Count-Min Sketch with Variable Number of Hash Functions: An Experimental Study

Éric Fusy and Gregory Kucherov(✉)

LIGM, CNRS, Univ. Gustave Eiffel, Marne-la-Vallée, France
{Eric.Fusy,Gregory.Kucherov}@univ-eiffel.fr

Abstract. Conservative Count-Min, a stronger version of the popular Count-Min sketch [Cormode, Muthukrishnan 2005], is an online-maintained hashing-based sketch summarizing element frequency information of a stream. Although several works attempted to analyze the error of conservative Count-Min, its behavior remains poorly understood. In [Fusy, Kucherov 2022], we demonstrated that under the uniform distribution of input elements, the error of conservative Count-Min follows two distinct regimes depending on its load factor.

In this work, we present a series of results providing new insights into the behavior of conservative Count-Min. Our contribution is twofold. On one hand, we provide a detailed experimental analysis of Count-Min sketch in different regimes and under several representative probability distributions of input elements. On the other hand, we demonstrate improvements that can be made by assigning a variable number of hash functions to different elements. This includes, in particular, reduced space of the data structure while still supporting a small error.

1 Introduction

In most general terms, *Count-Min sketch* is a data structure for representing an associative array of numbers indexed by elements (keys) drawn from a large universe, where the array is provided through a stream of (key, value) updates so that the current value associated to a key is the sum of all previous updates of this key. Perhaps the most common setting for applying Count-Min, that we focus on in this paper, is the *counting* setting where all update values are +1. In this case, the value of a key is its *count* telling how many times this key has appeared in the stream. In other words, Count-Min can be seen as representing a *multiset*, that is a mapping of a subset of keys to non-negative integers. With this latter interpretation in mind, each update will be called *insertion*. The main supported query of Count-Min is retrieving the count of a given key, and the returned estimate may not be exact, but can only overestimate the true count.

The counting version of Count-Min is applied to different practical problems related to data stream mining and data summarization. One example is tracking frequent items (*heavy hitters*) in streams [7,11,23]. It occurs in network traffic monitoring [17], optimization of cache usage [16]. It also occurs in non-streaming big data applications, e.g. in bioinformatics [1,25,29].

F. M. Nardini et al. (Eds.): SPIRE 2023, LNCS 14240, pp. 218–232, 2023.
https://doi.org/10.1007/978-3-031-43980-3_17

Count-Min relies on hash functions but, unlike classic hash tables, does not store elements but only count information (hence the term *sketch*). It was proposed in [12], however a very similar data structure was proposed earlier in [9] under the name *Spectral Bloom filter*. The latter, in turn, is closely related to *Counting Bloom filters* [19]. In this work, we adopt the definition of [9] but still call it Count-Min to be consistent with the name commonly adopted in the literature. A survey on Count-Min can be found e.g. in [10].

In this paper, we study a stronger version of Count-Min called *conservative*. This modification of Count-Min was introduced in [17] under the name *conservative update*, see [10]. It was also discussed in [9] under the name *minimal increase*. Conservative Count-Min provides strictly tighter count estimates using the same memory and thus strictly outperforms the original version. The price to pay is the impossibility to deal with deletions (negative updates), whereas the original Count-Min can handle deletions as well, provided that the cumulative counts remain non-negative (condition known as *strict turnstile model* [23]).

Analysis of error of conservative Count-Min is a difficult problem having direct consequences on practical applications. Below in Sect. 2.2 we survey known related results in more details. In our previous work [21], we approached this problem through the relationship with *random hypergraphs*. We proved, in particular, that if the elements represented in the data structure are uniformly distributed in the input, the error follows two different regimes depending on the *peelability* property of the underlying *hash hypergraph*. While properties of random hypergraphs have been known to be crucially related to some data structures (see Sect. 2.3), this had not been known for Count-Min.

Starting out from these results, in this paper we extend and strengthen this analysis in several ways, providing experimental demonstrations in support of our claims. Our first goal is to provide a fine analysis of the "anatomy" of conservative Count-Min, describing its behavior in different regimes. Our main novel contribution is the demonstration that assigning different number of hash functions to different elements can significantly improve the error, and, as a consequence, lead to memory saving. Another major extension concerns the probability distribution of input elements: here we study non-uniform distributions as well, in particular step distribution and Zipf's distribution, and analyze the behavior of Count-Min for these distributions. This analysis is important not only because non-uniform distributions commonly occur in practice, but also because this provides important insights for the *heavy hitters* problem [7,11,23]). In particular, we consider the "small memory regime" (*supercritical*, in our terminology) when the number of distinct represented elements is considerably larger than the size of the data structure, and analyse conditions under which most frequent elements are evaluated with negligible error. This has direct applications to the frequent elements problem.

2 Background and Related Work

2.1 Conservative Count-Min: Definitions

A Count-Min sketch is a counter array A of size n together with a set of hash functions mapping elements (keys) of a given universe U to $[1..n]$. In this work, each element $e \in U$ can in general be assigned a different number k_e of hash functions. Hash functions are assumed fully random, therefore we assume w.l.o.g. that an element e is assigned hash functions $h_1, \ldots, h_{k_e}$.

At initialization, counters $A[i]$ are set to 0. When processing an insertion of an input element e, basic Count-Min increments by 1 each counter $A[h_i(e)]$, $1 \leq i \leq k_e$. The conservative version of Count-Min increments by 1 only the smallest of all $A[h_i(e)]$. That is, $A[h_i(e)]$ is incremented by 1 if and only if $A[h_i(e)] = \min_{1 \leq j \leq k_e}\{A[h_j(e)]\}$ and is left unchanged otherwise.

In both versions, the *estimate* of the number of occurrences of a queried element e is computed by $c(e) = \min_{1 \leq i \leq k_e}\{A[h_i(e)]\}$. It is easily seen that for any input sequence of elements, the estimate computed by original Count-Min is greater than or equal to the one computed by the conservative version.

In this work, we study the conservative version of Count-Min. Let H denote a selection of hash functions $H = \{h_1, h_2, \ldots\}$. Consider an input sequence I of N insertions and let E be the set of distinct elements in I. The *relative error* of an element e is defined by $err(e) = (c(e) - occ(e))/occ(e)$, where $occ(e)$ is the number of occurrences of e in the input. The *combined error* is an average error over all elements in I weighted by the number of occurrences, i.e.

$$err = \frac{1}{N} \sum_{e \in E} occ(e) \cdot err(e) = \frac{1}{N} \sum_{e \in E} (c(e) - occ(e)).$$

We assume that I is an i.i.d. random sequence drawn from a probability distribution on a set of elements $E \subseteq U$. A key parameter is the size of E relative to the size n of A. By analogy to hash tables, $\lambda = |E|/n$ is called the *load factor*, or simply the *load*.

2.2 Analysis of Conservative Count-Min: Prior Works

Motivated by applications to traffic monitoring, [5] was probably the first work devoted to the analysis of conservative Count-Min in the counting setting. Their model assumed that all $\binom{n}{k}$ counter combinations are equally likely, where k hash functions are applied to each element. This implies the regime when $|E| \gg n$. The focus of [5] was on the analysis of the *growth rate* of counters, i.e. the average number of counter increments per insertion, using a technique based on Markov chains and differential equations. Another approach proposed in [16] simulates a conservative Count-Min sketch by a hierarchy of ordinary Bloom filters. Obtained error bounds are expressed via a recursive relation based on false positive rates of corresponding Bloom filters.

Recent works [2,3] propose an analytical approach for computing error bounds depending on element probabilities assumed independent but not necessarily uniform, in particular leading to improved precision bounds for detecting heavy hitters. However the efficiency of this technique is more limited when all element probabilities are small. In particular, if the input distribution is uniform, their approach does not bring out any improvement over the general bounds known for original Count-Min.

In our recent work [21], we proposed an analysis of conservative Count-Min based on its relationship with random hypergraphs. We summarize the main results of this work below in Sect. 2.4.

2.3 Hash Hypergraph

Many hashing-based data structures are naturally associated with hash hypergraphs so that hypergraph properties are directly related to the proper functioning of the data structure. This is the case with Cuckoo hashing [27] and Cuckoo filters [18], Minimal Perfect Hash Functions and Static Functions [24], Invertible Bloom Lookup Tables [22], and some others. [30] provides an extended study of relationships between hash hypergraphs and some of those data structures.

A Count-Min sketch is associated with a *hash hypergraph* $H = (V, E)$ where $V = \{1..n\}$ and $E = \{\{h_1(e), ...h_{k_e}(e)\}\}$ over all distinct input elements e. We use notation $\mathcal{H}_{n,m}$ for hypergraphs with n vertices and m edges, and $\mathcal{H}^k_{n,m}$ for k-uniform such hypergraphs, where all edges have cardinality k. In the latter case, since our hash functions are assumed fully random, a hash hypergraph is a k-uniform Erdős-Rényi random hypergraph.

As inserted elements are assumed to be drawn from a random distribution, it is convenient to look at the functioning of a Count-Min sketch as a stochastic process on the associated hash hypergraph [21]. Each vertex holds a counter initially set to zero, and therefore each edge is associated with a set of counters held by corresponding vertices. Inserting an element consists in incrementing the minimal counters of the corresponding edge, and retrieving the estimate of an element returns the minimum value among the counters of the corresponding edge. From now on in our presentation, we will interchangeably speak of distinct elements and edges of the associated hash hypergraph, as well as of counters and vertices. Thus, we will call the *vertex value* the value of the corresponding counter, and the *edge value* the estimate of the corresponding element. Also, we will speak about the *load* of a hypergraph understood as the density $|E|/|V|$.

2.4 Hypergraph Peelability and Phase Transition of Error

A hypergraph $H = (V, E)$ is called *peelable* if iterating the following step starting from H results in the empty graph: if the graph has a vertex of degree 1 or 0, delete this vertex together with the incident edge (if any). As many other properties of random hypergraphs, peelability undergoes a phase transition. Consider the Erdős-Rényi k-uniform hypergraph model where graphs are drawn from $\mathcal{H}^k_{n,m}$ uniformly at random. It is shown in [26] that a phase transition occurs at a

(computable) peelability threshold λ_k: a random graph from $\mathcal{H}^k_{n,\lambda n}$ is with high probability (w.h.p.) peelable if $\lambda < \lambda_k$, and w.h.p. non-peelable if $\lambda > \lambda_k$. The first values are $\lambda_2 = 0.5$, $\lambda_3 \approx 0.818$, $\lambda_4 \approx 0.772$, etc., λ_3 being the largest. Note that the case $k = 2$ makes an exception to peelability: for $\lambda < \lambda_2$, a negligible fraction of vertices remain after peeling.

Peelability is known to be directly relevant to certain constructions of Minimal Perfect Hash Functions [24] as well as to the proper functioning of Invertible Bloom filters [22]. In [21], we proved that it is relevant to Count-Min as well.

Theorem 1 ([21]). *Consider a conservative Count-Min where each element is hashed using k random hash functions. Assume that the input I of length N is drawn from a* uniform *distribution on a set $E \subseteq U$ of elements and let $\lambda = |E|/n$, where n is the number of counters. If $\lambda < \lambda_k$, then for a randomly chosen element e, the relative error $err(e)$ is $o(1)$ w.h.p. when both n and N/n grow.*

In the complementary regime $\lambda > \lambda_k$, we showed in [21], under some additional assumptions, that *err* is $\Theta(1)$. Thus, the peelability threshold for random hash hypergraphs corresponds to phase transition in the error produced by conservative Count-Min for uniform distribution of input. We call regimes $\lambda < \lambda_k$ and $\lambda > \lambda_k$ *subcritical* and *supercritical*, respectively.

2.5 Variable Number of Hash Functions: Mixed Hypergraphs

The best peelability threshold $\lambda_3 \approx 0.818$ can be improved in at least two different ways. One way is to use a carefully defined class of hash functions which replace uniform sampling of k-edges by a specific non-uniform sampling. Thus, [15] showed that the peelability threshold can be increased to $\approx$0.918 for $k = 3$ and up to $\approx$0.999 for larger k's if a special class of hypergraphs is used.

Another somewhat surprising idea, that we apply in this paper, is to apply a different number of hash functions to differents elements, that is to consider non-uniform hypergraphs. Following [14], [28] showed that non-uniform hypergraphs may have a larger peelability threshold than uniform ones. More precisely, [28] showed that *mixed hypergraphs* with two types of edges of different cardinalities, each constituting a constant fraction of all edges, may have a larger peelability threshold: for example, hypergraphs with a fraction of $\approx$0.887 of edges of cardinality 3 and the remaining edges of cardinality 21 have the peelability threshold $\approx$0.920, larger than the best threshold 0.818 achieved by uniform hypergraphs. We adopt the notation of [28] for mixed hypergraphs: by writing $k = (k_1, k_2)$ we express that the hypergraph contains edges of cardinality k_1 and k_2, and $k = (k_1, k_2; \alpha)$ specifies in addition that the fraction of k_1-edges is α.

The idea of using different number of hash functions for different elements has also appeared in data structures design. [6] proposed *weighted Bloom filters* which apply a different number of hash functions depending on the frequency with which elements are queried and on probabilities for elements to belong to the set. It is shown that this leads to a reduced false positive probability, where the latter is defined to be weighted by query frequencies. This idea was further refined in [31], and then further in [4], under the name *Daisy Bloom filter*.

3 Results

3.1 Uniform Distribution

We start with the case where input elements are uniformly distributed, i.e. edges of the associated hash hypergraph have equal probabilities to be processed for updates.

Subcritical Regime. Theorem 1 in conjunction with the results of Sect. 2.5 leads to the assumption that using a different number of hash functions for different elements one could "extend" the regime of $o(1)$ error of Count-Min sketch, which can be made into a rigorous statement (for simplicity we only give it with two different edge cardinalities).

Theorem 2. *Consider a conservative Count-Min with n counters. Assume that the input of length N is drawn from a uniform distribution on $E \subseteq U$ and let $\lambda < \lambda_k$. Assume further that elements of E are hashed according to a mixed hypergraph model $k = (k_1, k_2; \alpha)$. Let c_k be the peelability ratio associated to k. Then, when $\lambda < c_k$, the relative error $err(e)$ of a randomly chosen key e is $o(1)$ w.h.p., as both n and N/n grow.*

The proof can be found in the full version [20].

Figure 1 shows the average relative error as a function of the load factor for three types of hypergraphs: 2-uniform, 3-uniform and mixed hypergraph where a 0.885 fraction of edges are of cardinality 3 and the remaining ones are of cardinality 14. 2-uniform and 3-uniform hypergraphs illustrate phase transitions at load factors approaching respectively 0.5 and ≈0.818, peelability thresholds for 2-uniform and 3-uniform hypergraphs respectively. It is clearly seen that the phase transition for the mixed hypergraphs occurs at a larger value approaching ≈0.898 which is the peelability threshold for this class of hypergraphs [28].

While this result follows by combining results of [28] and [21], it has not been observed earlier and has an important practical consequence: *using a variable number of hash functions in Count-Min sketch allows one to increase the load factor while keeping negligibly small error*. In particular, for the same input, this leads to space saving compared to the uniform case.

Note that parameters $k = (3, 14; 0.885)$ are borrowed from [28] in order to make sure that the phase transition corresponds to the peelability threshold obtained in [28]. In practice, "simpler" parameters can be chosen, for example we found that $k = (2, 5; 0.5)$ produces essentially the same curve as $k = (3, 14; 0.885)$ (data not shown).

Supercritical Regime. When the load factor becomes large (supercritical regime), the situation changes drastically. When the load factor just surpasses the threshold, some edges are still evaluated with small or zero error, whereas for the other edges, the error becomes large. This "intermediate regime" has been illustrated in [21]. When the load factor goes even larger, the multi-level pattern of edge values disappears and all edge values become concentrated around the

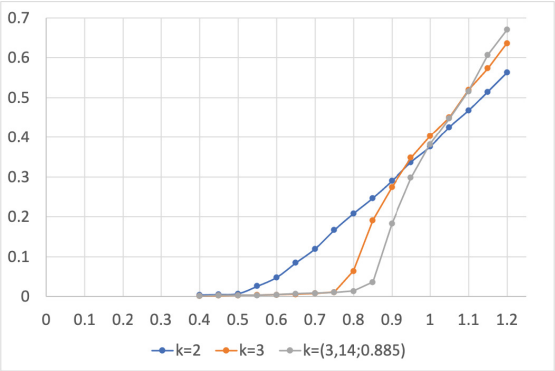


Fig. 1. *err* for small $\lambda = m/n$, for uniform distribution and different types of hypergraphs: 2-uniform, 3-uniform and (3,14)-mixed with a fraction of 0.885 of 3-edges (parameters borrowed from [28]). Data obtained for $n = 1000$. The input size in each experiment is $5,000$ times the number of edges. Each average is taken over 10 random hypergraphs.

same value. We call this phenomenon *saturation*. For example, for $k = 3$ saturation occurs at around $\lambda = 6$ (data not shown). Under this regime, the hash hypergraph is dense enough so that its specific topology is likely to be irrelevant and the largest counter level "percolates" into all vertex counters. In other words, all counters grow at the same rate, without any of them "lagging behind" because of particular graph structural patterns (such as edges containing leaf vertices).

3.2 Step Distribution

In this section, we focus on the simplest non-uniform distribution – *step distribution* – in order to examine the behavior of Count-Min sketch in presence of elements with different frequencies. Our model is as follows. We assume that input elements are classified into two groups that we call *hot* and *cold*, where a hot element has a larger appearance probability than a cold one. Note that we assume that we have a prior knowledge on whether a given element belongs to hot or cold ones. This setting is similar to the one studied for Bloom filters augmented with prior membership and query probabilities [4]. Note that our definition of *err* assumes that the query probability of an element and its appearance probability in the input are equal.

We assume that the load factors of hot and cold elements are λ_h and λ_c respectively. That is, there are $\lambda_h n$ hot and $\lambda_c n$ cold edges in the hash hypergraph. $G > 1$, called *gap factor*, denotes the ratio between probabilities of a hot and a cold element respectively. Let p_h (resp. p_c) denote the probability for an input element to be hot (resp. cold). Then $p_h/p_c = G\lambda_h/\lambda_c$, and since $p_h + p_c = 1$, we have

$$p_h = \frac{G\lambda_h}{\lambda_c + G\lambda_h}, \quad p_c = \frac{\lambda_c}{\lambda_c + G\lambda_h}.$$

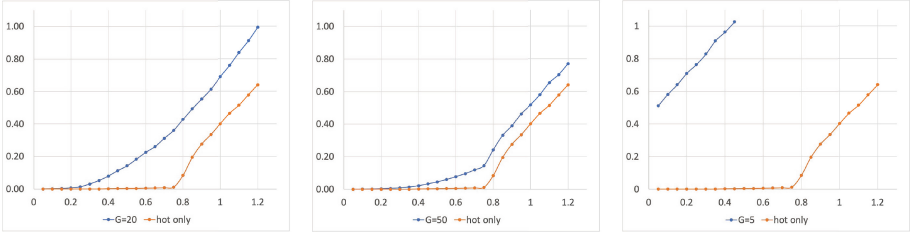
For example, if there are 10 times more distinct cold elements than hot ones ($\lambda_h/\lambda_c = 0.1$) but each hot element is 10 times more frequent than a cold one ($G = 10$), than we have about the same fraction of hot and cold elements in the input ($p_h = p_c = 0.5$).

In the rest of this section, we will be interested in the combined error of hot elements alone, denoted *errhot*. If $E_h \subseteq E$ is the subset of hot elements, and N_h is the total number of occurrences of hot elements in the input, then *errhot* is defined by

$$errhot = \frac{1}{N_h} \sum_{e \in E_h} occ(e) \cdot err(e) = \frac{1}{N_h} \sum_{e \in E_h} (c(e) - occ(e)).$$

"Interaction" of Hot and Cold Elements. A partition of elements into hot and cold induces the partition of the underlying hash hypergraph into two subgraphs that we call *hot* and *cold subgraphs* respectively. Since hot elements have larger counts, one might speculate that counters associated with hot edges are larger than counts of cold elements and therefore are not incremented by those. Then, *errhot* is entirely defined by the hot subgraph, considered under the uniform distribution of elements. In particular, *errhot* as a function of λ_h should behave the same way as *err* for the uniform distribution (see Sect. 3.1).

This conjecture, however, is not true in general. One reason is that there is a positive probability that all nodes of a cold edge are incident to hot edges as well. As a consequence, "hot counters" (i.e. those incident to hot edges) gain an additional increment due to cold edges, and the latter contribute to the overestimate of hot edge counts. Fig. 2a illustrates this point. It shows, for $k = 3$, *errhot* as a function of λ_h in presence of cold elements with $\lambda_c = 5$, for the gap value $G = 20$. For the purpose of comparison, the orange curve shows the error for the uniform distribution (as in Fig. 1), that is the error that hot elements would have if cold elements were not there. We clearly observe the contribution of cold elements to the error, even in the load interval below the peelability threshold.



(a) $G = 20$ (b) $G = 50$ (c) $G = 5$

Fig. 2. *errhot* for $k = 3$ depending on λ_h, in presence of cold elements with $\lambda_c = 5$ (blue curves) and without any cold elements (orange curve). (Color figure online)

Figure 2b illustrates that when the gap becomes larger (here, $G = 50$), the contribution of cold elements diminishes and the curve approaches the one of the uniform distribution. A larger gap leads to larger values of hot elements and, as a consequence, to a smaller relative impact of cold ones.

Another reason for which the above conjecture may not hold is the following: even if the number of hot elements is very small but the gap factor is not large enough, the cold edges may cause the counters to become large if λ_c is large enough, in particular in the saturation regime described in Sect. 3. As a consequence, the "background level" of counters created by cold edges may be larger than true counts of hot edges, causing their overestimates. As an example, consider again the configuration with $k = 3$ and $\lambda_c = 5$. The cold elements taken alone would have an error of about 6 on average ($\approx$6.25, to be precise, data not shown) which means an about 7$\times$ overestimate. Since the graph is saturated in this regime (see Sect. 3), this means that most of the counters will be about 7 times larger than counts of cold edges. Now, if a hot element is only 5 times more frequent than a cold one, those will be about 1.4$\times$ overestimated, i.e. will have an error of about 0.4, This situation is illustrated in Fig. 2c.

Mixed Hypergraphs. The analysis above shows that in presence of a "background" formed by large number of cold elements, the error of hot elements starts growing for much smaller load factors than without cold elements, even if the latter are much less frequent than the former. Inspired by results of Sect. 3, one may ask if the interval of negligible error can be extended by employing the idea of variable number of hash functions. Note that here this idea applies more naturally by assigning a different number of hash functions to hot and cold elements.

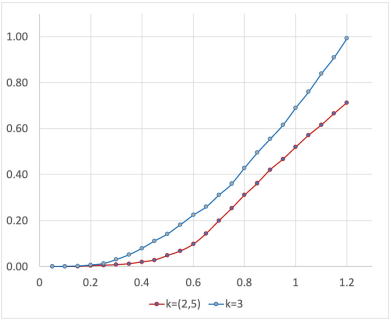


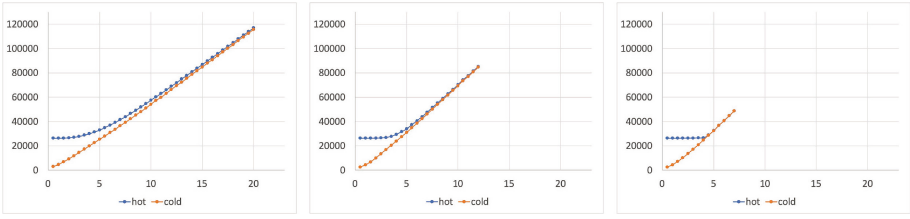
Fig. 3. *errhot* as a function of λ_h for $k = 3$, $\lambda_c = 5$ and $G = 20$ (same as in Fig. 2a) vs. $k = (2, 5)$ for hot and cold elements respectively

Figure 3 illustrates that this is indeed possible by assigning a smaller number of hash functions to hot elements and a larger number to cold ones. It is clearly seen that the interval supporting close-to-zero errors is extended. This happens

because when the hot subgraph is not too dense, increasing the cardinality of cold edges leads to a higher probability that at least one of the vertices of such an edge is not incident to a hot edge. As a consequence, this element does not affect the error of hot edges. For the same reason, decreasing the cardinality of hot edges (here, from 3 to 2) improves the error, as this increases the fraction of vertices non-incident to hot edges.

Saturation in Supercritical Regime. In Sect. 3 we discussed the saturation regime occurring for large load values: when the load grows sufficiently large, i.e. the hash hypergraph becomes sufficiently dense, all counters reach the same level, erasing distinctions between edges. In this regime, assuming a fixed load (graph density) and the uniform distribution of input, the edge value depends only on input size and not on the graph structure (with high probability).

It is an interesting, natural and practically important question whether this saturation phenomenon holds for non-uniform distributions as well, as it is directly related to the capacity of distinguishing elements of different frequency. A full and precise answer to this question is not within the scope of this work. We believe that the answer is positive at least when the distribution is piecewise uniform, when edges are partitioned into several classes and are equiprobable within each class, provided that each class takes a linear fraction of all elements. Here we illustrate this thesis with the step distribution.



(a) 2-uniform (b) 3-uniform (c) (2, 5)-mixed

Fig. 4. Convergence of average estimates of hot and cold elements for 2-uniform (4a), 3-uniform (4b) and (2,5)-mixed (4c) hypergraphs. x-axis shows the total load $\lambda = \lambda_h + \lambda_c$ with $\lambda_h = 0.1 \cdot \lambda$ and $\lambda_c = 0.9 \cdot \lambda$ and $G = 10$ in all cases.

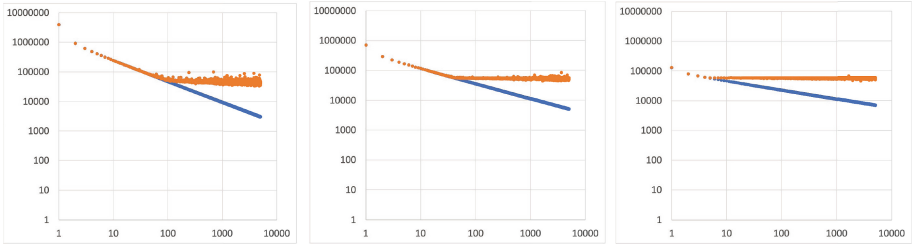
Figure 4 illustrates the saturation phenomenon by showing average values of hot and cold edges ($G = 10$) with three different configurations: 2-uniform, 3-uniform, and (2,5)-mixed. Note that the x-axis shows here the total load $\lambda = \lambda_h + \lambda_c$, where $\lambda_h = 0.1 \cdot \lambda$ and $\lambda_c = 0.9 \cdot \lambda$. That is, the number of both hot and cold edges grows linearly when the total number of edges grows.

One can observe that in all configurations, values of hot and cold edges converge, which is a demonstration of the saturation phenomenon. Interestingly, the "convergence speed" heavily depends on the configuration: the convergence is "slower" for uniform configurations, whereas in the mixed configuration, it occurs right after the small error regime for hot edges.

3.3 Zipf's Distribution

Power law distributions are omnipresent in practical applications. The simplest of those is Zipf's distribution which is often used as a test case for different algorithms including Count-Min sketches [3,5,8,13,16]. Under Zipf's distribution, element probabilities in descending order are proportional to $1/i^{\beta}$, where i is the rank of the key and $\beta \geq 0$ is the *skewness* parameter. Note that for $\beta = 0$, Zipf's distribution reduces to the uniform one.

Zipf's distribution is an important test case for our study as well, as it forces several (few) most frequent elements to have very large counts and a large number of elements (*heavy tail*) to have small counts whose values decrease only polynomially on the element rank and are therefore of the same order of magnitude. Bianchi et al. [5, Fig. 1] observed that for Zipf's distribution in the supercritical regime, the estimates follow the "waterfall-type behavior": the most frequent elements have essentially exact estimates whereas the other elements have all about the same estimate regardless of their frequency. Figure 5 illustrates this phenomenon for different skewness values.



(a) $\beta = 0.7$ (b) $\beta = 0.5$ (c) $\beta = 0.3$

Fig. 5. Exact (blue) and estimated (orange) edge values for Zipf's distribution as a function on the element frequency rank, plotted in double log scale. All plots obtained for $n = 1000$, $\lambda = 5$, $k = 2$, and the input size $50 \cdot 10^6$. Estimates are averaged over 10 hash function draws. (Color figure online)

The waterfall-type behavior for Zipf's distribution is well explained by the analysis we developed in the previous sections. The "waterfall pool level" of values (called *error floor* in [5]) is the effect of saturation formed by heavy tail elements. The few "exceptionally frequent" elements are too few to affect the saturation level (their number is $\ll n$), they turn out to constitute "peaks" above the level and are thus estimated without error. Naturally, smaller skewness values make the distribution less steep and reduce the number of "exceptionally frequent" elements. For example, according to Fig. 5, for $\lambda = 5$ and $k = 2$, about 50 most frequent elements are evaluated without error for $\beta = 0.7$, about 40 for $\beta = 0.5$ and only 5 for $\beta = 0.3$.

Following our results from previous sections, we studied whether using a variable number of hash functions can extend the range of frequent elements

estimated with small error. We found that for moderate loads λ, this is possible indeed. More specifically, using a variable number of hash functions can lead to a sharper "break point" compared to the constant number of hash functions, see Fig. 5. As a result, although the "waterfall pool level" may be higher, a larger range of most frequent elements are evaluated with small error. This observation matches the phenomenon illustrated earlier in Fig. 4. Due to space limitation, we refer to the full version [20] for the data illustrating this point.

4 Conclusions

In this paper, we presented a series of experimental results providing new insights into the behavior of conservative Count-Min sketch. Some of them have direct applications to practical usage of this data structure. Main results can be summarized as follows.

- For the uniform distribution of input elements, assigning a different number of hash functions to different elements extends the subcritical regime (range of load factors λ) that supports asymptotically vanishing relative error. This immediately implies space saving for Count-Min configurations verifying this regime. For non-uniform distributions, variable number of hash functions allows extending the regime of negligible error for most frequent elements,
- Under "sufficiently uniform distributions", including uniform and step distributions, a Count-Min sketch reaches a saturation regime when λ becomes sufficiently large. In this regime, counters become concentrated around the same value and elements with different frequency become indistinguishable,
- Frequent elements that can be estimated with small error can be seen as those which surpass the saturation level formed by the majority of other elements. For example, in case of Zipf's distribution, those elements are a few "exceptionally frequent elements", whereas the saturation is insured by the heavy-tail elements. Applying a variable number of hash functions can increase the number of those elements for moderate loads λ.

Many of those results lack a precise mathematical analysis. Perhaps the most relevant to practical usage of Count-Min is the question of saturation level ("waterfall pool level"), as it provides a lower bound to the frequency of elements that will be estimated with small error, which in turn is a fundamental information for heavy-hitter type of applications. Bianchi et al. [5] observed that in the case of non-uniform distribution of input elements, the "waterfall pool level" is upper-bounded by the saturation level for the uniform distribution of input. This latter is computed in [5] using a method based on Markov chains and differential equations. We believe that this method can be extended to the case of mixed graphs as well and leave it for future work. However, providing an analysis for more complex distributions including Zipf's distribution is an open problem.

References

1. Behera, S., Gayen, S., Deogun, J.S., Vinodchandran, N.: KmerEstimate: a streaming algorithm for estimating k-mer counts with optimal space usage. In: Proceedings of the 2018 ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics, pp. 438–447 (2018)
2. Ben Mazziane, Y., Alouf, S., Neglia, G.: A formal analysis of the count-min sketch with conservative updates. In: IEEE INFOCOM WNA 2022 - The second Workshop on Networking Algorithms (WNA), New York, USA (2022). https://doi.org/10.1109/INFOCOMWKSHPS54753.2022.9798146
3. Ben Mazziane, Y., Alouf, S., Neglia, G.: Analyzing count min sketch with conservative updates. Comput. Netw. **217**, 109315 (2022). https://www.sciencedirect.com/science/article/pii/S1389128622003607
4. Bercea, I.O., Houen, J.B.T., Pagh, R.: Daisy Bloom filters. CoRR abs/2205.14894 (2022)
5. Bianchi, G., Duffy, K., Leith, D.J., Shneer, V.: Modeling conservative updates in multi-hash approximate count sketches. In: 24th International Teletraffic Congress, ITC 2012, Kraków, Poland, 4–7 September 2012, pp. 1–8. IEEE (2012). https://ieeexplore.ieee.org/document/6331813/
6. Bruck, J., Gao, J., Jiang, A.: Weighted Bloom filter. In: Proceedings 2006 IEEE International Symposium on Information Theory, ISIT 2006, The Westin Seattle, Seattle, Washington, USA, 9–14 July 2006, pp. 2304–2308. IEEE (2006). https://doi.org/10.1109/ISIT.2006.261978
7. Charikar, M., Chen, K., Farach-Colton, M.: Finding frequent items in data streams. Theor. Comput. Sci. **312**(1), 3–15 (2004)
8. Chen, P., Wu, Y., Yang, T., Jiang, J., Liu, Z.: Precise error estimation for sketch-based flow measurement. In: Proceedings of the 21st ACM Internet Measurement Conference, IMC 2021, pp. 113–121. Association for Computing Machinery, New York (2021). https://doi.org/10.1145/3487552.3487856
9. Cohen, S., Matias, Y.: Spectral Bloom filters. In: Halevy, A.Y., Ives, Z.G., Doan, A. (eds.) Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data, San Diego, California, USA, 9–12 June 2003, pp. 241–252. ACM (2003). https://doi.org/10.1145/872757.872787
10. Cormode, G.: Count-min sketch. In: Liu, L., Özsu, M.T. (eds.) Encyclopedia of Database Systems, 2nd edn., pp. 653–659. Springer, New York (2018). https://doi.org/10.1007/978-1-4614-8265-9_87
11. Cormode, G., Hadjieleftheriou, M.: Finding frequent items in data streams. Proc. VLDB Endow. **1**(2), 1530–1541 (2008)
12. Cormode, G., Muthukrishnan, S.: An improved data stream summary: the count-min sketch and its applications. J. Algorithms **55**(1), 58–75 (2005)
13. Cormode, G., Muthukrishnan, S.: Summarizing and mining skewed data streams. In: Kargupta, H., Srivastava, J., Kamath, C., Goodman, A. (eds.) Proceedings of the 2005 SIAM International Conference on Data Mining, SDM 2005, Newport Beach, CA, USA, 21–23 April 2005, pp. 44–55. SIAM (2005). https://doi.org/10.1137/1.9781611972757.5
14. Dietzfelbinger, M., Goerdt, A., Mitzenmacher, M., Montanari, A., Pagh, R., Rink, M.: Tight thresholds for cuckoo hashing via XORSAT. In: Abramsky, S., Gavoille, C., Kirchner, C., Meyer auf der Heide, F., Spirakis, P.G. (eds.) ICALP 2010. LNCS, vol. 6198, pp. 213–225. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14165-2_19

15. Dietzfelbinger, M., Walzer, S.: Dense peelable random uniform hypergraphs. In: Bender, M.A., Svensson, O., Herman, G. (eds.) 27th Annual European Symposium on Algorithms, ESA 2019, Munich/Garching, Germany, 9–11 September 2019. LIPIcs, vol. 144, pp. 38:1–38:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2019). https://doi.org/10.4230/LIPIcs.ESA.2019.38
16. Einziger, G., Friedman, R.: A formal analysis of conservative update based approximate counting. In: International Conference on Computing, Networking and Communications, ICNC 2015, Garden Grove, CA, USA, 16–19 February 2015, pp. 255–259. IEEE Computer Society (2015). https://doi.org/10.1109/ICCNC.2015.7069350
17. Estan, C., Varghese, G.: New directions in traffic measurement and accounting. In: Mathis, M., Steenkiste, P., Balakrishnan, H., Paxson, V. (eds.) Proceedings of the ACM SIGCOMM 2002 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication, Pittsburgh, PA, USA, 19–23 August 2002, pp. 323–336. ACM (2002). https://doi.org/10.1145/633025.633056
18. Fan, B., Andersen, D.G., Kaminsky, M., Mitzenmacher, M.D.: Cuckoo filter: practically better than Bloom. In: Proceedings of the 10th ACM International on Conference on Emerging Networking Experiments and Technologies, CoNEXT 2014, pp. 75–88. Association for Computing Machinery, New York (2014). https://doi.org/10.1145/2674005.2674994
19. Fan, L., Cao, P., Almeida, J., Broder, A.: Summary cache: a scalable wide-area web cache sharing protocol. IEEE/ACM Trans. Netw. **8**(3), 281–293 (2000). https://doi.org/10.1109/90.851975
20. Fusy, É., Kucherov, G.: Count-min sketch with variable number of hash functions: an experimental study. CoRR abs/2302.05245 (2023). https://doi.org/10.48550/arXiv.2302.05245, to appear in SPIRE'23
21. Fusy, É., Kucherov, G.: Phase transition in count approximation by count-min sketch with conservative updates. In: Mavronicolas, M. (ed.) CIAC 2023. LNCS, vol. 13898, pp. 232–246. Springer, Cham (2023). https://doi.org/10.1007/978-3-031-30448-4_17. Full version in arxiv:2203.15496
22. Goodrich, M.T., Mitzenmacher, M.: Invertible Bloom lookup tables. In: 2011 49th Annual Allerton Conference on Communication, Control, and Computing (Allerton), pp. 792–799. IEEE (2011)
23. Liu, H., Lin, Y., Han, J.: Methods for mining frequent items in data streams: an overview. Knowl. Inf. Syst. **26**(1), 1–30 (2011)
24. Majewski, B.S., Wormald, N.C., Havas, G., Czech, Z.J.: A family of perfect hashing methods. Comput. J. **39**(6), 547–554 (1996)
25. Mohamadi, H., Khan, H., Birol, I.: ntCard: a streaming algorithm for cardinality estimation in genomics data. Bioinformatics **33**(9), 1324–1330 (2017)
26. Molloy, M.: Cores in random hypergraphs and Boolean formulas. Random Struct. Algorithms **27**(1), 124–135 (2005)
27. Pagh, R., Rodler, F.F.: Cuckoo hashing. J. Algorithms **51**(2), 122–144 (2004)
28. Rink, M.: Mixed hypergraphs for linear-time construction of denser hashing-based data structures. In: van Emde Boas, P., Groen, F.C.A., Italiano, G.F., Nawrocki, J., Sack, H. (eds.) SOFSEM 2013. LNCS, vol. 7741, pp. 356–368. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-35843-2_31
29. Shibuya, Y., Kucherov, G.: Set-min sketch: a probabilistic map for power-law distributions with application to k-mer annotation. bioRxiv, p. 2020.11.14.382713 (2020). https://doi.org/10.1101/2020.11.14.382713

30. Walzer, S.: Random hypergraphs for hashing-based data structures. Ph.D. thesis, Technische Universität Ilmenau, Germany (2020). https://www.db-thueringen.de/receive/dbt_mods_00047127
31. Wang, X., Ji, Y., Dang, Z., Zheng, X., Zhao, B.: Improved weighted bloom filter and space lower bound analysis of algorithms for approximated membership querying. In: Renz, M., Shahabi, C., Zhou, X., Cheema, M.A. (eds.) DASFAA 2015. LNCS, vol. 9050, pp. 346–362. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-18123-3_21