. effect of varying d on unfairness, NYC BUS, $w = 512$.

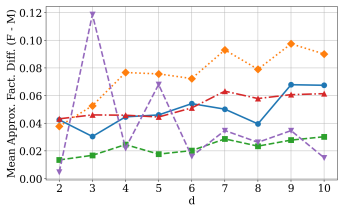


Fig. 7. effect of varying d on unfairness, CENSUS, $w = 64$.

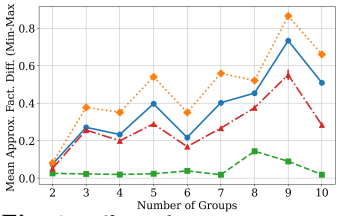


Fig. 8. effect of varying ℓ on unfairness, CENSUS, $w = 64, d = 5$.

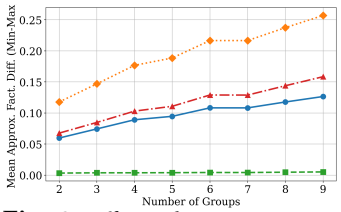


Fig. 9. effect of varying # ℓ on unfairness, GOOGLE NGRAM, $w = 65536$. $d = 5$.

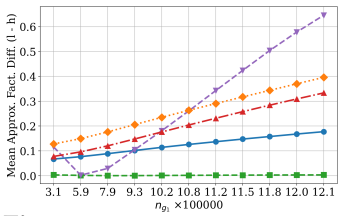


Fig. 10. effect of varying n_{g_1} on unfairness, GOOGLE NGRAM, $w = 65536, d = 5$.

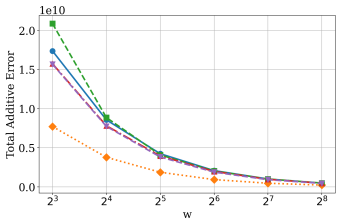


Fig. 11. effect of varying w on price of fairness, NYC BUS, $d = 5$.

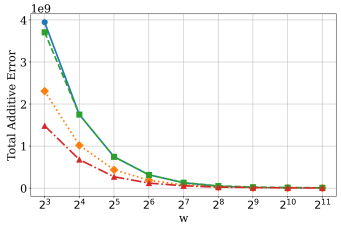


Fig. 12. effect of varying w on price of fairness, DDOS, $d = 5$.

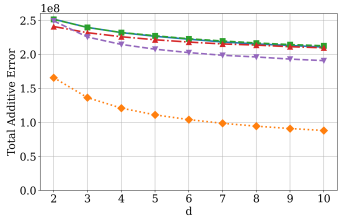


Fig. 13. effect of varying d on price of fairness, NYC BUS, $w = 512$.

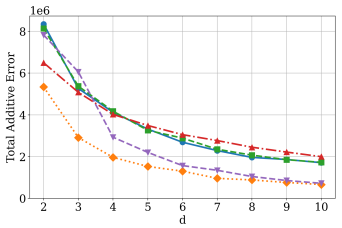


Fig. 14. effect of varying d on price of fairness, CENSUS, $w = 64$.

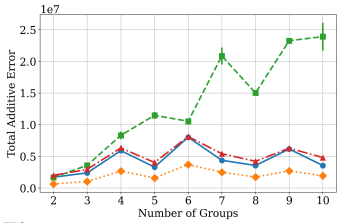


Fig. 15. effect of varying # of groups ℓ on price of fairness, CENSUS, $w = 64, d = 5$.

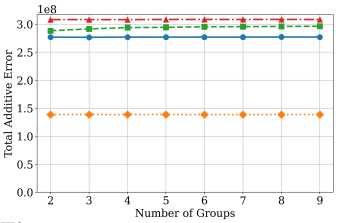


Fig. 16. effect of varying ℓ on price of fairness, GOOGLE NGRAM, $w = 65536, d = 5$.

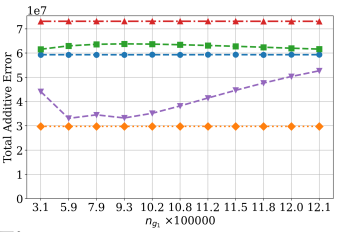


Fig. 17. effect of varying n_{g_1} on price of fairness, GOOGLE NGRAM, $w = 65536, d = 5$.

All the other baselines exhibit a trend similar to that of CM. Another key observation concerns the performance of Learned-CM: when the model successfully captures the heavy hitters, it outperforms the other baselines achieving better approximation factor difference. Interestingly, CM-CU—despite being shown to achieve the lowest mean relative error among traditional frequency-estimation methods—performs among the worst in terms of equalizing the approximation factor between the two groups across all datasets, indicating a bias toward the popular group. For the Row-Partitioning baseline, increasing w makes it harder to find suitable integral $d_{\mathbf{g}_1}$ values for allocating rows to the unpopular group, causing the approximation-factor difference to grow more rapidly than in the other baselines.

Varying the number of rows d: Next, we evaluate the impact of increasing the sketch depth d on unfairness. The results are presented in Figures 6 and 7. For the standard CM baseline, the sketch initially provides more accurate estimates (approximation factors closer to 1) for the popular group. As d increases, the accuracy improves for both groups, but at a slightly faster rate for the popular group. As in the previous experiment, Learned-CM achieves the best performance among the baselines in equalizing the approximation factor across groups, whereas CM-CU performs poorly, showing a steeper rate of decline as d increases. As before, FCM maintains near-zero unfairness regardless of the sketch depth. For the Row-Partitioning approach, we note some interesting observations. As shown in Figure 7, depending on the value of d and its divisibility, the unfairness fluctuates between nearly zero and a significant value as d increases. In this experiment, $n_{\mathbf{g}_1}$ and $n_{\mathbf{g}_2}$ are very close, so as long as the sketch can assign equal $d_{\mathbf{g}_1}$ and $d_{\mathbf{g}_2}$ values to each group, it performs well w.r.t keeping unfairness at zero. This occurs only when d takes even values.

Varying the number of groups ℓ: In our next experiment on unfairness, we investigate the impact of increasing the number of groups. The results are shown in Figures 8 and 9. Having demonstrated that the Row-Partitioning baseline performs inconsistently and is highly sensitive to the value of d, we henceforth compare our approach only with the other baselines. As expected, FCM maintains zero unfairness, independent of the number of groups or the grouping strategy used. However, for the standard CM, an increase in the number of groups generally leads to higher unfairness. This is particularly noticeable in the case of equi-width frequency-based grouping, where the sketch is more accurate for smaller groups with high-frequency elements and less accurate for those larger groups with low-frequency elements (Figure 9). In the case of arbitrary grouping, unfairness is entirely dependent on the size of each group. As shown in Figure 8, the number of groups is determined based on a different attribute with specific cardinality at each iteration. While the general trend is an increase in unfairness as the number of groups grows, in some iterations (e.g., 5 and 10), the unfairness values are closer, indicating that the groups are more evenly sized. In contrast, for iteration 9, one group is significantly larger or smaller than the others, resulting in higher unfairness. Learned-CM and CM-CU baselines follow a trend similar to the standard CM and are generally consistent with the results observed in previous experiments.

Varying unpopular group size $n_{\mathbf{g}_1}$: Next, we examine a specific scenario in which groups are divided into low-frequency ($\mathbf{g}_1$) and high-frequency ($\mathbf{g}_2$) elements based on the frequency of unique elements in the stream. Our goal is to evaluate the impact of varying the number of element types $n_{\mathbf{g}_1}$ in the unpopular group on unfairness. To do so, we vary the cutoff threshold on the frequency of each element: elements with frequencies below the threshold are assigned to group $\mathbf{g}_1$, while those above it are assigned to group $\mathbf{g}_2$. As the threshold increases, $n_{\mathbf{g}_1}$ grows while $n_{\mathbf{g}_2}$ decreases, keeping the total number of elements n constant. Let us now focus on Figure 10. As $n_{\mathbf{g}_1}$ increases, the standard CM sketch exhibits greater unfairness because the expected approximation factor for group $\mathbf{g}_1$ decreases as the group grows larger, while group $\mathbf{g}_2$ becomes smaller. This is

due to the standard CM sketch favoring more accurate estimates for the smaller group $\mathbf{g}_2$, which contains high-frequency elements, over the larger group $\mathbf{g}_1$, which consists of numerous low-frequency elements. For the Row-Partitioning baseline, the optimal integral values of $d_{\mathbf{g}_1}$ and $d_{\mathbf{g}_2}$ selected by the algorithm gradually deviate from the optimal fractional values determined by the equality 9. Specifically, with $d = 5$, the sketch would ideally assign a value $4 < d_{\mathbf{g}_1} < 5$ (closer to 5) to group $\mathbf{g}_1$, which is not feasible. As a result, we observe a significant disparity between the approximation factors of the groups for larger values of $n_{\mathbf{g}_1}$. On the other hand, the FCM maintains the approximation factor difference near zero, regardless of the value of $n_{\mathbf{g}_1}$. For the other baselines, Learned-CM starts out comparable to CM when $n_{\mathbf{g}_1}$ is small but diverges more sharply as $n_{\mathbf{g}_1}$ increases. As before, CM-CU delivers the poorest performance among traditional methods in equalizing the mean approximation factor.

Varying sample size for estimating $n_{\mathbf{g}_1}$: Finally, we evaluate the effect of increasing the sample size used to estimate the unpopular group size $n_{\mathbf{g}_1}$ on the resulting unfairness. Our first observation is that, for the DDOS dataset shown in Figure 31, the unfairness remains essentially unchanged as the sample size grows. This behavior arises because the estimate obtained from the sample extrapolates accurately to the full data stream. However, this is not universally the case. As illustrated in Figures 32, increasing the sample size leads to a more accurate estimation of $n_{\mathbf{g}_1}$, which in turn reduces the overall unfairness. Although the magnitude of this improvement is dataset dependent, it is generally modest. These results suggest that, for a reasonable error tolerance, estimates derived from relatively small samples are typically sufficient for constructing the FCM.

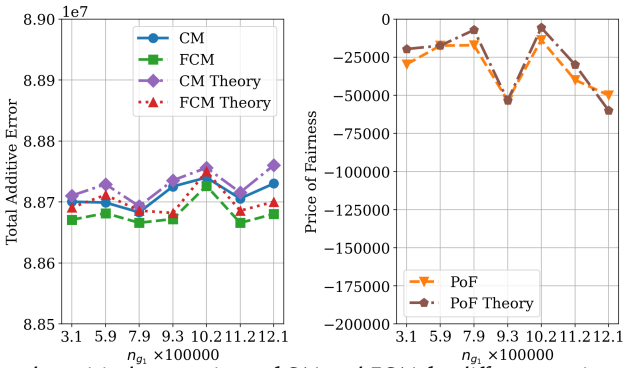


Fig. 18. Theoretical and empirical comparison of CM and FCM for different $n_{\mathbf{g}_1}$ in GOOGLE NGRAM for $d = 1$.

6.6 Price of Fairness Evaluation

Thus far, we have established that FCM guarantees group-fairness regardless of the parameters involved. In the next set of experiments, we evaluate the cost of enforcing this constraint relative to the standard CM and other baselines. Following the discussion in Section 5 and Equation 11, we measure the price of fairness in terms of the total additive error required to achieve fairness.

First, let us empirically validate the interesting result that PoF is negative for $d = 1$, proven in Section 5. To do so, we use a GOOGLE NGRAM dataset where we vary the unpopular group size n_l from 310K to 1.2M. The price of fairness is computed as the difference in total additive error between FCM and standard CM sketches (PoF = $\mathcal{L}_{\text{FCM}} - \mathcal{L}_{\text{CM}}$). The results, along with the empirical and theoretical additive errors shown in Figure 18, emphasize the superiority of FCM over standard CM in terms of additive error when $d = 1$. To further evaluate the price of fairness for $d > 1$ across various settings, similar to the previous section, we varied (a) number of columns (Figure 12 and 11), (b) number of rows (Figures 13 and 14), and (c) the number of groups (Figures 15 and 16) and (d)

group size (Figure 17), and measured the total additive error for CM versus FCM. As expected, FCM and all other baselines exhibit a decrease in the total additive error as the sketch width or depth increases. Moreover, depending on the grouping and the number of element types within each group, FCM may experience an increase in total additive error as the number of groups grows, since some groups may be relatively small or highly skewed, making it more difficult to enforce the equality constraints on the mean approximation factor (Figure 15). However, in cases where the groups are not strongly skewed, the additive error appears to be largely independent of the number of groups (Figure 16). Next, as the group size n_{g_1} increases, FCM exhibits a trend of slight initial increase followed by a decrease in the price of fairness (17). Overall, across all settings, the results were consistent, indicating that FCM achieves group fairness with costs that are generally small but can vary depending on the setting.

Finally, we examine how the price of fairness varies with the sample size used to estimate n_{g_1}. The results are presented in Figures 33 and 34. For the DDOS dataset, since the level of unfairness remains unchanged as the sample size increases, the price of fairness exhibits a similar stable trend. In contrast, for the NYC BUS dataset, increasing the sample size leads to a higher price of fairness. This occurs because, as the sample size increases, the estimation accuracy of n_{g_1} for the unpopular group typically improves, resulting in bucket assignments that are closer to the exact solution. Consequently, unfairness decreases, but this reduction comes at the expense of an increased price of fairness. Due to the space limitations, extensive results are moved to the technical report [87]. We confirm that we observed similar results across all settings and all other datasets.

6.7 Efficiency Evaluation

Having established the efficacy of the FCM sketch, we now turn our attention to its efficiency. A frequency-estimation sketch involves two main stages: (1) construction and (2) query time. In the following, we evaluate the impact of varying key parameters on the time required for each stage and provide a comparison with the standard CM sketch:

6.7.1 Construction Time. We define the construction time as the duration required to initialize the sketch and insert the entire stream into it. In summary, the construction time of FCM is comparable to that of CM and slightly faster than CM-CU. Learned-CM is considerably slower due to the additional time required for model training and inference. For all sketches, the construction time is independent of the sketch width w (Figures 19 and 20), the number of groups ℓ (Figure 23), and the unpopular group size n_{g_1} (Figure 24); varying these parameters has no impact on construction time. The construction time for both sketches grows linearly with the sketch depth d, as expected, since it corresponds to the number of times each element is inserted into the sketch. This result is presented in Figures 21 and 22.

Optimization Time: Finally, as part of the preprocessing analysis, we examine the impact of varying key parameters on the optimization time, defined as the time required to solve Equation 7 to compute the column widths following the procedure described in Section 4. We first observe that the optimization times are generally negligible and are dominated by other preprocessing steps, such as populating the sketch. Second, the optimization time increases with the value of w, due to the expansion of the binary search range. The optimization time is largely independent of the value of d. Furthermore, the optimization time remains consistently unchanged as the value of n_{g_1} is increased. Finally, we observe that the optimization time scales linearly with the number of groups ℓ. Representative results for each setting are shown in Figures 35, 36, 37, and 38.

6.7.2 Query Time. Query time is defined as the time required for the sketch to perform an estimate operation for a given element. The query time results exhibit similar patterns to the construction times, with FCM, CM and CM-CU showing comparable values and trends. Learned-CM can be

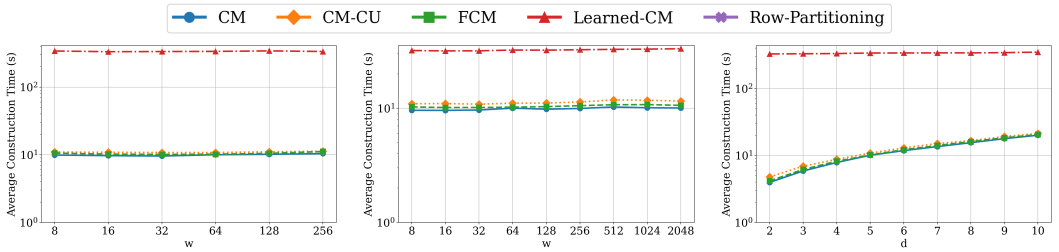


Fig. 19. effect of varying w on construction time, NYC BUS, $d = 5$.

Fig. 20. effect of varying w on construction time, DDOS, $d = 5$.

Fig. 21. effect of varying d on construction time, NYC BUS, $w = 512$.

Fig. 22. effect of varying d on construction time, DDOS, $w = 64$.

Fig. 23. effect of varying ℓ on construction time, GOOGLE NGRAM, $w = 65536$, $d = 5$.

Fig. 24. effect of varying n_{g_1} on construction time, GOOGLE NGRAM, $w = 65536$, $d = 5$.

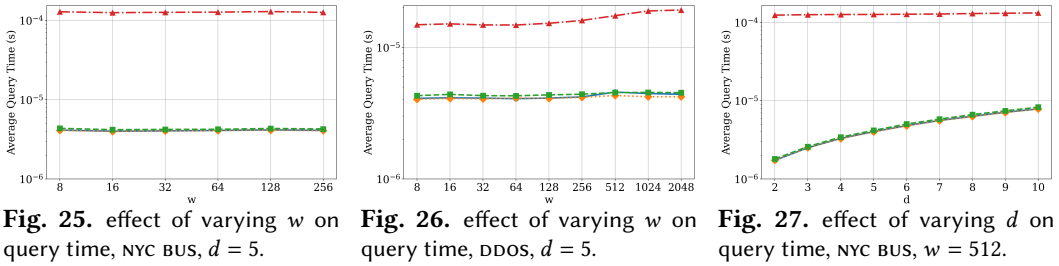


Fig. 25. effect of varying w on query time, NYC BUS, $d = 5$.

Fig. 26. effect of varying w on query time, DDOS, $d = 5$.

Fig. 27. effect of varying d on query time, NYC BUS, $w = 512$.

Fig. 28. effect of varying d on query time, DDOS, $w = 64$.

Fig. 29. effect of varying ℓ on query time, GOOGLE NGRAM, $w = 65536$, $d = 5$.

Fig. 30. effect of varying n_{g_1} on query time, GOOGLE NGRAM, $w = 65536$, $d = 5$.

orders of magnitude slower due to the additional model-inference time required during querying. The query times are independent of the sketch width w (Figures 25 and 26), the number of groups ℓ (Figure 29), and the unpopular group size n_{g_1} (Figure 30). Similarly, the query time for both sketches also increases linearly with the depth d, as shown in Figures 27 and 28.

6.8 Validating Group Columns Allocation

In our final experiment, we validate our mathematical analysis in Section 3.2, using a Monte Carlo method for estimating the value of w_{g_1} – the number of columns allocated to the group g_1.

We note that instead of the mathematical derivation, one could develop the following Monte Carlo method based on repeated sampling to estimate the value of w_{g_1}:

Consider n_{g_1}, n_{g_2}, w, d, and the parameters of arbitrary distributions from which the frequencies of groups g_1 and g_2 are drawn. For each group $g \in \{g_1, g_2\}$, we draw n_g samples from the corresponding

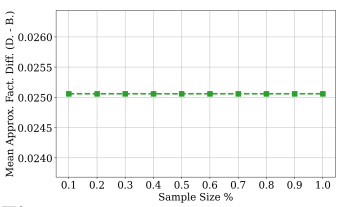


Fig. 31. effect of varying sample size for estimating n_{g_1} on unfairness, DDOS, $w = 64, d = 1$.

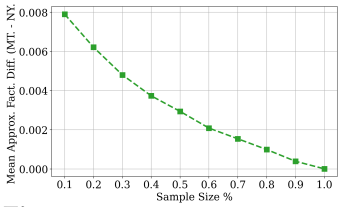


Fig. 32. effect of varying sample size for estimating n_{g_1} on unfairness, NYC BUS, $w = 512, d = 5$.

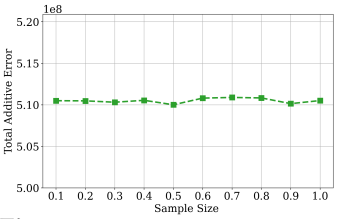


Fig. 33. effect of varying sample size for estimating n_{g_1} on price of fairness, DDOS, $w = 64, d = 1$.

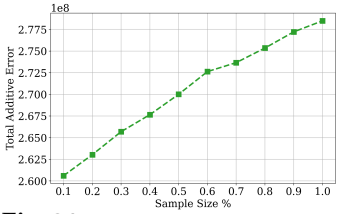


Fig. 34. effect of varying sample size for estimating n_{g_1} on price of fairness, NYC BUS, $w = 512, d = 5$.

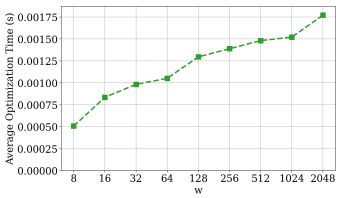


Fig. 35. effect of varying w on optimization time, DDOS, $d = 5$.

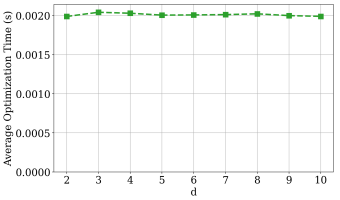


Fig. 36. effect of varying d on optimization time, CENSUS, $w = 64$.

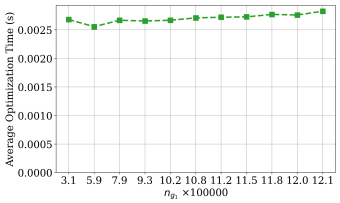


Fig. 37. effect of varying n_{g_1} on optimization time, GOOGLE NGRAM, $w = 65536, d = 5$.

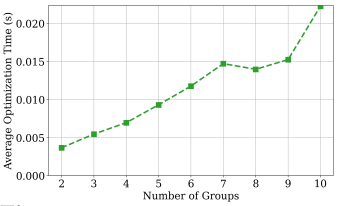


Fig. 38. effect of varying ℓ on optimization time, CENSUS, $w = 64, d = 5$.

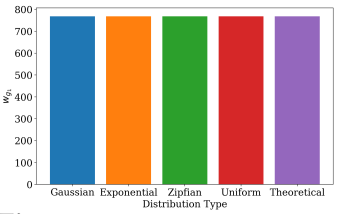


Fig. 39. Monte Carlo-based estimate of w_{g_1} vs. theory; depends on n_{g_1} and independent of distribution of group frequencies.

distribution to serve as the frequencies of the elements. As a result, we obtain a list of n tuples of the form `(frequency, group)`. Next, we perform a binary search over the range of w to find the optimal value of w_{g_1} that minimize the empirical difference in approximation factors between the two groups, where the empirical difference is evaluated by inserting the sampled frequencies into a FCM sketch with 1 row and measuring the approximation factors for each group. This procedure is repeated d times (once per sketch row), and we compute the average of the resulting w_{g_1} values. Finally, we repeat the entire process over multiple iterations and report the overall average of w_{g_1}.

We use this Monte Carlo method to estimate w_{g_1} under a range of synthetic distributions. We note from our analysis that the value of w_{g_1} does not depend on the distribution of data. Hence, we expect to observe similar results from different runs of the Monte Carlo estimation. We then compare the number of columns assigned to each group by this method to the theoretical values derived from the analysis in Section 3.2. The results are shown in Figure 39. The values of w_{g_1} were identical across different frequency distributions and, as expected, depended solely on the values of n_{g_1} and n_{g_2}.

7 Related Work

Count-Min sketch. The CM sketch was first described by Cormode and Muthukrishnan [30, 31]. It uses hash functions to map events to frequencies, but unlike a hash table uses only sub-linear space, at the expense of overcounting some events due to collisions. Count–min sketch is an alternative

to count sketch [24] and AMS sketch [6] and can be considered an implementation of a counting Bloom filter [44, 51] or multistage-filter [30]. Over the years multiple variations of min-count sketches have been proposed in the literature [46, 65, 75, 79, 94] improving the estimation or the performance under different settings. While CM sketches have been thoroughly studied there is still room for improvement. For example, the random hashing procedure may induce substantial uncertainty in the estimation, especially for low-frequency tokens. Furthermore, it is often the case that there exists an a priori knowledge on the data, and therefore it may be desirable to incorporate such a knowledge into the estimates

Recently, machine learning is used to resolve these issues. More specifically, machine learning techniques [1, 21, 36, 37, 56, 72, 100] are used to learn either the hash functions or a partition of the elements into two groups (based on their frequency) to improve the estimations or the performance compared to the traditional CM sketch. Interestingly, partitioned-learned CM sketches [56, 72] learn a partition of the elements into low frequency and high frequency elements. While this partition is the same as the one we used in our analysis, our objective is orthogonal to their problem. They do not necessarily aim to achieve the same estimation error between low and high frequency elements. Instead, they process light and heavy elements separately trying to improve the overall estimation error. To the best of our knowledge, none of these traditional and learned schemes can handle fair count-min estimations with theoretical guarantees. Finally, learning has been used to obtain other data structures as well, such as B-trees [61] or bloom filters [61, 69, 95].

Fairness. Fairness in data-driven systems has been studied by various research communities but mostly in machine learning (ML) [8, 16, 67, 74, 92]. Most of the existing work is on training a ML model that satisfies some fairness constraints. Some pioneering fair-ML efforts include [22, 23, 40, 45, 52, 59, 97]. Biases in data has also been studied extensively [11, 12, 42, 70, 73, 80, 84, 85] to ensure data has been prepared responsibly [71, 81, 82, 88]. Recent studies of fair algorithm design include fair clustering [4, 18, 20, 26, 27, 55, 64, 83], fairness in resource allocation and facility location problem [19, 38, 48, 53, 57, 102], min cut [62], max cover [9], set cover [33], game theoretic approaches [5, 39, 101], hiring [7, 77], ranking [10, 89, 90, 98], recommendation [25, 63, 93], representation learning [54], generative AI [28, 41, 47, 58], etc.

Fairness in data structures is significantly under-studied, with the existing work being limited to [3, 13–15, 34, 86]. Aumuller et al. [13–15] study individual fairness in near-neighbor search, while Shahbazi et al. [86] and Dehghankar et al. [34] study group fairness in hashing and ε-nets, respectively. In prior works, the objectives are fundamentally different, so these techniques cannot be applied to our sketch.

8 Discussions

8.1 Detecting Heavy Hitters

CM sketches have a wide range of range of applications including heavy hitters detection to detecting rare words (infrequent items that are not stop-word) in NLP settings [50], to AQP [31].

Still, identifying heavy hitters is known as one of the traditional applications of CM sketches. For heavy hitters detection, FCM performs equally to the standard CM. A heavy hitter in the standard sketch remains a heavy hitter in FCM. This is because CM sketches, including FCM, *always* overestimate the frequencies. As a result, FCM *never misses* a heavy hitter due to its fairness mechanism. Compared to the standard CM sketches, FCM focuses on stronger multiplicative error guarantees. In a standard CM sketch, the frequencies of a rare group (e.g., average frequency 10) could significantly get overestimated (e.g., 10,000) due to the collision with heavy hitters. FCM strictly equalizes this relative error across different groups. It is important to note that, while providing a stronger guarantee, *FCM does not degrade the total error.* Specifically, the total additive error (sum of all additive errors) and average additive errors *remain the same* across the standard

CM and FCM. Moreover, in cases where the groups are not correlated with frequencies, the average additive errors of groups (including their heavy hitters) do not change.

8.2 Intersectional Fairness

In cases where groups overlap –for example, race and gender– groups are defined at the intersection level (e.g., `Black-female, Black-male`). Handling the intersections in this way allows our method to cover overlapping group structures. While doing so increases the number of groups and therefore increases the price of fairness, satisfying fairness at the intersection level automatically satisfies fairness at the higher-level group definitions.

8.3 Dynamic Settings

a) Dynamic frequency patterns: The design of FCM does not depend on the frequency distributions (within or across groups). As a result, a drift in the frequency distributions will not require any update in the sketch. That is, *FCM can be readily adapted for popular dynamic settings, where frequency patterns change.* This was also confirmed in our experiments (Figure 39), where different frequency distributions do not affect the sketch configuration. Even if the frequency distributions drift – either in total or within individual groups – the setting of the sketch does not change.

b) Dynamic universe of element types: As stated in Section 3.5, our approach assumes a fixed universe model, where the universe of element types $\mathcal{U}$ is known. This is a limitation of our results that require further study in future work.

Still, in the following, we discuss an extension of our approach for a dynamic universe case, based on a *bookkeeping* approach that comes at an *extra memory cost* and a small amount of unfairness.

Specifically, we note that in real-world scenarios, often small amounts of unfairness is tolerable. That is, given a threshold $\epsilon > 0$, a sketch needs to guarantee its unfairness is at most ϵ. Recall from Section 6.5, that unfairness is measured as the difference between the expected approximation factor of the groups. Hence, the fairness requirement can be formulated as

$$\min_{\forall \mathrm{g}_i \in \mathcal{G}} (\mathbb{E}[\alpha_{\mathrm{g}_i}]) - \max_{\forall \mathrm{g}_j \in \mathcal{G}} (\mathbb{E}[\alpha_{\mathrm{g}_j}]) \leq \epsilon.$$

Following this observation, every time the universe of the element types get updated, we evaluate its unfairness guarantees, based on Lemmas 1 and 2, while using the *updated values* of n and $n_{\mathrm{g}}, \forall \mathrm{g} \in \mathcal{G}$. In particular, based on Lemma 1, for $d = 1$, so long as the fractions $\frac{n_{\mathrm{g}}}{n}$ do not change, the unfairness remains zero, and it increases based on $\max |\frac{n_{\mathrm{g}}^{old}}{n^{old}} - \frac{n_{\mathrm{g}}^{new}}{n^{new}}|$.

Once the unfairness of the sketch surpasses the threshold ϵ, we introduce a new FCM sketch based on the new element type counts. However, we cannot remove the old sketch, since we no longer have access to the exact element frequencies. Therefore, to address this issue, we need to *keep* the previous sketches.

Every time a new element in observed in the stream, we only update its corresponding counts in the latest sketch. However, the frequency of an element type e should be calculated as the sum of its frequencies across all sketches. Specifically, let $\{FCM_1, \cdots, FCM_k\}$ be the set of populated sketches so far. The estimated frequency of an element type $e \in \mathcal{U}$ is: $\hat{f}(e) = \sum_{i=1}^{k} FCM_i[e]$.

9 Conclusion

In this paper we introduced FCM, a new frequency estimation framework that ensures equal expected approximation factors across different groups, addressing a key fairness limitation of traditional CM sketches. By proposing a fairness-preserving design—column partitioning with group-aware hashing, our work offers strong theoretical guarantees and validates its effectiveness through extensive experiments. Future research directions include extending FCM integrating learned hash functions for further efficiency gains, and exploring fairness notions beyond group-wise guarantees, such as individual fairness in streaming data sketches.

References

[1] Anders Aamand, Piotr Indyk, and Ali Vakilian. 2019. frequency estimation algorithms under Zipfian distribution. *arXiv preprint arXiv:1908.05198* (2019).

[2] Anders Aamand, Piotr Indyk, and Ali Vakilian. 2019. (Learned) Frequency Estimation Algorithms under Zipfian Distribution. *arXiv preprint arXiv:1908.05198* (2019).

[3] Rishi Advani, Abolfazl Asudeh, Mohsen Dehghankar, and Stavros Sintos. 2026. Dynamic Necklace Splitting. *International Conference on Database Theory* (2026).

[4] Sara Ahmadian, Alessandro Epasto, Marina Knittel, Ravi Kumar, Mohammad Mahdian, Benjamin Moseley, Philip Pham, Sergei Vassilvitskii, and Yuyan Wang. 2020. Fair hierarchical clustering. *Advances in Neural Information Processing Systems* 33 (2020), 21050–21060.

[5] Encarnación Algaba, Vito Fragnelli, and Joaquín Sánchez-Soriano. 2019. The Shapley value, a paradigm of fairness. *Handbook of the Shapley value* (2019), 17–29.

[6] Noga Alon, Yossi Matias, and Mario Szegedy. 1996. The space complexity of approximating the frequency moments. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*. 20–29.

[7] Mohammad Reza Aminian, Vahideh Manshadi, and Rad Niazadeh. 2023. Fair markovian search. *Available at SSRN 4347447* (2023).

[8] Abolfazl Asudeh. 2021. Enabling responsible data science in practice. *ACM SIGMOD Blog* (2021).

[9] Abolfazl Asudeh, Tanya Berger-Wolf, Bhaskar DasGupta, and Anastasios Sidiropoulos. 2023. Maximizing coverage while ensuring fairness: A tale of conflicting objectives. *Algorithmica* 85, 5 (2023), 1287–1331.

[10] Abolfazl Asudeh, HV Jagadish, Julia Stoyanovich, and Gautam Das. 2019. Designing fair ranking schemes. In *Proceedings of the 2019 international conference on management of data*. 1259–1276.

[11] Abolfazl Asudeh, Zhongjun Jin, and HV Jagadish. 2019. Assessing and remedying coverage for a given dataset. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. IEEE, 554–565.

[12] Abolfazl Asudeh, Nima Shahbazi, Zhongjun Jin, and HV Jagadish. 2021. Identifying insufficient data coverage for ordinal continuous-valued attributes. In *Proceedings of the 2021 international conference on management of data*. 129–141.

[13] Martin Aumuller, Sariel Har-Peled, Sepideh Mahabadi, Rasmus Pagh, and Francesco Silvestri. 2021. Fair near neighbor search via sampling. *ACM SIGMOD Record* 50, 1 (2021), 42–49.

[14] Martin Aumüller, Sariel Har-Peled, Sepideh Mahabadi, Rasmus Pagh, and Francesco Silvestri. 2022. Sampling a Near Neighbor in High Dimensions—Who is the Fairest of Them All? *ACM Transactions on Database Systems (TODS)* 47, 1 (2022), 1–40.

[15] Martin Aumüller, Rasmus Pagh, and Francesco Silvestri. 2020. Fair near neighbor search: Independent range sampling in high dimensions. In *Proceedings of the 39th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems (PODS)*. 191–204.

[16] Solon Barocas, Moritz Hardt, and Arvind Narayanan. 2017. Fairness in machine learning. *Nips tutorial* 1 (2017), 2017.

[17] Solon Barocas, Moritz Hardt, and Arvind Narayanan. 2023. *Fairness and machine learning: Limitations and opportunities*. MIT press.

[18] Suman Bera, Deeparnab Chakrabarty, Nicolas Flores, and Maryam Negahbani. 2019. Fair algorithms for clustering. *Advances in Neural Information Processing Systems* 32 (2019).

[19] Víctor Blanco and Ricardo Gázquez. 2023. Fairness in maximal covering location problems. *Computers & Operations Research* 157 (2023), 106287.

[20] Matteo Böhm, Adriano Fazzone, Stefano Leonardi, and Chris Schwiegelshohn. 2020. Fair clustering with multiple colors. *arXiv preprint arXiv:2002.07892* (2020).

[21] Diana Cai, Michael Mitzenmacher, and Ryan P Adams. 2018. A Bayesian nonparametric view on count-min sketch. *Advances in neural information processing systems* 31 (2018).

[22] Flavio Calmon, Dennis Wei, Bhanukiran Vinzamuri, Karthikeyan Natesan Ramamurthy, and Kush R Varshney. 2017. Optimized pre-processing for discrimination prevention. *Advances in neural information processing systems* 30 (2017).

[23] L Elisa Celis, Lingxiao Huang, Vijay Keswani, and Nisheeth K Vishnoi. 2019. Classification with fairness constraints: A meta-algorithm with provable guarantees. In *Proceedings of the conference on fairness, accountability, and transparency*. 319–328.

[24] Moses Charikar, Kevin Chen, and Martin Farach-Colton. 2002. Finding frequent items in data streams. In *International colloquium on automata, languages, and programming*. Springer, 693–703.

[25] Jiawei Chen, Hande Dong, Xiang Wang, Fuli Feng, Meng Wang, and Xiangnan He. 2023. Bias and debias in recommender system: A survey and future directions. *ACM Transactions on Information Systems* 41, 3 (2023), 1–39.

[26] Flavio Chierichetti, Ravi Kumar, Silvio Lattanzi, and Sergei Vassilvitskii. 2017. Fair clustering through fairlets. *Advances in neural information processing systems* 30 (2017).

[27] Eden Chlamtáč, Yury Makarychev, and Ali Vakilian. 2022. Approximating fair clustering with cascaded norm objectives. In *Proceedings of the 2022 annual ACM-SIAM symposium on discrete algorithms (SODA)*. SIAM, 2664–2683.

[28] Zhibo Chu, Zichong Wang, and Wenbin Zhang. 2024. Fairness in large language models: A taxonomic survey. *ACM SIGKDD explorations newsletter* 26, 1 (2024), 34–48.

[29] Saar Cohen and Yossi Matias. 2003. Spectral bloom filters. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*. 241–252.

[30] Graham Cormode. 2009. *Count-Min Sketch*. Springer US, 511–516. doi:10.1007/978-0-387-39940-9_87

[31] Graham Cormode and Shan Muthukrishnan. 2005. An improved data stream summary: the count-min sketch and its applications. *Journal of Algorithms* 55, 1 (2005), 58–75.

[32] Graham Cormode and Ke Yi. 2020. *Small summaries for big data*. Cambridge University Press.

[33] Mohsen Dehghankar, Rahul Raychaudhury, Stavros Sintos, and Abolfazl Asudeh. 2025. Fair Set Cover. *KDD* (2025).

[34] Mohsen Dehghankar, Stavros Sintos, and Abolfazl Asudeh. 2026. On Fair Epsilon Net and Geometric Hitting Set. *Proceedings of the VLDB Endowment* 19, 5 (2026), 1032–1045.

[35] Fan Deng and Davood Rafiei. 2007. New estimation algorithms for streaming data: Count-min can do more. *Webdocs. Cs. Ualberta. Ca* (2007).

[36] Emanuele Dolera, Stefano Favaro, and Stefano Peluchetti. 2021. A Bayesian nonparametric approach to count-min sketch under power-law data streams. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 226–234.

[37] Emanuele Dolera, Stefano Favaro, and Stefano Peluchetti. 2023. Learning-augmented count-min sketches via Bayesian nonparametrics. *Journal of Machine Learning Research* 24, 12 (2023), 1–60.

[38] Kate Donahue and Jon Kleinberg. 2020. Fairness and utilization in allocating resources with uncertain demand. In *Proceedings of the 2020 conference on fairness, accountability, and transparency*. 658–668.

[39] Kate Donahue and Jon Kleinberg. 2023. Fairness in model-sharing games. In *Proceedings of the ACM Web Conference 2023*. 3775–3783.

[40] Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard Zemel. 2012. Fairness through awareness. In *Proceedings of the 3rd innovations in theoretical computer science conference*. ACM, 214–226.

[41] Sana Ebrahimi and Abolfazl Asudeh. 2025. A Survey on Reliability, Transparency, Accountability, and Fairness in LLM-based Multi-Agent Systems through the Responsibility Lens. (2025).

[42] Mahdi Erfanian, HV Jagadish, and Abolfazl Asudeh. 2024. Chameleon: Foundation Models for Fairness-Aware Multi-Modal Data Augmentation to Enhance Coverage of Minorities. *Proceedings of the VLDB Endowment* 17, 11 (2024), 3470–3483.

[43] Cristian Estan and George Varghese. 2002. New directions in traffic measurement and accounting. In *Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications*. 323–336.

[44] Li Fan, Pei Cao, Jussara Almeida, and Andrei Z Broder. 2000. Summary cache: a scalable wide-area web cache sharing protocol. *IEEE/ACM transactions on networking* 8, 3 (2000), 281–293.

[45] Michael Feldman, Sorelle A Friedler, John Moeller, Carlos Scheidegger, and Suresh Venkatasubramanian. 2015. Certifying and removing disparate impact. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 259–268.

[46] Éric Fusy and Gregory Kucherov. 2023. Count-min sketch with variable number of hash functions: an experimental study. In *International Symposium on String Processing and Information Retrieval*. Springer, 218–232.

[47] Isabel O Gallegos, Ryan A Rossi, Joe Barrow, Md Mehrab Tanjim, Sungchul Kim, Franck Dernoncourt, Tong Yu, Ruiyi Zhang, and Nesreen K Ahmed. 2024. Bias and fairness in large language models: A survey. *Computational linguistics* 50, 3 (2024), 1097–1179.

[48] Pranay Gorantla, Kunal Marwaha, and Santhoshini Velusamy. 2023. Fair allocation of a multiset of indivisible items. In *Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. SIAM, 304–331.

[49] Amit Goyal, Hal Daumé III, and Graham Cormode. 2012. Sketch algorithms for estimating point queries in nlp. In *Proceedings of the 2012 joint conference on empirical methods in natural language processing and computational natural language learning*. 1093–1103.

[50] Amit Goyal, Jagadeesh Jagarlamudi, Hal Daumé III, and Suresh Venkatasubramanian. 2010. Sketching techniques for large scale NLP. In *Proceedings of the NAACL HLT 2010 Sixth Web as Corpus Workshop*. 17–25.

[51] Deke Guo, Yunhao Liu, Xiangyang Li, and Panlong Yang. 2010. False negative problem of counting bloom filter. *IEEE transactions on knowledge and data engineering* 22, 5 (2010), 651–664.

[52] Moritz Hardt, Eric Price, and Nati Srebro. 2016. Equality of opportunity in supervised learning. In *Advances in neural information processing systems*. 3315–3323.

[53] Yuzi He, Keith Burghardt, Siyi Guo, and Kristina Lerman. 2020. Inherent trade-offs in the fair allocation of treatments. *arXiv preprint arXiv:2010.16409* (2020).

[54] Yuzi He, Keith Burghardt, and Kristina Lerman. 2020. A geometric solution to fair representations. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*. 279–285.
[55] Sedjro Salomon Hotegni, Sepideh Mahabadi, and Ali Vakilian. 2023. Approximation Algorithms for Fair Range Clustering. In *International Conference on Machine Learning*. PMLR, 13270–13284.
[56] Chen-Yu Hsu, Piotr Indyk, Dina Katabi, and Ali Vakilian. 2019. Learning-based frequency estimation algorithms.. In *International Conference on Learning Representations*.
[57] Yanmin Jiang, Xiaole Wu, Bo Chen, and Qiying Hu. 2021. Rawlsian fairness in push and pull supply chains. *European Journal of Operational Research* 291, 1 (2021), 194–205.
[58] Pradeep Kamboj, Shailender Kumar, and Vikram Goyal. 2025. Mitigating Social Bias in Generative AI: A Comprehensive Review. *KSII Transactions on Internet & Information Systems* 19, 10 (2025).
[59] Faisal Kamiran and Toon Calders. 2012. Data preprocessing techniques for classification without discrimination. *Knowledge and Information Systems* 33, 1 (2012), 1–33.
[60] Jonas Koenig. 2025. Reddit-Blogspot-Twitter Dataset. https://huggingface.co/datasets/jonaskoenig/reddit-blogspot-twitter Accessed: 2025-09-16.
[61] Tim Kraska, Alex Beutel, Ed H Chi, Jeffrey Dean, and Neoklis Polyzotis. 2018. The case for learned index structures. In *Proceedings of the 2018 international conference on management of data*. 489–504.
[62] Jason Li, Danupon Nanongkai, Debmalya Panigrahi, and Thatchaphol Saranurak. 2023. Near-Linear Time Approximations for Cut Problems via Fair Cuts. In *Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. SIAM, 240–275.
[63] Yunqi Li, Hanxiong Chen, Shuyuan Xu, Yingqiang Ge, Juntao Tan, Shuchang Liu, and Yongfeng Zhang. 2022. Fairness in recommendation: A survey. *arXiv preprint arXiv:2205.13619* (2022).
[64] Yury Makarychev and Ali Vakilian. 2021. Approximation algorithms for socially fair clustering. In *Conference on Learning Theory*. PMLR, 3246–3264.
[65] Younes Ben Mazziane and Othmane Marfoq. 2024. Count-Min Sketch with Conservative Updates: Worst-Case Analysis. *arXiv preprint arXiv:2405.12034* (2024).
[66] Thiesson Bo Meek, Chris and David Heckerman. 2001. US Census Data (1990). UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5VP42.
[67] Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. 2021. A survey on bias and fairness in machine learning. *ACM computing surveys (CSUR)* 54, 6 (2021), 1–35.
[68] Jean-Baptiste Michel, Yuan Kui Shen, Aviva Presser Aiden, Adrian Veres, Matthew K Gray, Google Books Team, Joseph P Pickett, Dale Hoiberg, Dan Clancy, Peter Norvig, et al. 2011. Quantitative analysis of culture using millions of digitized books. *science* 331, 6014 (2011), 176–182.
[69] Michael Mitzenmacher. 2018. A model for learned bloom filters and optimizing by sandwiching. *Advances in Neural Information Processing Systems* 31 (2018).
[70] Melika Mousavi, Nima Shahbazi, and Abolfazl Asudeh. 2024. Data Coverage for Detecting Representation Bias in Image Datasets: A Crowdsourcing Approach. In *International Conference on Extending Database Technology (EDBT)*.
[71] Fatemeh Nargesian, Abolfazl Asudeh, and HV Jagadish. 2021. Tailoring data source distributions for fairness-aware data integration. *Proceedings of the VLDB Endowment* 14, 11 (2021), 2519–2532.
[72] Thuy Trang Nguyen and Cameron N Musco. [n. d.]. Partitioned-Learned Count-Min Sketch. ([n. d.]).
[73] Alexandra Olteanu, Carlos Castillo, Fernando Diaz, and Emre Kıcıman. 2019. Social data: Biases, methodological pitfalls, and ethical boundaries. *Frontiers in big data* 2 (2019), 13.
[74] Dana Pessach and Erez Shmueli. 2022. A review on fairness in machine learning. *ACM Computing Surveys (CSUR)* 55, 3 (2022), 1–44.
[75] Guillaume Pitel and Geoffroy Fouquier. 2015. Count-Min-Log sketch: Approximately counting with approximate counters. In *International Symposium on Web AlGorithms*.
[76] M Devendra Prasad, V Prasanta Babu, and C Amarnath. 2019. Machine learning ddos detection using stochastic gradient boosting. *Int. J. Comput. Sci. Eng* 7, 4 (2019), 157–16.
[77] Manish Raghavan, Solon Barocas, Jon Kleinberg, and Karen Levy. 2020. Mitigating bias in algorithmic hiring: Evaluating claims and practices. In *Proceedings of the 2020 conference on fairness, accountability, and transparency*. 469–481.
[78] Kavya Ravichandran. 2019. Learning-augmented frequency estimation with an online oracle.
[79] Ori Rottenstreich, Pedro Reviriego, Ely Porat, and S Muthukrishnan. 2021. Avoiding flow size overestimation in Count-Min sketch with Bloom filter constructions. *IEEE Transactions on Network and Service Management* 18, 3 (2021), 3662–3676.
[80] Babak Salimi, Bill Howe, and Dan Suciu. 2019. Data management for causal algorithmic fairness. *arXiv preprint arXiv:1908.07924* (2019).

[81] Babak Salimi, Bill Howe, and Dan Suciu. 2020. Database repair meets algorithmic fairness. *ACM SIGMOD Record* 49, 1 (2020), 34–41.
[82] Babak Salimi, Luke Rodriguez, Bill Howe, and Dan Suciu. 2019. Interventional fairness: Causal database repair for algorithmic fairness. In *Proceedings of the 2019 International Conference on Management of Data*. 793–810.
[83] Melanie Schmidt, Chris Schwiegelshohn, and Christian Sohler. 2020. Fair coresets and streaming algorithms for fair k-means. In *Approximation and Online Algorithms: 17th International Workshop, WAOA 2019, Munich, Germany, September 12–13, 2019, Revised Selected Papers 17*. Springer, 232–251.
[84] Nima Shahbazi, Mahdi Erfanian, and Abolfazl Asudeh. 2024. Coverage-based data-centric approaches for responsible and trustworthy ai. *A Quarterly bulletin of the IEEE Computer Society Technical Committee on Database Engineering* 48, 1 (2024).
[85] Nima Shahbazi, Yin Lin, Abolfazl Asudeh, and HV Jagadish. 2023. Representation Bias in Data: A Survey on Identification and Resolution Techniques. *Comput. Surveys* (2023).
[86] Nima Shahbazi, Stavros Sintos, and Abolfazl Asudeh. 2024. Fairhash: A fair and memory/time-efficient hashmap. *Proceedings of the ACM on Management of Data* 2, 3 (2024), 1–29.
[87] Nima Shahbazi, Stavros Sintos, and Abolfazl Asudeh. 2025. Fair-Count-Min: Frequency Estimation under Equal Group-wise Approximation Factor. *arXiv preprint arXiv:2505.18919* (2025).
[88] Suraj Shetiya, Ian P Swift, Abolfazl Asudeh, and Gautam Das. 2022. Fairness-aware range queries for selecting unbiased data. In *ICDE*. IEEE, 1423–1436.
[89] Ashudeep Singh and Thorsten Joachims. 2018. Fairness of exposure in rankings. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*. 2219–2228.
[90] Ashudeep Singh and Thorsten Joachims. 2019. Policy learning for fairness in ranking. *Advances in neural information processing systems* 32 (2019).
[91] Michael Stone. 2025. New York City Transport Statistics. https://www.kaggle.com/datasets/stoney71/new-york-city-transport-statistics/data Accessed: 2025-09-16.
[92] Julia Stoyanovich, Serge Abiteboul, Bill Howe, HV Jagadish, and Sebastian Schelter. 2022. Responsible data management. *Commun. ACM* 65, 6 (2022), 64–74.
[93] Ian P Swift, Sana Ebrahimi, Azade Nova, and Abolfazl Asudeh. 2022. Maximizing fair content spread via edge suggestion in social networks. *Proceedings of the VLDB Endowment* 15, 11 (2022), 2692–2705.
[94] Daniel Ting. 2018. Count-min: Optimal estimation and tight error bounds using empirical error distributions. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2319–2328.
[95] Kapil Vaidya, Eric Knorr, Michael Mitzenmacher, and Tim Kraska. 2020. Partitioned Learned Bloom Filters. In *International Conference on Learning Representations*.
[96] Xinyu Yuan, Yan Qiao, Meng Li, Zhenchun Wei, Cuiying Feng, Zonghui Wang, and Wenzhi Chen. 2024. Learning-based Sketches for Frequency Estimation in Data Streams without Ground Truth. *arXiv preprint arXiv:2412.03611* (2024).
[97] Muhammad Bilal Zafar, Isabel Valera, Manuel Gomez Rodriguez, and Krishna P Gummadi. 2017. Fairness beyond disparate treatment & disparate impact: Learning classification without disparate mistreatment. In *Proceedings of the 26th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 1171–1180.
[98] Meike Zehlike, Ke Yang, and Julia Stoyanovich. 2022. Fairness in ranking, part i: Score-based ranking. *Comput. Surveys* 55, 6 (2022), 1–36.
[99] Meifan Zhang, Sixin Lin, and Lihua Yin. 2023. Local differentially private frequency estimation based on learned sketches. *Information Sciences* 649 (2023), 119667.
[100] Meifan Zhang, Hongzhi Wang, Jianzhong Li, and Hong Gao. 2020. Learned sketches for frequency estimation. *Information Sciences* 507 (2020), 365–385.
[101] Yan Zhao, Kai Zheng, Jiannan Guo, Bin Yang, Torben Bach Pedersen, and Christian S Jensen. 2021. Fairness-aware task assignment in spatial crowdsourcing: Game-theoretic approaches. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE, 265–276.
[102] Liping Zhou, Na Geng, Zhibin Jiang, and Xiuxian Wang. 2019. Public hospital inpatient room allocation and patient scheduling considering equity. *IEEE Transactions on Automation Science and Engineering* 17, 3 (2019), 1124–1139.

Received October 2025; revised January 2026; accepted February 2026