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

Nima Shahbazi
University of Illinois Chicago
nshahb3@uic.edu

Stavros Sintos
University of Illinois Chicago
stavros@uic.edu

Abolfazl Asudeh
University of Illinois Chicago
asudeh@uic.edu

ABSTRACT

Frequency estimation in streaming data often relies on sketches like Count-Min (CM) to provide approximate answers with sub-linear space. However, CM sketches introduce additive errors that disproportionately impact low-frequency elements, creating fairness concerns across different groups of elements. We introduce Fair-Count-Min, a frequency estimation sketch that guarantees equal expected approximation factors across element groups, thus addressing the unfairness issue. 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 the price of fairness, and validate our theoretical findings through extensive experiments on real-world and synthetic datasets. Our experimental results show that Fair-Count-Min achieves fairness with minimal additional error and maintains competitive efficiency compared to standard CM sketches.

PVLDB Reference Format:
Nima Shahbazi, Stavros Sintos, and Abolfazl Asudeh. Fair-Count-Min: Frequency Estimation under Equal Group-wise Approximation Factor. PVLDB, 14(1): XXX-XXX, 2020.
doi:XX.XX/XXX.XX

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/neemashahbazi/Fair-Count-Min/.

1 INTRODUCTION

Estimating item 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. 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 elements 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 low-frequency elements. This imbalance gives rise to a fairness issue – an aspect overlooked in the existing work.

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 strategy 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 grouping of the elements, while the grouping can be based on element popularity or any other grouping strategy, e.g., based on the demographic groups.

FCM does not increase the memory requirement of the CM. 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. Interestingly, we show that 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.

In addition to theoretical analysis, we perform extensive experiments on multiple real-world and synthetic datasets to evaluate the performance of our proposed approach in practice. In summary, our experiments verify (a) while CM is usually not fair, FCM achieves fairness in all experiments, (b) the price of fairness is negligible, and (c) FCM has the same memory and time efficiency as CM.

Proceedings of the VLDB Endowment, Vol. 14, No. 1 ISSN 2150-8097.
doi:XX.XX/XXX.XX

Summary of 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 elements with low frequency. Using this notion of fairness, 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 the regular count-min sketch.
- 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 using real-world and synthetic datasets. The results demonstrate that FCM successfully achieves group fairness while incurring only a minor additional additive error, and it preserves the time and space efficiency characteristic of standard CM sketches.

Paper organization: The rest of the paper is organized as follows. In Section 2, we provide necessary preliminaries, introduce the fairness challenge in CM sketches, and formally define the group-fair frequency estimation problem. Section 3 details our FCM sketch, including its construction via group-aware semi-uniform hashing and fairness guarantees. Section 4 presents efficient algorithms for computing bucket allocations across groups. In Section 5, we theoretically analyze the price of fairness. Section 6 reports our experimental results, comparing FCM to standard CM and Row-Partitioning baselines. Finally, we conclude with a discussion of implications and future work.

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 $\mathcal{U} = \{e_1, e_2, \ldots, e_n\}$. 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.

Count-Min (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[1] 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 sketch 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 sketch exist, including the CM sketch with conservative updates [53], the Count-Mean-Min sketch [30], and other related frequency estimation structures such as the Count sketch [21] and Spectral Bloom Filters [25]. In this study, we focus exclusively on the CM sketch, leaving the exploration of these alternatives for future work.

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, high-frequency elements, it can be intolerable for the unpopular, low-frequency ones. In other words, the significance of the frequency estimation error is relative

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

to the frequency value. To further clarify this using a toy example, let us consider Example 1.

Example 1: *Consider a CM sketch with a hash function h that maps the events of types $\mathcal{U} = \{e_1, e_2, \ldots, e_n\}$ to $w = 200$ bins. Let $N = 10000$ be the number of elements in the data stream. Hence, the additive estimation error is bounded by $\varepsilon_A \leq 50$. Suppose the frequencies of the element types e_1 and e_2 are $f(e_1) = 1000$ and $f(e_2) = 10$. As a result, the expected frequency estimation is bounded by 1050 for e_1, resulting in a small error of only 5% of $f(e_1)$. On the other hand, this error is as high as 500% for e_2.*

From Example 1, 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 low-frequency element types. This motivates us to instead promote **approximation factor** as a stronger frequency estimation guarantee.

Definition 1 (Approximation Factor). *The approximation factor of count-min sketch CM for an element type e is $\alpha(e)$, if its estimation $\hat{f}(e)$ of the frequency of e satisfies the following condition:*

$$\frac{f(e)}{\hat{f}(e)} = \alpha(e).$$

Using Definition 1, we define a notion of *group fairness* provides equal multiplicative frequency estimation errors for various element groups.

Let each element type e be associated with a group $g \in \mathcal{G}$, where $\mathcal{G} = \{\mathbf{g}_1, \mathbf{g}_2, \ldots, \mathbf{g}_\ell\}$. For example, each group $\mathbf{g}_i$ might contain element types with comparable popularity. Then, a count-min sketch is called fair if it satisfies Definition 2.

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

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

.

Without loss of generality and to ease the explanations, we use the binary groups l (`low-frequency`) and h (`high-frequency`) 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.*

2.5 Solution Overview

A main issue that causes the unbounded multiplicative error in traditional count-min sketches is that high-frequency elements can collide with low-frequency ones, adding a major overestimation to their counts.

One way to reduce the chance of such collisions is by significantly increasing the number of bins w and/or the value of d, the number of independent hash functions used for the estimation, i.e., the number of rows in the sketch. This, however, has two major issues: first, given a low-frequency element e to ensure it is unlikely that a high-frequency element collides with it in at least one of the d rows, the size of the sketch increases in the order of n, the number of element types – losing its purpose. Second, even after increasing the size of the sketch, it cannot provide a guarantee of a multiplicative error as low and high-frequency elements can still collide.

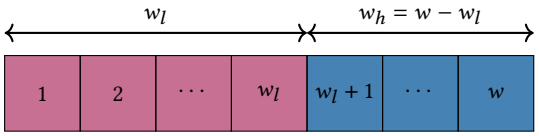

Figure 1: Illustration of a column-partitioning based group-fair min-count with one row, i.e., one hash function $\mathsf{h}(.)$.

Instead, to design group-fair min-count sketches, 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 low and high-frequency elements in Section 3. Proving that our sketch satisfies Definition 2, we study its price of fairness in Section 5.

3 FAIR-COUNT-MIN USING GROUP-AWARE SEMI-UNIFORM HASHING

Our first idea for generating group-fair count-min is to utilize "semi-uniform" hash functions that separate element types of low and high frequency. Specifically, reserving w_l bins for the `low-frequency` group l and $w_h = w - w_l$ for the `high-frequency` group h, the semi-uniform hash function $\mathsf{h}(.)$ assigns each low-frequency element to the first w_l bins and the others to the remaining w_h bins (Figure 1):

$$\mathsf{h} : \begin{cases} l \rightarrow [w_l] \\ h \rightarrow w_l + [w_h] \end{cases}$$

The key question we shall answer in the rest of this section is whether there exists a value w_l (hence $w_h = w - w_l$) for which a count-min sketch 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 $d = 2$ and finally to the general values of d.

3.1 $d = 1$

First, let us derive the expected approximation factor for an element e in the `low-frequency` group l. Let n_l be the number of element types in l.

Let C_i be the total frequency counts in each low-frequency bucket $i \in [w_l]$. That is,

$$C_i = \sum_{e_j : \mathsf{h}(e_j) = i} f(e_j)$$

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

$$\hat{f}(e_j) = \frac{f(e_j)}{\alpha(e_j)} = C_{\mathsf{h}(e_j)}.$$

Hence,

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

Therefore,

$$\begin{aligned}
\mathbb{E}[\alpha_l] &= \frac{1}{n_l}\sum_{e_j \in l} \alpha(e_j) = \frac{1}{n_l}\sum_{e_j \in l} \frac{f(e_j)}{C_{\mathsf{h}(e_j)}} \\
&= \frac{1}{n_l}\sum_{i=1}^{w_l} \sum_{e_j:\mathsf{h}(e_j)=i} \frac{f(e_j)}{C_i} && \texttt{//breaking the calculation per cell} \\
&= \frac{1}{n_l}\sum_{i=1}^{w_l} \frac{1}{C_i} \sum_{e_j:\mathsf{h}(e_j)=i} f(e_j) && \texttt{//factoring out the common term} \\
&= \frac{1}{n_l}\sum_{i=1}^{w_l} \frac{1}{C_i} C_i && \texttt{//replacing sum of frequencies with } C_i \\
&= \frac{1}{n_l}\sum_{i=1}^{w_l} 1 = \frac{w_l}{n_l} && (4)
\end{aligned}$$

Similarly, the expected approximation factor for the group h can be computed as

$$\mathbb{E}\big[\alpha_h\big] = \frac{w_h}{n_h} = \frac{w - w_l}{n - n_l}$$

To satisfy group fairness (Definition 2), we need to ensure that the expected group fairness for the two groups is the same. That is, $\mathbb{E}\big[\alpha_l\big] = \mathbb{E}\big[\alpha_h\big]$.

Hence, to ensure fairness, w_l is selected as follows:

$$\frac{w_l}{n_l} = \frac{w - w_l}{n - n_l} \Rightarrow\ w_l = \frac{n_l}{n} w \tag{5}$$

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

Theorem 1. *A Count-Min sketch with a group-aware semi-uniform hash function* $\mathsf{h}(.)$ *is group-fair, if the number of bins* l *allocated to each group is proportional to the ratio of element types from that group. That is,* $w_l = \frac{n_l}{n} w, \forall \mathsf{g}_l \in \mathcal{G}$.

Before moving to the larger values of d, i.e., count-min sketches with more number of hash functions, let us take a closer look at Theorem 1 and what it indicates. A regular min-count sketch 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, $w_l = \frac{n_l}{n} w, \forall l \in \mathcal{G}$. As a result, the number of elements in each bucket for each group l is:

$$\frac{n_l}{w_l} = \frac{n_l}{\frac{n_l}{n} w} = \frac{n}{w} \tag{6}$$

Interestingly, this means to ensure fairness, we need to *allocate enough bins to each group such that expected number of element types hashed to each bucket remains unchanged*, i.e., $\frac{n}{w}$. This helps us in extending d to more than one row by *equalizing the expected number of element types hashed in each bin* across various groups to ensure fairness.

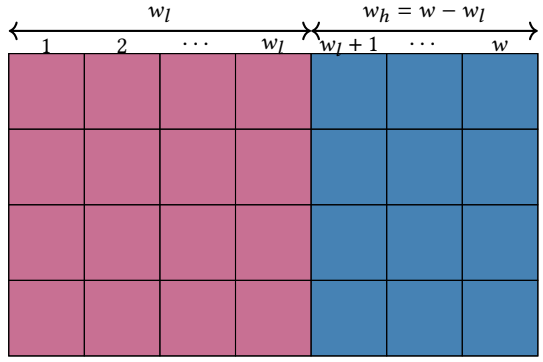

Figure 2: Illustration of a column-partitioning based group-fair min-count with d rows.

3.2 General Value of d

Having established that a count-min sketch 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.

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 $e_j \in l$ and a row i, let $\ell = \mathsf{h}_i(e_j)$. Let the random integer variable X_i be the size of the bucket of the element e at row i , i.e., $|\{e \in l | \mathsf{h}_i(e) = \ell\}|$. 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 $\frac{1}{w_l}$. As a result, X_i follows the binomial distribution $X_i \sim Bin\left(n_l, \frac{1}{w_l}\right)$:

$$\Pr(X_i = x) = \binom{n_l}{x} \cdot \left(\frac{1}{w_l}\right)^x \cdot \left(\frac{w_l - 1}{w_l}\right)^{n_l - x}$$

Let μ_l be the expected frequency for group l. Then, the expected frequency count of each bucket is μ_l 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 l. 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

$$X_1, \cdots, X_d \sim Bin\left(n_l, \frac{1}{w_l}\right)$$

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

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

We aim to find the expected value of Y:

$$\mathbb{E}[Y] = \sum_{x=1}^{n} xP(Y = x)$$

Since the domain of the random variable Y is the nonnegative integers $\{1, 2, \cdots\}$, the expected value can alternatively be computed using the following equation [38].

$$\mathbb{E}[Y] = \sum_{x=1}^{n} P(Y \geq x)$$

$$\Pr(Y \geq x) = \Pr(\min(X_1, \cdots, X_d) \geq x) = \Pr(X_1 \geq x, \cdots, X_d \geq x).$$

Since $X_1, \cdots, X_d$ are independent and identically distributed, the probability is computed as:

$$\begin{aligned}\Pr(Y \geq x) &= \Pr(X_1 \geq x, \cdots, X_d \geq x) \\ &= \prod_{i=1}^{d} \Pr(X_i \geq x) = \Pr(X \geq x)^d\end{aligned}$$

where $\Pr(X = x) = \binom{n_l}{x} \cdot \left(\frac{1}{w_l}\right)^x \cdot \left(\frac{w_l - 1}{w_l}\right)^{n_l - x}$. Hence, the probability $\Pr(X \geq x)$ is:

$$\Pr(X \geq x) = \sum_{i=x}^{n_l} \binom{n_l}{i} \left(\frac{1}{w_l}\right)^i \left(\frac{w_l - 1}{w_l}\right)^{n_l - i}.$$

Putting everything together, we obtain the expected value of Y as:

$$\begin{aligned}\mathbb{E}[Y] &= \sum_{x=1}^{n_l} \Pr(X \geq x)^d \\ &= \sum_{x=1}^{n_l} \left[\sum_{i=x}^{n_l} \binom{n_l}{i} \left(\frac{1}{w_l}\right)^i \left(\frac{w_l - 1}{w_l}\right)^{n_l - i}\right]^d. \end{aligned} \tag{7}$$

Finally, equalizing Equation 7 for the groups l and h, we specify the number of columns allocated to each group as w_l and $w_h = w - w_l$, by solving the Equation 8 ($n_h = n - n_l$):

$$\begin{aligned}&\mathbb{E}_{e \in l}[Y] = \mathbb{E}_{e \in h}[Y] \Rightarrow \\ &\sum_{x=1}^{n_l} \left[\sum_{i=x}^{n_l} \binom{n_l}{i} \left(\frac{1}{w_l}\right)^i \left(\frac{w_l - 1}{w_l}\right)^{n_l - i}\right]^d \\ &= \sum_{x=1}^{n-n_l} \left[\sum_{i=x}^{n-n_l} \binom{n - n_l}{i} \left(\frac{1}{w - w_l}\right)^i \left(\frac{w - w_l - 1}{w - w_l}\right)^{n - n_l - i}\right]^d \end{aligned} \tag{8}$$

Later in Section 4, we shall provide an efficient algorithm for solving Equation 8.

3.3 Negative Result: Row Partitioning Baseline

FCM follows a column partitioning approach, using the group-aware semi-uniform hashing schemes. Alternatively, one can partition the rows 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 (Figure 3).

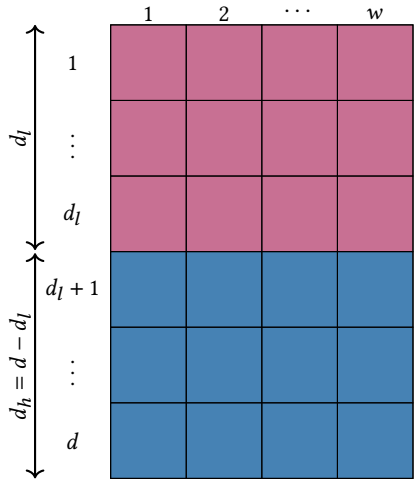

Figure 3: Illustration of the row-partitioning baseline.

Following Equation 7, the expected size of the minimum-size bucket across the d' rows, w' columns, and a group with n' elements can be computed as,

$$\mathcal{E}(n', d', w') = \sum_{x=1}^{n'} \left[\sum_{i=x}^{n'} \binom{n'}{i} \left(\frac{1}{w'}\right)^i \left(\frac{w' - 1}{w'}\right)^{n' - i}\right]^{d'} \tag{9}$$

Let d_l and $d_h = d - d_l$ be the number of rows allocated to the groups l and h. Then, we can find the proper values of d_l and d_h that achieve fairness by solving the following equation:

$$\mathcal{E}(n_l, d_l, w) = \mathcal{E}(n_h, d_h, w) \tag{10}$$

There, however, is a major issue making this approach unlikely to work: unlike w, which is a relatively large number, d is a small constant (usually in the order of tens). As a result, there are only a small, constant number of choices for $d_l \in [d]$. This, in practice, makes it unlikely for d to be divisible such that Equation 10 is satisfied. We shall verify this negative result in our experiments in Section 6.

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 $\mathbf{g}_l \in \mathcal{G}$, transform the problem into the binary grouping by merging all other groups as the super-group "others". Then, for $d = 1$, following Theorem 1, $w_l = \frac{n_l}{n} w$. Similarly, for $d > 1$, the value of $w_l, \forall \mathbf{g}_l \in \mathcal{G}$, can be computed by solving Equation 8. We propose an efficient algorithm for finding w_l values when $d > 1$ in Section 4.

4 COMPUTING THE COLUMN WIDTHS

Our algorithm is straightforward; we should compute the integer value of w_l such that Equation (8) holds. We note that Equation (8) may not hold for an integer value of w_l. In this case the goal is to compute the integer value of w_l such that the left side of Equation (8) is as close as possible to the right side of Equation (8). A naive implementation is to try every value of w_l and calculate the value of the left and right side of Equation (8) 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_l, and for each such value the calculation of each side of Equation (8) takes $\Omega(n^2)$ time to evaluate the summations.

We first show an exact near-linear time algorithm to compute the value of w_l that solves Equation 8 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 high-frequency elements and the low-frequency elements. Then we extend our method to multiple groups $\mathcal{G} = \{\mathbf{g}_1, \ldots, \mathbf{g}_\ell\}$. Similarly, to the previous section, let $\mathcal{E}(n_l, d, w_l)$ be the function that represents the left side of Equation 8. Equivalently, the right side of Equation (8) is captured by $\mathcal{E}(n_h, d, w - w_l)$. By definition, the function $\mathcal{E}$ computes the expected error of light (resp. heavy) elements. The values n_l, n_h, and d are fixed, so the more bins allocated to light (respectively, heavy) items, the smaller the expected error. Hence, the function $\mathcal{E}(n_l, d, w_l)$ is monotone non-increasing with respect to w_l, while $\mathcal{E}(n_h, d, w - w_l)$ is monotone non-decreasing with respect to w_l. This property leads to the observation that instead of trying all values of w_l, we can run a binary search in the range $[1, \ldots, n]$ and compute the best value of w_l. The overall idea of our algorithm is the following. For every value of w_l in the binary search, we evaluate $\mathcal{E}(n_l, d, w_l)$ and $\mathcal{E}(n_h, d, w - w_l)$. If $\mathcal{E}(n_l, d, w_l) < \mathcal{E}(n_h, d, w - w_l)$ then we try larger values of w_l in the binary search. Otherwise, we continue the binary search with smaller values. In the end, we return the value of w_l found by the binary search such that $|\mathcal{E}(n_l, d, w_l) - \mathcal{E}(n_h, d, w - w_l)|$ 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_l we show how to compute $\mathcal{E}(n_l, d, w_l)$ in near-linear time with respect to n. The computation of $\mathcal{E}(n_h, d, w - w_l)$ is equivalent. First, for $x = 1$ we compute the sum $\sum_{i=1}^{n_l} \binom{n_l}{i} \left(\frac{1}{w_l}\right)^i \left(\frac{w_l - 1}{w_l}\right)^{n_l - i}$ as follows. For $i = 1$, we compute $b_1 = \binom{n_l}{1}$, $t_1 = \frac{1}{w_l}$, and $s_1 = \left(\frac{w_l - 1}{w_l}\right)^{n_l - 1}$. We store $\alpha_1 = b_1 \cdot t_1 \cdot s_1$. Then for $i \in \{2, \ldots, n_l\}$, we observe that $b_i = b_{i-1} \cdot \frac{n_l + 1 - i}{i}$, $t_i = t_{i-1} \cdot \frac{1}{w_l}$, $s_i = s_{i-1} \cdot \frac{w_l}{w_l - 1}$, and we set $\alpha_i = b_i \cdot t_i \cdot s_i$. Then, we set $\beta_{n_l} = \alpha_{n_l}$ and for every $i = n_l - 1, \ldots, 1$, we compute $\beta_i = \alpha_i + \beta_{i+1}$. The main observation of our algorithm is that $\mathcal{E}(n_l, d, w_l) = \sum_{x=1}^{n_l} \beta_x^d$.

In order to analyze the runtime of our algorithm, we assume a modification of the real-RAM model of computation where every basic arithmetic operation is computed in $O(1)$ time. We note that for any pair of positive integer numbers q, k, it holds that $\binom{q}{k} = \prod_{i=1}^{k} \frac{q+1-i}{i}$ so $\binom{q}{k}$ can be computed in $O(k)$ time. Finally, the k-th power of any real number can be computed in $O(\log k)$ time.

We analyze the runtime in one iteration of the binary search. By definition of our model of computation, we compute b_1, t_1 in $O(1)$ time and s_1 in $O(\log n)$ time. Then α_1 is computed in $O(1)$ time. For every value of $i \in \{2, \ldots, n_l\}$, given $b_{i-1}, t_{i-1}, s_{i-1}$ the values of b_i, t_i, s_{i-1} and hence α_i are computed in $O(1)$ time. So all values α_i for $i \in \{1, \ldots, n_l\}$ are computed in $O(n + \log n) = O(n)$ time in total. Next, we observe that given β_{i+1}, the value β_i is computed in $O(1)$ time, so all values β_i are computed in $O(n)$ time. Finally, the sum $\sum_{x=1}^{n_l} \beta_x^d$ is computed in $O\left(\sum_{x=1}^{n} \log(d)\right) = O(n \log d)$ time. The binary search is executed for $O(\log n)$ iterations, so in total the running time of our algorithm is $O(n \log(n) \log(d))$. Notice that usually, $n \gg d$ so the total running time 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 Equation (7). 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 (light and heavy) $\ell - 1$ 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 works in the real-RAM model of computation, a direct calculation of $\binom{n}{i}$ might cause overflow in a practical implementation. In practice, while we run the binary search on w_l, instead of computing the binomial coefficients 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 $\mathcal{E}(n_l, d, w_l)$ and $\mathcal{E}(n_h, d, w - w_l)$ for each value w_l in the binary search.

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 [14]. 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,

$$\mathcal{L}_{CM} = \sum_{i=1}^{n} \varepsilon_A(e_i) \tag{11}$$

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

$$PoF = \mathcal{L}_{FCM} - \mathcal{L}_{CM} \tag{12}$$

In the following, first, we provide our price of fairness analysis for $d = 1$, proving that fairness even improves the expected performance, i.e., $PoF < 0$, when $d = 1$. For $d > 1$, however, fairness comes at a positive (but small) cost.

5.1 $d = 1$

Let us begin by computing $\mathcal{L}_{CM}$, the total additive error for the standard CM sketch. By Chapter 3 of [28], 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

$$\mathcal{L}_{CM} = \sum_{j=1}^{n} \varepsilon_A(e_j) = \sum_{j=1}^{n} \frac{N - f(e_j)}{w} = n\frac{N}{w} - \frac{N}{w} = (n-1)\frac{N}{w} \tag{13}$$

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 (13), for every group $\mathrm{g}_l = \mathcal{G}$,

$$\sum_{e_j \in \mathrm{g}_l} \varepsilon_A(e_j) = (n_l - 1)\frac{N_l}{w_l}$$

Hence,

$$\mathcal{L}_{FCM} = \sum_{e_j \in \mathrm{g}_1} \varepsilon_A(e_j) + \cdots + \sum_{e_j \in \mathrm{g}_\ell} \varepsilon_A(e_j) = \sum_{l=1}^{\ell} (n_l - 1)\frac{N_l}{w_l}$$

From Theorem 1, we know, $w_l = \frac{n_l}{n} w, \forall \mathrm{g}_l \in \mathcal{G}$, while $\sum_{l=1}^{\ell} N_l = N$. Therefore,

$$\begin{aligned}
\mathcal{L}_{FCM} &= \sum_{l=1}^{\ell} (n_l - 1)\frac{N_l}{w_l} \\
&= \sum_{l=1}^{\ell} (n_l - 1)\frac{N_l}{\frac{n_l}{n} w} = \sum_{l=1}^{\ell} n\left(1 - \frac{1}{n_l}\right)\frac{N_l}{w} \\
&= n\sum_{l=1}^{\ell} \frac{N_l}{w} - \sum_{l=1}^{\ell} \frac{n}{n_l} \cdot \frac{N_l}{w} = n\frac{N}{w} - \sum_{l=1}^{\ell} \frac{n}{n_l} \cdot \frac{N_l}{w}
\end{aligned} \tag{14}$$

Interestingly, combining Equations 12, 13, and 14, we show that $PoF < 0$, for a CM with random hashing scheme:

$$\begin{aligned}
PoF &= \mathcal{L}_{FCM} - \mathcal{L}_{CM} \\
&= n\frac{N}{w} - \sum_{l=1}^{\ell} \left(\frac{n}{n_l} \cdot \frac{N_l}{w}\right) - (n-1)\frac{N}{w} \\
&= \frac{N}{w} - \sum_{l=1}^{\ell} \frac{n}{n_l} \cdot \frac{N_l}{w} \\
&< \frac{N}{w} - \sum_{l=1}^{\ell} \frac{N_l}{w} \qquad \text{//since } \frac{n}{n_l} > 1, \forall g_l \in \mathcal{G} \\
&= \frac{N}{w} - \frac{N}{w} = 0
\end{aligned}$$

The negative PoF sounds counterintuitive, but as we shall further elaborate in our appendix , 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 $d > 1$

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

$$\varepsilon_A(e_j) = \mathbb{E}\left[\min_{p \in [d]} \sum_{e \neq e_j : \mathrm{h}_p(e) = \mathrm{h}_p(e_j)} f(e)\right]. \tag{15}$$

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 (15) to compute the sum considering w buckets. For the FCM sketch the total additive error is computed as $\mathcal{L}_{FCM} = \sum_{l=1}^{\ell} \sum_{e_j \in \mathrm{g}_l} \varepsilon_A(e_j)$, where each sum $\sum_{e_j \in \mathrm{g}_l} \varepsilon_A(e_j)$ is computed using Equation (15) for w_l buckets. Since we do not have a closed form for the w_l values (for $d > 1$) from Equation (8), we cannot directly compute $PoF = \mathcal{L}_{CM} - \mathcal{L}_{FCM}$. Instead, we compute the PoF experimentally in real and synthetic datasets in Section 6.6. Particularly, Table 1 and other experiments, including Figures 11 to 16 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 Experiments Setup

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

6.2 Datasets

For evaluation, we used two real-world datasets along with several synthetic ones. To assess the scalability of our proposed sketch in large-scale scenarios, we selected datasets that are sufficiently large to reflect real-world streaming environments.

Census: This dataset contains a one percent sample of the *Public Use Microdata Samples* person records derived from the complete 1990 census dataset. It includes around 2.5 million records with 68 categorical attributes. We use this dataset for experiments involving arbitrarily defined groups. Entity types are defined by concatenating the values of `iYearsch`, `dIndustry`, and the grouping attribute. For binary group experiments, the `iSex` attribute is used to define the groups resulting in 430 element types. To extend the grouping to ℓ groups, we utilize other attributes with higher cardinalities.

Google N-Grams: This dataset is a large-scale collection of n-gram frequency counts extracted from texts in books digitized through the Google Books project. It has been used in related research to evaluate frequency estimation sketches. In our evaluations, we use a sample of 2-grams that begin with *a_* to generate a stream of approximately 1.25 million element types, with a total frequency of around 5 million. We utilize this dataset for experiments in which groups are defined based on the frequency of elements associated with a given element type. If an element's frequency is below a specified threshold, it is assigned to group l; otherwise, it is assigned to group h. This approach is further extended to partition elements into ℓ frequency-based groups.

Synthetic: Finally, we generate several synthetic datasets in which the frequency of element types follows different distributions, including Gaussian, Zipfian, Exponential, and Uniform. We apply a frequency-based grouping to these datasets, similar to the approach used for the GOOGLE N-GRAMS dataset. Each dataset contains $n = 20,000$ element types, with the total frequency determined by the parameters of the underlying distribution. In our experiments on unfairness and the price of fairness, we report results for the case where frequencies are drawn from a Zipfian distribution.

6.3 Baselines

To demonstrate the effectiveness of our approach, we compare it against two baseline methods. The first baseline is the standard **Count-Min** sketch, which we use across all of our experiments. 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.

6.4 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 number of element types in the disadvantaged group n_l, the width of the sketch w, the depth of the sketch d, and the number of groups ℓ. 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 appendix. Finally, we repeat each experiment five times and report the average values for each setting.

6.5 Unfairness

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 disadvantaged and advantaged groups, i.e., $\mathbb{E}[\alpha_l] - \mathbb{E}[\alpha_h]$. In the multi-group setting, unfairness is measured as the difference between the maximum and minimum mean approximation factors across all groups, i.e., $\min_{\forall \mathbf{g}_i \in \mathcal{G}}(\mathbb{E}[\alpha_{\mathbf{g}_i}]) - \max_{\forall \mathbf{g}_j \in \mathcal{G}}(\mathbb{E}[\alpha_{\mathbf{g}_j}])$. A sketch is considered group-fair if the unfairness value is zero.

Varying group size: We begin with the experiment that examines the impact of varying the number of element types n_l 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 l, while those above it are assigned to group h. As the threshold increases, n_l grows while n_h decreases, keeping the total number of elements n constant. Let us now focus on Figure 4. As n_l increases, the standard CM sketch exhibits greater unfairness because the expected approximation factor for group l decreases as the group grows larger, while group h becomes smaller. This is due to the standard CM sketch favoring more accurate estimates for the smaller group h, which contains high-frequency elements, over the larger group l, which consists of numerous low-frequency elements. For the Row-Partitioning baseline, the optimal integral values of d_l and d_h selected by the algorithm gradually deviate from the optimal fractional values determined by the equality 10. Specifically, with $d = 5$, the sketch would ideally assign a value $4 < d_l < 5$ (closer to 5) to group l, which is not feasible. As a result, we observe a significant disparity between the approximation factors of the groups for larger values of n_l. On the other hand, the FCM maintains the approximation factor difference near zero, regardless of the value of n_l. Results conveying a similar message for the SYNTHETIC dataset are also presented in Figure 5.

Varying the number of columns: Next, we examine the impact of varying the sketch width w on unfairness. An instance of this experiment is illustrated in Figure 6. Since $n_l = 1.2M$ is relatively large, the approximation factors for all groups and among all approaches are initially close to zero because the values of w are comparatively small. For the standard CM baseline, as w increases, the approximation factor for group h gradually approaches 1, while it remains relatively low for group l. After surpassing a certain threshold, w becomes large enough for the mean approximation factor of group l to also approach 1. For the Row-Partitioning baseline, as w initially increases, it struggles to find suitable integral d_l values to allocate rows to group l. However, after surpassing a certain threshold, as the sketch becomes wide enough to accommodate fewer elements per bucket, the approximation factor difference begins to decrease. As expected, FCM maintains a zero difference

[2]For this experiment, we used a larger sample of 18 million elements as the stream to ensure that all groups contained a meaningful number of element types. This accounts for the larger additive error values observed in this dataset compared to the other experiments.

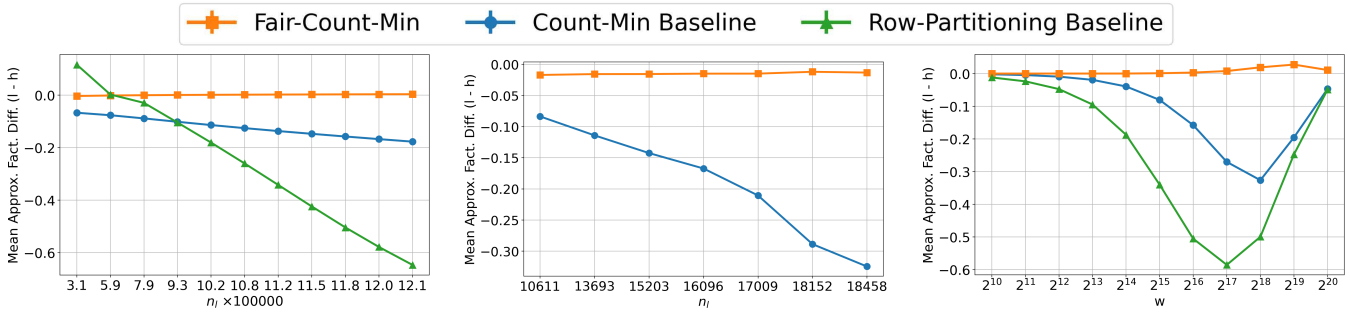

Figure 4: effect of varying disadvantaged group size n_I on unfairness, GOOGLE N-GRAMS, $w = 65536, d = 5$.

Figure 5: effect of varying disadvantaged group size n_I on unfairness, SYNTHETIC, $n = 20K, w = 512, d = 10$.

Figure 6: effect of varying sketch width w on unfairness, GOOGLE N-GRAMS, $n = 1.2M, d = 5$.

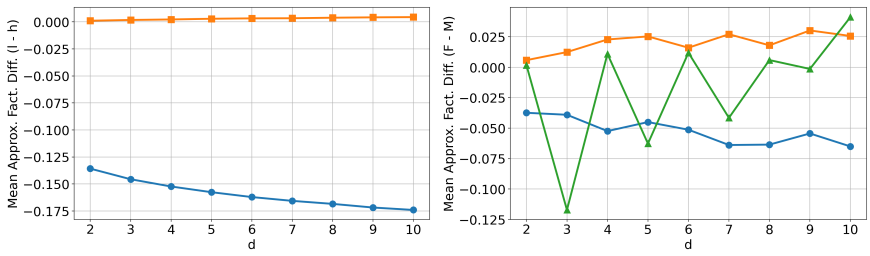

Figure 7: effect of varying sketch depth d on unfairness, GOOGLE N-GRAMS, $n = 1.2M, w = 65536$.

Figure 8: effect of varying sketch depth on unfairness, CENSUS $n = 430, w = 64$.

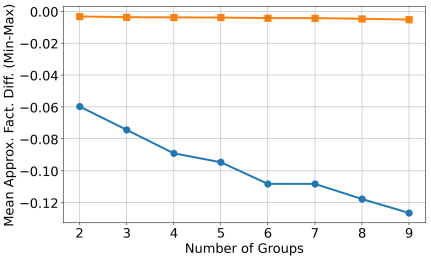

Figure 9: effect of varying number of groups ℓ on unfairness, GOOGLE N-GRAMS, $n = 1.2M, w = 65536, d = 5$.

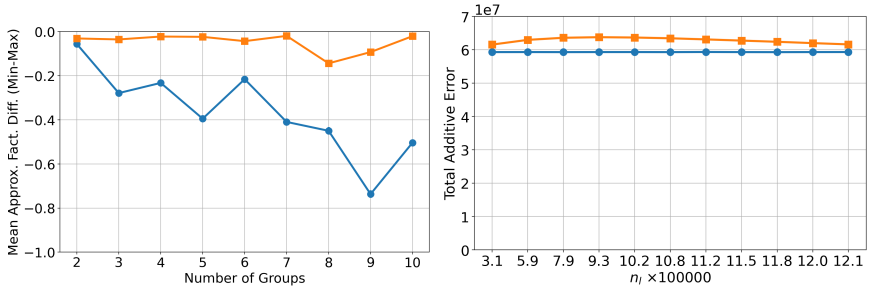

Figure 10: effect of varying number of groups ℓ on unfairness, CENSUS, $n = 430, w = 64, d = 10$.

Figure 11: effect of varying disadvantaged group size n_I on price of fairness, GOOGLE N-GRAMS, $w = 65536, d = 5$.

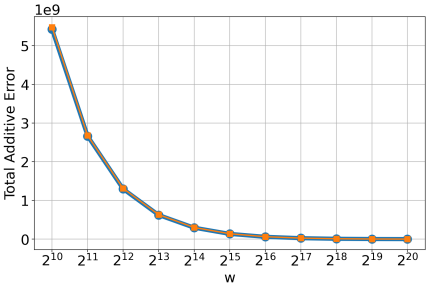

Figure 12: effect of varying sketch width w on price of fairness, GOOGLE N-GRAMS, $n = 1.2M, d = 5$.

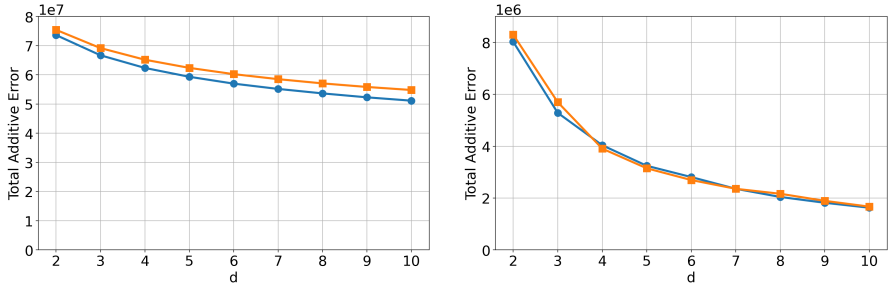

Figure 13: effect of varying sketch depth d on price of fairness, GOOGLE N-GRAMS, $n = 1.2M, w = 65536$.

Figure 14: effect of varying sketch depth d on price of fairness, CENSUS, $n = 430, w = 64$.

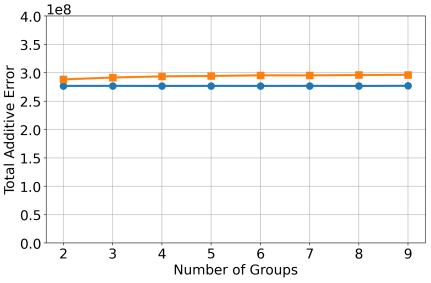

Figure 15: effect of varying number of groups ℓ on price of fairness, GOOGLE N-GRAMS, $n = 1.2M, w = 65536, d = 5^2$.

between the approximation factors of the two groups, regardless of the value of w.

Varying the number of rows: Next, we evaluate the impact of increasing the sketch depth d on unfairness. The results are presented in Figures 7 and 8. For the standard CM baseline, the sketch initially provides more accurate estimates (approximation factors closer to 1) for group h. As d increases, the accuracy improves for both groups, but at a slightly faster rate for group h. As before, FCM maintains zero unfairness regardless of the sketch depth. For the Row-Partitioning approach, we note some interesting observations in As shown in Figure 8, 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_l and n_h are very close, so as long as the sketch can assign equal d_l and d_h 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: In our final experiment on unfairness, we investigate the impact of increasing the number of groups. The results are shown in Figures 9 and 10. 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 standard CM sketch. 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 10, 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.

6.6 Price of Fairness

Thus far, we have established that FCM guarantees group-fairness regardless of the parameters involved. In the next set of experiments, we examine the cost of enforcing this constraint compared to the standard CM. Following the discussion in Section 5 and Equation 12, 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 synthetic dataset where the frequencies of groups l and h are drawn from Gaussian distributions with $\mu_l = 100$, $\sigma_l = 50$ and $\mu_h = 1000$, $\sigma_h = 500$, respectively, and a total of $n = 10{,}000$ element types, we vary the unpopular group size n_l from 9000 to 1000. 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 Table 1, emphasize the superiority of FCM over standard CM in terms of additive error when $d = 1$.

However, when $d > 1$, FCM incurs a positive but small price of fairness, and is generally outperformed by standard CM. We repeated the above experiment with $d = 1$, and the results in Table 1 further support our findings. Still, we notice that the PoF is much smaller in the real datasets compared to the synthetic dataset. That is because synthetic frequency values are generated using a Gaussian distribution, resulting in larger values with lower variance. This is further clear when comparing Table 1 with Figure 11 – the results on the GOOGLE N-GRAMS dataset, where one can confirm the small gap between the total additive error of the CM (the blue line) with FCM (the orange line). To further evaluate the price of fairness for $d > 1$ across various settings, similar to the previous section, we varied (a) group size (Figure 11), (b) number of columns (Figure 12), (c) number of rows (Figures 13 and 14), and (d) the number of groups (Figures 15 and 16), and measured the total additive error for CM versus FCM. The results across all settings were consistent, conveying a unified message: FCM achieves group-fairness at a negligible, if any, price of fairness. Due to the space limitations, extensive results are moved to the appendix. We confirm that we observed similar results across all settings and all other datasets.

6.7 Efficiency

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 the standard CM. For both sketches, the construction time is independent of the disadvantaged group size n_l (Figure 17), the sketch width w (Figure 18), and the number of groups ℓ (Figure 20); 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 Figure 19.

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 and standard CM showing comparable values and trends. The query times are independent of the disadvantaged group size n_l (Figure 21), the sketch width w (Figure 22), and the number of groups ℓ (Figure 24). Similarly, the query time for both sketches also increases linearly with the sketch depth d, as shown in Figure 23.

6.8 Validating Group Columns Allocation

In our final experiment, we evaluate experimentally validate our mathematical analysis in Section 3.2, using a Monte Carlo method

Total Additive Error			$n_l\,(n = 10{,}000)$								
			9000	**8000**	**7000**	**6000**	**5000**	**4000**	**3000**	**2000**	**1000**
$d = 1$	theoretical	**CM**	19,047,019	28,054,816	37,081,001	46,092,289	55,121,050	64,147,002	73,186,882	82,323,243	91,265,426
		FCM	18,985,224	28,028,987	37,003,399	46,050,674	55,115,538	64,089,586	73,165,549	82,249,909	91,228,281
	empirical	**CM**	19,039,343	28,085,806	37,053,427	46,096,489	55,113,194	64,234,176	73,227,052	82,283,374	91,328,074
		FCM	18,991,509	27,982,573	37,002,246	46,071,394	55,086,890	64,133,815	73,168,189	82,230,649	91,276,271
		PoF	↓-47,834	↓-103,233	↓-51,181	↓-25,095	↓-26,304	↓-100,361	↓-58,863	↓-52,725	↓-51,803
$d = 5$	empirical	**CM**	7,964,348	11,572,548	16,458,026	22,023,181	28,305,699	34,442,308	40,959,703	47,230,290	54,257,770
		FCM	11,695,556	17,114,933	22,666,115	28,053,739	33,856,154	39,448,942	44,913,211	50,089,698	55,893,637
		PoF	↑3,731,208	↑5,542,385	↑6,208,089	↑6,030,558	↑5,550,455	↑5,006,634	↑3,953,508	↑2,859,408	↑1,635,867

Table 1: comparison of CM and FCM w.r.t additive errors and the price of fairness across different n_l values for $d = 1$ and $d = 5$.

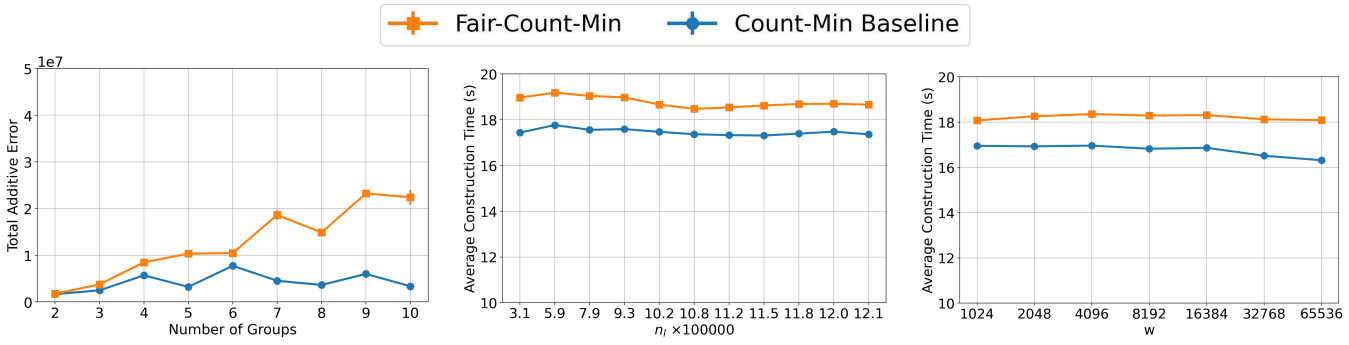

Figure 16: effect of varying number of groups ℓ on price of fairness, CENSUS, $n = 430$, $w = 64$, $d = 10$.

Figure 17: effect of varying disadvantaged group size n_l on construction time, GOOGLE N-GRAMS, $n = 1.2M$, $w = 65536$.

Figure 18: effect of varying sketch width w on construction time, GOOGLE N-GRAMS, $n = 1.2M$, $d = 5$.

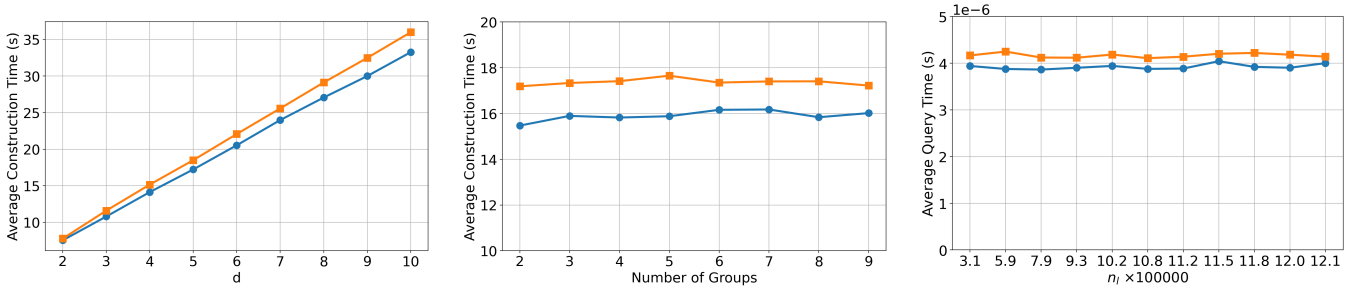

Figure 19: effect of varying sketch depth d on construction time, GOOGLE N-GRAMS, $n = 1.2M$, $w = 65536$.

Figure 20: effect of varying number of groups ℓ on construction time, GOOGLE N-GRAMS, $n = 1.2M$, $w = 65536$, $d = 5$.

Figure 21: effect of varying disadvantaged group size n_l on query time, GOOGLE N-GRAMS, $n = 1.2M$, $w = 65536$.

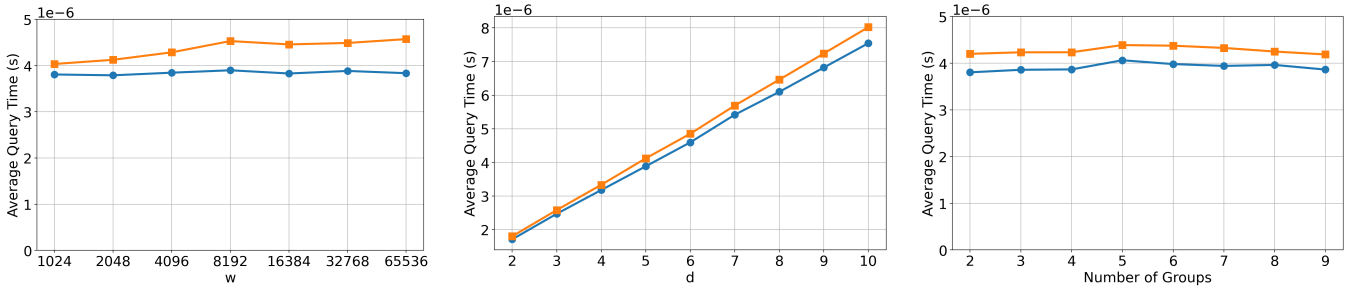

Figure 22: effect of varying sketch width w on query time, GOOGLE N-GRAMS, $n = 1.2M$, $d = 5$.

Figure 23: effect of varying sketch depth d on query time, GOOGLE N-GRAMS, $n = 1.2M$, $w = 65536$.

Figure 24: effect of varying number of groups ℓ on query time, GOOGLE N-GRAMS, $n = 1.2M$, $w = 65536$, $d = 5$.

for estimating the value of w_l – the number of columns allocated to the group l.

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_l:

Consider n_l, n_h, w, d, and the parameters of arbitrary distributions from which the frequencies of groups l and h are drawn. For each group $\mathbf{g} \in \{l, h\}$, we draw $n_\mathbf{g}$ samples from the corresponding 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_l 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 for each sketch row), and we compute the average of the resulting w_l values. Finally, we repeat the entire process over multiple iterations and report the overall average of w_l.

We use this Monte Carlo method to estimate w_l under a range of synthetic distributions. We note from our analysis that the value of w_l 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 25. The values of w_l were identical across different frequency distributions and, as expected, depended solely on the values of n_l and n_h.

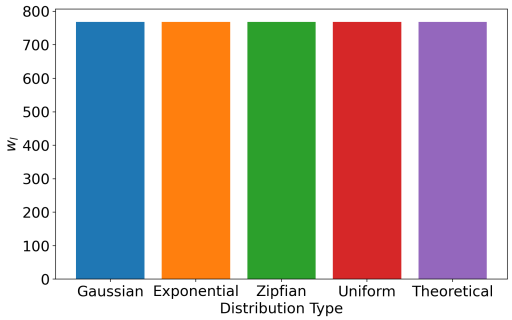

Figure 25: Monte Carlo-based approach to finding w_l vs. theoretical value. w_l only depends on n_l and is independent of the distribution of frequencies of each group.

7 RELATED WORK

Count-Min sketch. The CM sketch was first described by Cormode and Muthukrishnan [26, 27]. 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 [21] and AMS sketch [4] and can be considered an implementation of a counting Bloom filter [36, 41] or multistage-filter [26]. Over the years multiple variations of min-count sketches have been proposed in the literature [39, 54, 62, 64, 76] 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, 18, 31, 32, 46, 59, 80] 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 [46, 59] 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 [49] or bloom filters [49, 56, 77].

Fairness. Fairness in data-driven systems has been studied by various research communities but mostly in machine learning (ML) [13, 55, 61]. Most of the existing work is on training a ML model that satisfies some fairness constraints. Some pioneering fair-ML efforts include [19, 20, 35, 37, 42, 48, 78]. Biases in data has also been studied extensively [8, 9, 60, 65, 69, 74] to ensure data has been prepared responsibly [58, 66, 67, 71]. Recent studies of fair algorithm design include fair clustering [2, 15, 17, 23, 24, 45, 52, 68], fairness in resource allocation and facility location problem [16, 33, 40, 43, 47, 82], min cut [50], max cover [6], set cover [29], game theoretic approaches [3, 34, 81], hiring [5, 63], ranking [7, 72, 73, 79], recommendation [22, 51, 75], representation learning [44], etc.

Fairness in data structures is significantly under-studied, with the existing work being limited to [10–12, 70]. Aumuller et al. [10–12] study individual fairness in near-neighbor search, while to the best of our knowledge, Shahbazi et al. [70] is the only work that studies group fairness in data structure design, and more specifically in hashing. Both in nearest-neighbor search and fair hashing, the objective is fundamentally different, so these techniques cannot be used for our fair-CM sketch.

8 CONCLUSION

In this paper we introduced Fair-Count-Min, a new frequency estimation framework that ensures equal expected approximation factors across different groups, addressing a key fairness limitation of traditional Count-Min 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 Fair-Count-Min 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] 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.

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

[4] 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.

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

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

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

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

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

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

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

[18] 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).

[19] 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).

[20] 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.

[21] 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.

[22] 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.

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

[24] 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.

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

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

[27] 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.

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

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

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

[31] 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.

[32] 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.

[33] 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.

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

[35] 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.

[36] 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.

[37] 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.

[38] William Feller. 1991. *An introduction to probability theory and its applications, Volume 2*. Vol. 2. John Wiley & Sons.

[39] 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.

[40] 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.

[41] 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.

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

[43] 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).

[44] 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.

[45] Sedjro Salomon Hotegni, Sepideh Mahabadi, and Ali Vakilian. 2023. Approximation Algorithms for Fair Range Clustering. In *International Conference on Machine Learning*. PMLR, 13270–13284.

[46] Chen-Yu Hsu, Piotr Indyk, Dina Katabi, and Ali Vakilian. 2019. Learning-based frequency estimation algorithms.. In *International Conference on Learning Representations*.

[47] 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.

[48] Faisal Kamiran and Toon Calders. 2012. Data preprocessing techniques for classification without discrimination. *Knowledge and Information Systems* 33, 1 (2012), 1–33.

[49] 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.

[50] 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.

[51] 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).

[52] Yury Makarychev and Ali Vakilian. 2021. Approximation algorithms for socially fair clustering. In *Conference on Learning Theory*. PMLR, 3246–3264.

[53] Younes Ben Mazziane, Sara Alouf, and Giovanni Neglia. 2022. Analyzing count min sketch with conservative updates. *Computer Networks* 217 (2022), 109315.

[54] Younes Ben Mazziane and Othmane Marfoq. 2024. Count-Min Sketch with Conservative Updates: Worst-Case Analysis. *arXiv preprint arXiv:2405.12034* (2024).

[55] 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.

[56] Michael Mitzenmacher. 2018. A model for learned bloom filters and optimizing by sandwiching. *Advances in Neural Information Processing Systems* 31 (2018).
[57] Rajeev Motwani and Prabhakar Raghavan. 1996. Randomized algorithms. *ACM Computing Surveys (CSUR)* 28, 1 (1996), 33–37.
[58] 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.
[59] Thuy Trang Nguyen and Cameron N Musco. [n. d.]. Partitioned-Learned Count-Min Sketch. ([n. d.]).
[60] 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.
[61] Dana Pessach and Erez Shmueli. 2022. A review on fairness in machine learning. *ACM Computing Surveys (CSUR)* 55, 3 (2022), 1–44.
[62] Guillaume Pitel and Geoffroy Fouquier. 2015. Count-Min-Log sketch: Approximately counting with approximate counters. In *International Symposium on Web AlGorithms*.
[63] 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.
[64] 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.
[65] Babak Salimi, Bill Howe, and Dan Suciu. 2019. Data management for causal algorithmic fairness. *arXiv preprint arXiv:1908.07924* (2019).
[66] Babak Salimi, Bill Howe, and Dan Suciu. 2020. Database repair meets algorithmic fairness. *ACM SIGMOD Record* 49, 1 (2020), 34–41.
[67] 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.
[68] 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.
[69] Nima Shahbazi, Yin Lin, Abolfazl Asudeh, and HV Jagadish. 2023. Representation Bias in Data: A Survey on Identification and Resolution Techniques. *Comput. Surveys* (2023).
[70] 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.
[71] Suraj Shetiya, Ian P Swift, Abolfazl Asudeh, and Gautam Das. 2022. Fairness-aware range queries for selecting unbiased data. In *ICDE*. IEEE, 1423–1436.
[72] 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.
[73] Ashudeep Singh and Thorsten Joachims. 2019. Policy learning for fairness in ranking. *Advances in neural information processing systems* 32 (2019).
[74] Julia Stoyanovich, Serge Abiteboul, Bill Howe, HV Jagadish, and Sebastian Schelter. 2022. Responsible data management. *Commun. ACM* 65, 6 (2022), 64–74.
[75] 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.
[76] 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.
[77] Kapil Vaidya, Eric Knorr, Michael Mitzenmacher, and Tim Kraska. 2020. Partitioned Learned Bloom Filters. In *International Conference on Learning Representations*.
[78] 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.
[79] Meike Zehlike, Ke Yang, and Julia Stoyanovich. 2022. Fairness in ranking, part i: Score-based ranking. *Comput. Surveys* 55, 6 (2022), 1–36.
[80] Meifan Zhang, Hongzhi Wang, Jianzhong Li, and Hong Gao. 2020. Learned sketches for frequency estimation. *Information Sciences* 507 (2020), 365–385.
[81] 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.
[82] 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.

APPENDIX

A PRICE OF FAIRNESS: RANDOMNESS VS. UNIFORMITY

In Section 5, we observed a counterintuitive result for $d = 1$, proving a negative price of fairness (PoF) for FCM, compared to a regular CM sketch that uses a random hash function. This result can be explained by the fact that a random hash function is unlikely to *uniformly* distribute the elements to the buckets (an interesting related topic is *the occupant problem* [57]). As a result, some of the buckets will have more than $\frac{n}{w}$ elements in them. For example, Figure 26 illustrates a random hashing of $n = 100$ element types into $w = 10$ buckets. While a uniform distribution would allocate 10 elements to each bucket, the random hashing allocated up to 18 elements in one bucket.

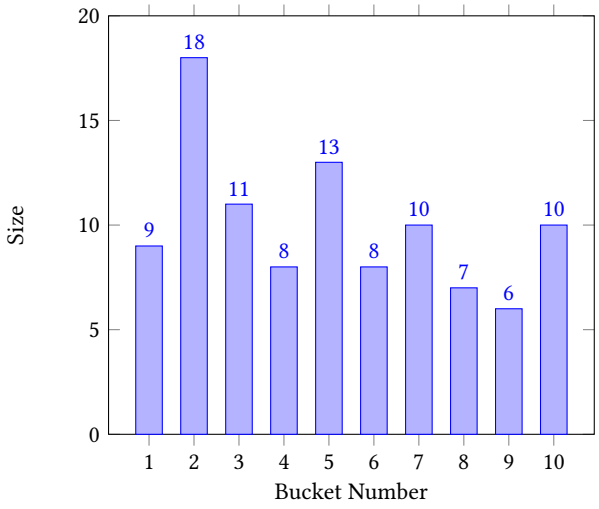

Figure 26: Illustration of a random distribution of $n = 100$ elements to $w = 10$ buckets.

Elements in large buckets will suffer from a large additive error, and there are a large number of them in such buckets. Hence, such buckets significantly increase the total additive error of the CM.

FCM increases the size-uniformity of the buckets by reserving $w_l = \frac{n_l}{n} w$ for each group g_l. For example, Figure 27 illustrates the allocation of the 100 elements to the 10 buckets, by dividing (arbitrarily) the elements into two groups of size $n_l = 50$, $n_h = 50$. As a result, each group is allocated 5 buckets, and the distribution of the elements is more uniform, reducing the maximum size of the buckets (in this example) to 14. Increasing the uniformity (reducing the change of large-size buckets) helps to reduce the total additive error of FCM versus CM.

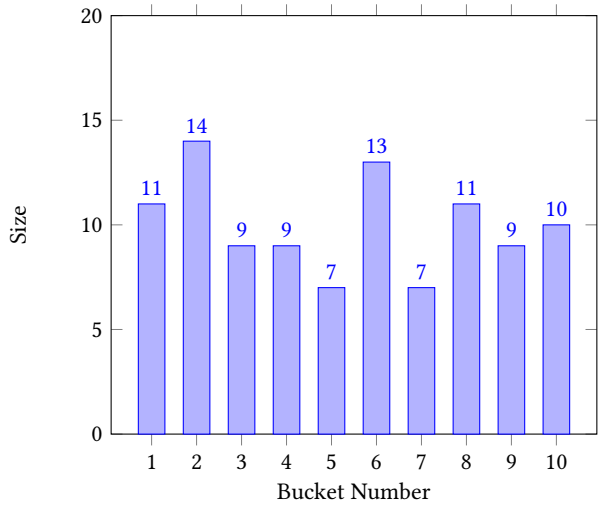

Figure 27: Illustration of a random distribution of $n = 100$ elements with two groups of sizes $n_l = 50$ and $n_h = 50$ to $w = 10$ buckets, based on FCM.

A.1 PoF for Uniform hashing ($d = 1$)

Using a hashing scheme (such as data-informed hashmaps [49]) that ensures uniformity by allocating exactly $\frac{n}{w}$ elements to each bucket can resolve the non-uniformity issue in the CM sketches. In the following, we compute the PoF of SCM for $d = 1$ when a uniform hashing scheme is used.

Since the hashing is uniform, the size of each bucket i in the CM sketch is

$$C_i = \frac{n}{w}$$

Therefore, each element e_j collides with exactly $\frac{n}{w} - 1$ other elements. In other words, the frequency of each element contributes to the additive error of exactly $\frac{n}{w} - 1$ other elements.

As a result, the total additive error can be computed as

$$\begin{aligned}\mathcal{L}_{CM} &= \sum_{j=1}^{n} \varepsilon_A(e_j) \\ &= \sum_{j=1}^{n} f(e_j)\left(\frac{n}{w} - 1\right) = \left(\frac{n}{w} - 1\right)\sum_{j=1}^{n} f(e_j) \\ &= N\left(\frac{n}{w} - 1\right) \end{aligned} \tag{16}$$

Now, let us compute $\mathcal{L}_{FCM}$, the total additive error for the regular fair-count-min 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 16, for every group $\mathrm{g}_l = \mathcal{G}$,

$$\sum_{e_j \in \mathrm{g}_l} \varepsilon_A(e_j) = N_l\left(\frac{n_l}{w_l} - 1\right)$$

Hence,

$$\mathcal{L}_{FCM} = \sum_{e_j \in \mathrm{g}_1} \varepsilon_A(e_j) + \cdots + \sum_{e_j \in \mathrm{g}_\ell} \varepsilon_A(e_j) = \sum_{l=1}^{\ell} N_l\left(\frac{n_l}{w_l} - 1\right)$$

From Theorem 1, we know, $w_l = \frac{n_l}{n} w, \forall \mathrm{g}_l \in \mathcal{G}$, while $\sum_{l=1}^{\ell} N_l = N$. Therefore,

$$\begin{aligned}\mathcal{L}_{FCM} &= \sum_{l=1}^{\ell} N_l\left(\frac{n_l}{w_l} - 1\right) \\ &= \sum_{l=1}^{\ell} N_l\left(\frac{n_l}{w_l}\right) - \sum_{l=1}^{\ell} N_l = \sum_{l=1}^{\ell} N_l\left(\frac{n_l}{w_l}\right) - N \\ &= \left(\frac{n}{w}\right)\sum_{l=1}^{\ell} N_l - N \qquad \texttt{//since } \frac{n_l}{w_l} = \frac{n}{w}, \forall \mathrm{g}_l \in \mathcal{G} \\ &= N\left(\frac{n}{w}\right) - N = N\left(\frac{n}{w} - 1\right) \end{aligned} \tag{17}$$

Finally, combining Equations 12, 16, and 17, we get,

$$PoF = \mathcal{L}_{FCM} - \mathcal{L}_{CM} = N\left(\frac{n}{w} - 1\right) - N\left(\frac{n}{w} - 1\right) = 0 \tag{18}$$

That is, the PoF is zero for $d = 1$, when a uniform hashing scheme is used.

B ADDITIONAL EXPERIMENT RESULTS

In this section, we present comprehensive experimental results across three datasets: 1) GOOGLE N-GRAMS, 2) CENSUS, and 3) SYNTHETIC. We also provide the absolute values of the approximation factors and additive errors for each group to facilitate a clearer understanding of the trends in unfairness and the associated price of fairness under each setting.

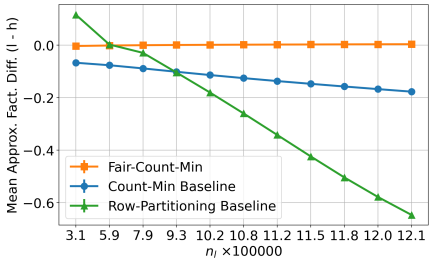

Figure 28: effect of varying group size n_l on unfairness, GOOGLE N-GRAMS, $w = 65536, d = 5$.

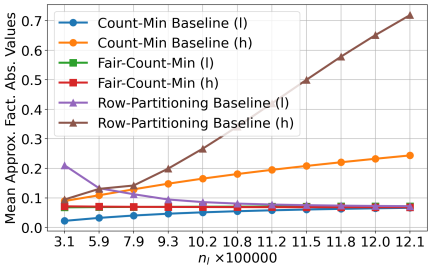

Figure 29: effect of varying group size n_l on approximation factors, GOOGLE N-GRAMS, $w = 65536, d = 5$.

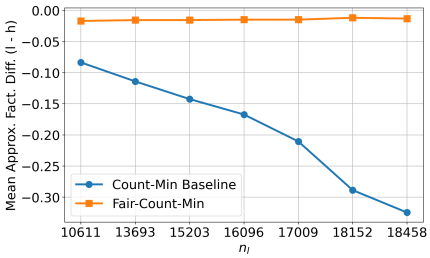

Figure 30: effect of varying group size n_l on unfairness, SYNTHETIC, $w = 512, d = 10$.

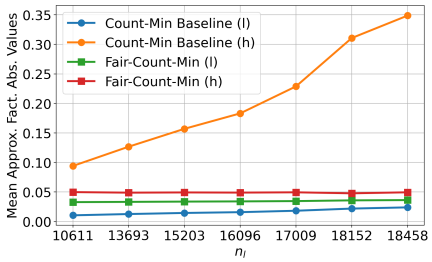

Figure 31: effect of varying group size n_l on approximation factors, SYNTHETIC, $w = 512, d = 10$.

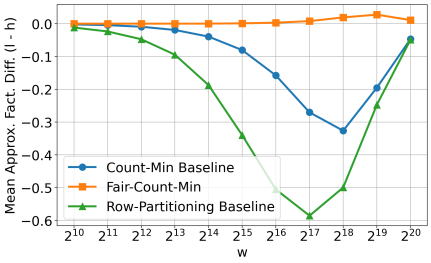

Figure 32: effect of varying sketch width w on unfairness, GOOGLE N-GRAMS, $n = 1.2M, d = 5$.

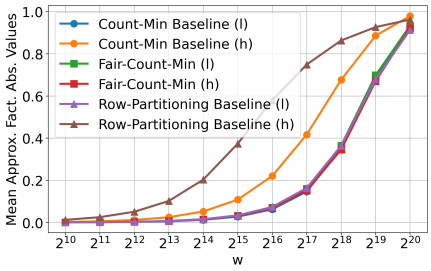

Figure 33: effect of varying sketch width w on approximation factors, GOOGLE N-GRAMS, $n = 1.2M, d = 5$.

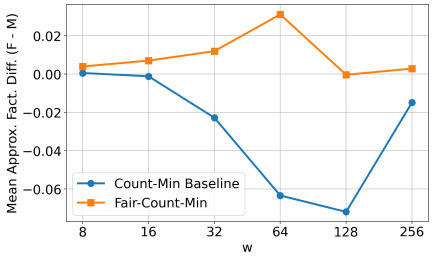

Figure 34: effect of varying sketch width w on unfairness, CENSUS, $n = 430, d = 5$.

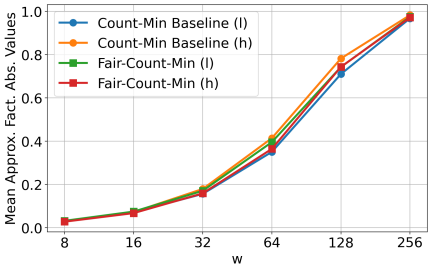

Figure 35: effect of varying sketch width w on approximation factors, CENSUS, $n = 430, d = 5$.

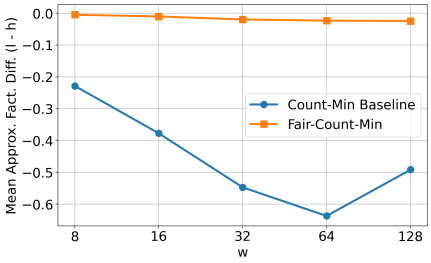

Figure 36: effect of varying sketch width w on unfairness, SYNTHETIC, $n = 20000, d = 10$.

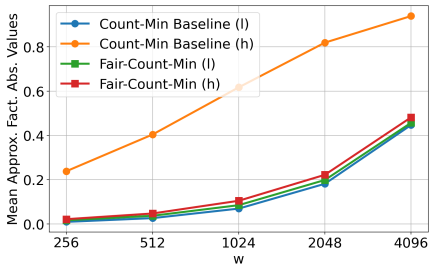

Figure 37: effect of varying sketch width w on approximation factors, SYNTHETIC, $n = 20000, d = 10$.

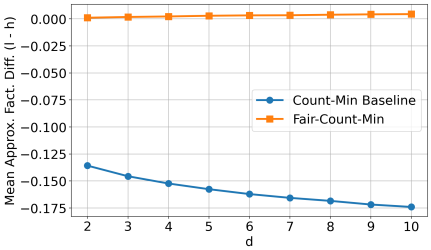

Figure 38: effect of varying sketch depth d on unfairness, GOOGLE N-GRAMS, $n = 1.2M, w = 65536$.

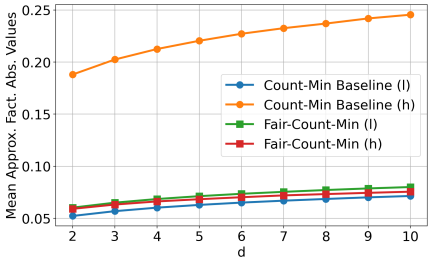

Figure 39: effect of varying sketch depth d on approximation factors, GOOGLE N-GRAMS, $n = 1.2M, w = 65536$.

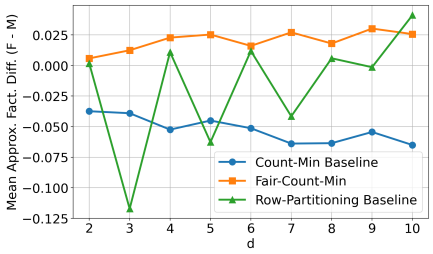

Figure 40: effect of varying sketch depth d on unfairness, CENSUS, $n = 430$, $w = 64$.

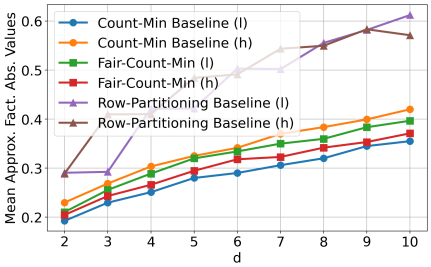

Figure 41: effect of varying sketch depth d on approximation factors, CENSUS, $n = 430$, $w = 64$.

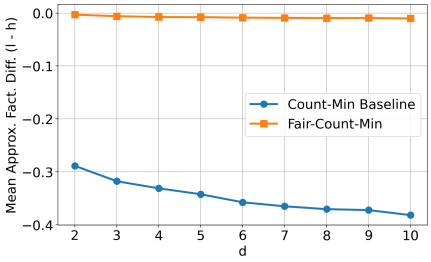

Figure 42: effect of varying sketch depth d on unfairness, SYNTHETIC, $n = 20000$, $w = 512$.

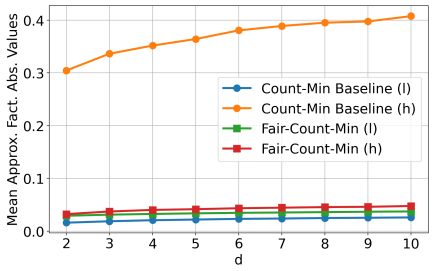

Figure 43: effect of varying sketch depth d on approximation factors, SYNTHETIC, $n = 20000$, $w = 512$.

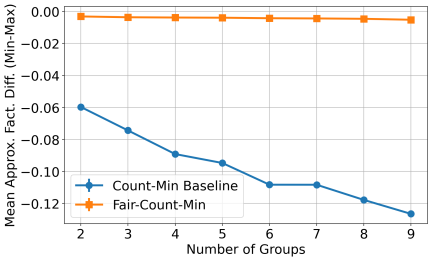

Figure 44: effect of varying number of groups ℓ on unfairness, GOOGLE N-GRAMS, $n = 1.2M$, $W = 65536$, $d = 5$.

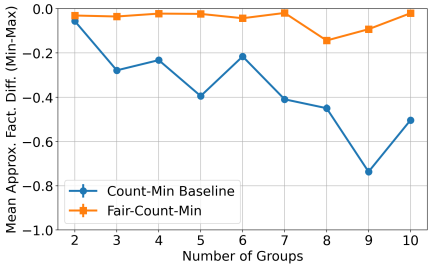

Figure 45: effect of varying number of groups ℓ on unfairness, CENSUS, $n = 400$, $w = 64$, $d = 5$.

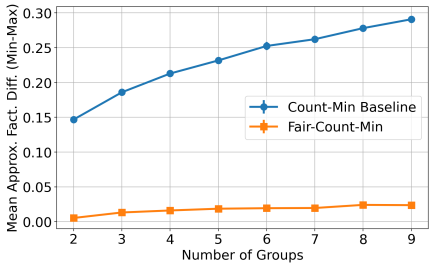

Figure 46: effect of varying number of groups ℓ on unfairness, SYNTHETIC, $n = 20000$, $w = 512$, $d = 10$.

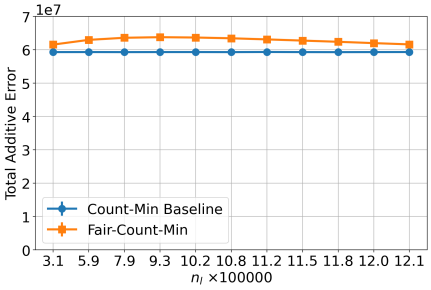

Figure 47: effect of varying group size n_l on price of fairness, GOOGLE N-GRAMS, $w = 65536$, $d = 5$.

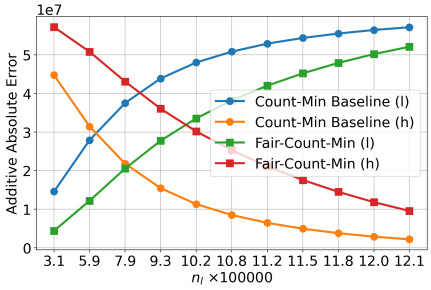

Figure 48: effect of varying group size n_l on absolute additive errors, GOOGLE N-GRAMS, $w = 65536$, $d = 5$.

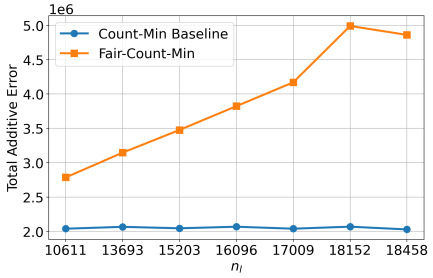

Figure 49: effect of varying group size n_l on price of fairness, SYNTHETIC, $w = 512$, $d = 10$.

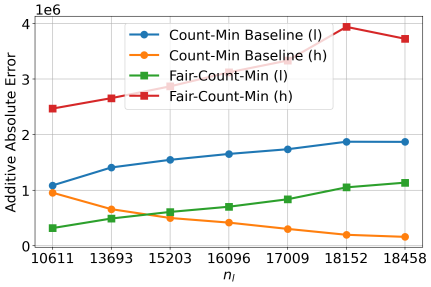

Figure 50: effect of varying group size n_l on absolute additive errors, SYNTHETIC, $w = 512$, $d = 10$.

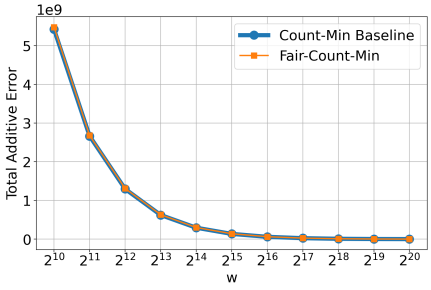

Figure 51: effect of varying sketch width w on price of fairness, GOOGLE N-GRAMS, $n = 1.2M$, $d = 5$.

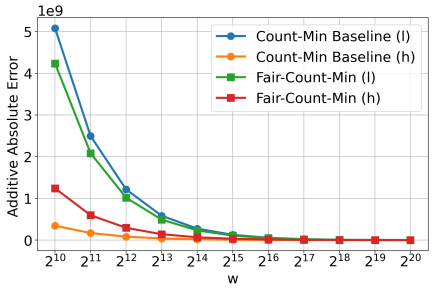

Figure 52: effect of varying sketch width w on absolute additive errors, GOOGLE N-GRAMS, $n = 1.2M$, $d = 5$.

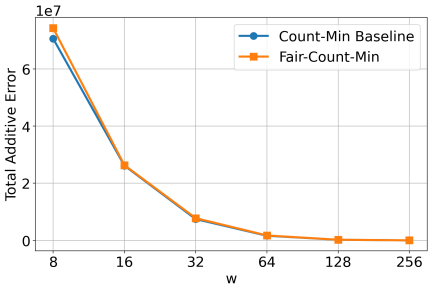

Figure 53: effect of varying sketch width w on price of fairness, CENSUS, $n = 430$, $d = 5$.

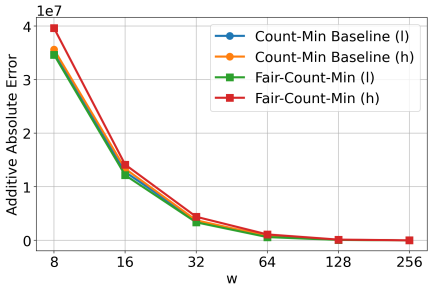

Figure 54: effect of varying sketch width w on absolute additive errors, CENSUS, $n = 430$, $d = 5$.

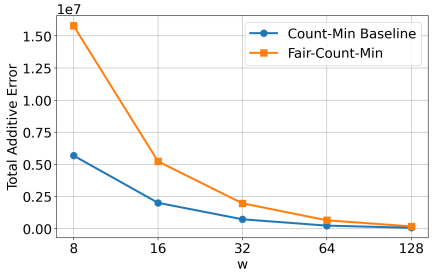

Figure 55: effect of varying sketch width w on price of fairness, SYNTHETIC, $n = 20000$, $d = 10$.

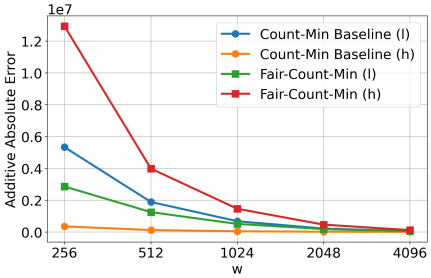

Figure 56: effect of varying sketch width w on absolute additive errors, SYNTHETIC, $n = 20000$, $d = 10$.

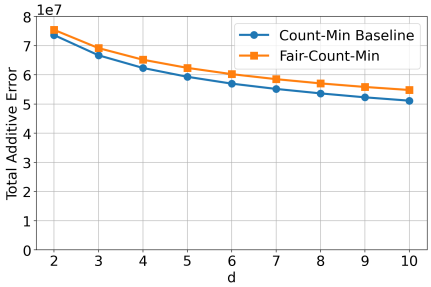

Figure 57: effect of varying sketch depth d on price of fairness, GOOGLE N-GRAMS, $n = 1.2M$, $w = 65536$.

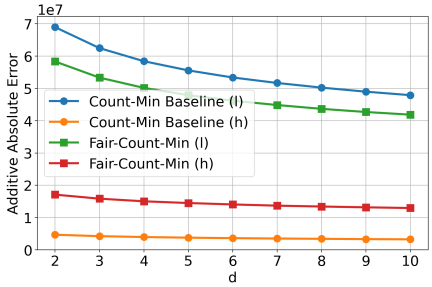

Figure 58: effect of varying sketch depth d on absolute additive errors, GOOGLE N-GRAMS, $n = 1.2M$, $w = 65536$.

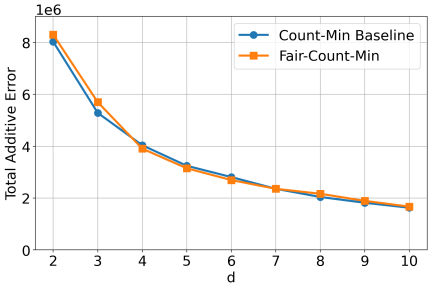

Figure 59: effect of varying sketch depth d on price of fairness, CENSUS, $n = 430$, $w = 64$.

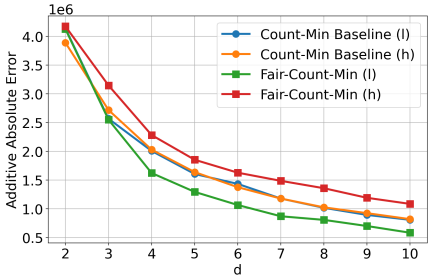

Figure 60: effect of varying sketch depth d on absolute additive errors, CENSUS, $n = 430$, $w = 64$.

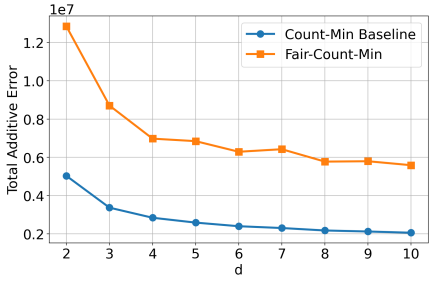

Figure 61: effect of varying sketch depth d on price of fairness, SYNTHETIC, $n = 20000$, $w = 512$.

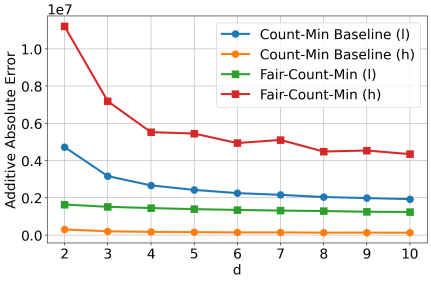

Figure 62: effect of varying sketch depth d on absolute additive errors, SYNTHETIC, $n = 20000$, $w = 512$.

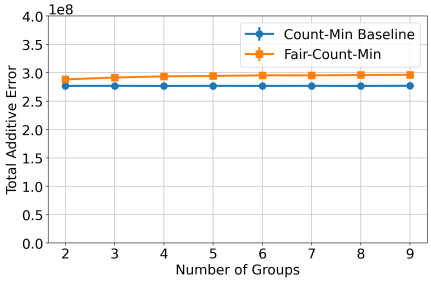

Figure 63: effect of varying number of groups ℓ on price of fairness, GOOGLE N-GRAMS, $n = 1.2M$, $W = 65536$, $d = 5$.

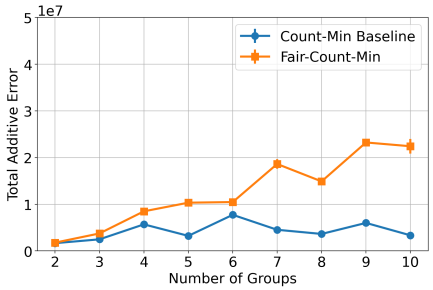

Figure 64: effect of varying number of groups ℓ on price of fairness, CENSUS, $n = 400$, $w = 64$, $d = 5$.

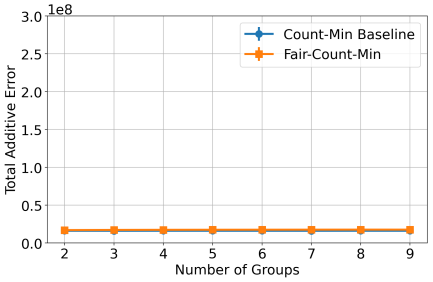

Figure 65: effect of varying number of groups ℓ on price of fairness, SYNTHETIC, $n = 20000$, $w = 512$, $d = 10$.

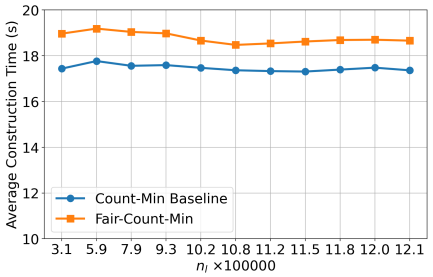

Figure 66: effect of varying group size n_l on construction time, GOOGLE N-GRAMS, $w = 65536$, $d = 5$.

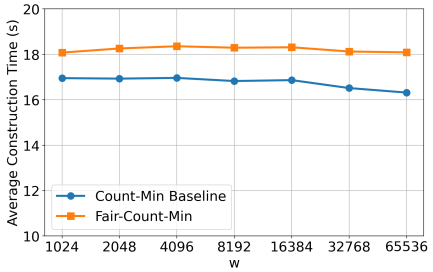

Figure 67: effect of varying sketch width w on construction time, GOOGLE N-GRAMS, $n = 1.2M$, $d = 5$.

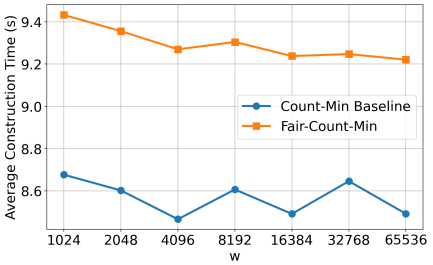

Figure 68: effect of varying sketch width w on construction time, CENSUS, $n = 430$, $d = 5$.

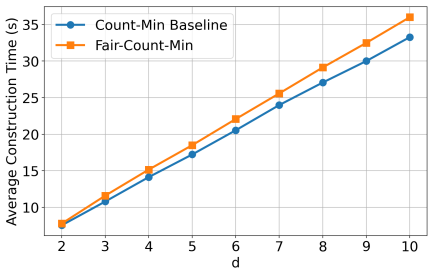

Figure 69: effect of varying sketch depth d on construction time, GOOGLE N-GRAMS, $n = 1.2M$, $w = 65536$.

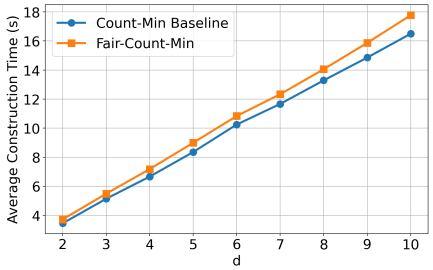

Figure 70: effect of varying sketch depth d on construction time, CENSUS, $n = 430$, $w = 64$.

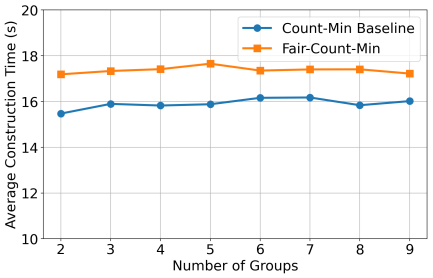

Figure 71: effect of varying number of groups ℓ on construction time, GOOGLE N-GRAMS, $n = 1.2M$, $w = 65536$, $d = 5$.

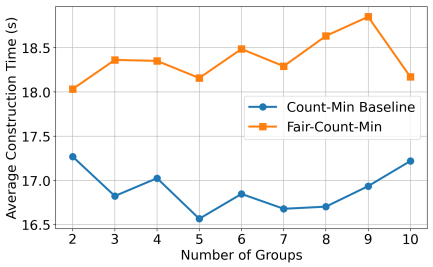

Figure 72: effect of varying number of groups ℓ on construction time, CENSUS, $n = 430$, $w = 64$, $d = 5$.

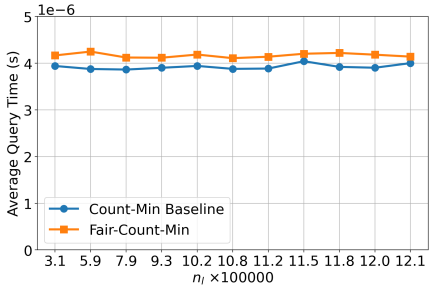

Figure 73: effect of varying group size n_l on query time, GOOGLE N-GRAMS, $w = 65536$, $d = 5$.

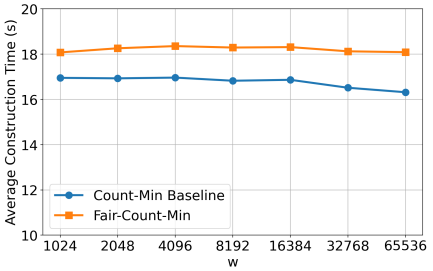

Figure 74: effect of varying sketch width w on query time, GOOGLE N-GRAMS, $n = 1.2M$, $d = 5$.

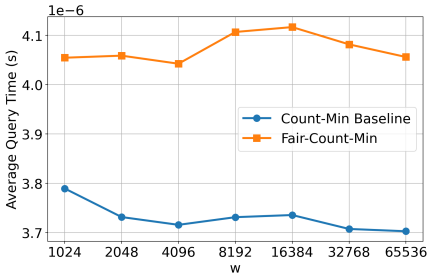

Figure 75: effect of varying sketch width w on query time, CENSUS, $n = 430$, $d = 5$.

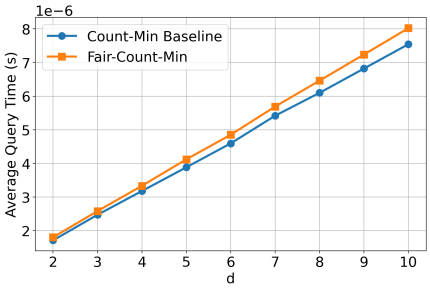

Figure 76: effect of varying sketch depth d on query time, GOOGLE N-GRAMS, $n = 1.2M$, $w = 65536$.

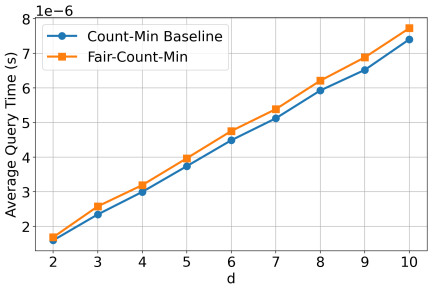

Figure 77: effect of varying sketch depth d on query time, CENSUS, $n = 430$, $w = 64$.

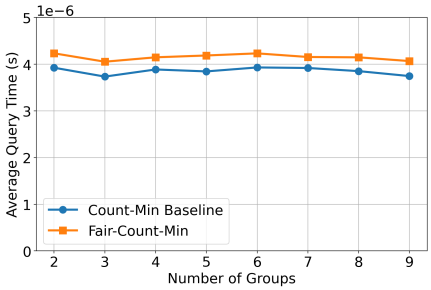

Figure 78: effect of varying number of groups ℓ on query time, GOOGLE N-GRAMS, $n = 1.2M$, $w = 65536$, $d = 5$.

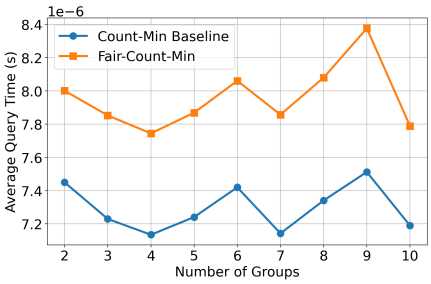

Figure 79: effect of varying number of groups ℓ on query time, CENSUS, $n = 430$, $w = 64$, $d = 5$.