On the Evolutionary of Bloom Filter False Positives - An Information Theoretical Approach to Optimizing Bloom Filter Parameters

Zhuochen Fan, Gang Wen, Zhipeng Huang, Yang Zhou, Qiaobin Fu, Tong Yang, *Member, IEEE*, Alex X. Liu, *Fellow, IEEE*, and Bin Cui, *Senior Member, IEEE*

Abstract—The fundamental issue of how to calculate the false positive probability of widely used Bloom Filters (BF), from which the conventional wisdom is to derive the optimal value of k, remains elusive. Since Bloom gave the false positive formula in 1970, in 2008, Bose et al. pointed out that Bloom's formula is flawed; and in 2010, Christensen et al. pointed out that Bose's formula is also flawed and gave another formula. Although Christensen's formula is perfectly accurate, it is time-consuming and impossible to calculate the optimal value of k. Based on the following observation: for a BF with m bits and n elements, if and only if its entropy is the largest, its false positive probability is the smallest, we propose the first approach to calculating the optimal k without any false positive formula. Furthermore, we propose a new and more accurate upper bound for the false positive probability. When the size of a Bloom Filter becomes infinitely large, our upper bound turns equal to the lower bound, which becomes Bloom's formula and deepens our understanding towards it. Besides, we derive the bounds of correct rate of Counting Bloom Filters (CBFs) by applying our proposed formulas about BFs to them.

Index Terms—Bloom filter, false positive, information entropy, compression, upper bound, lower bound, counting Bloom Filter

1 Introduction

1.1 Motivation

A Bloom Filter (BF) is a compact data structure used for quickly checking whether an element belongs to a set or not [1]. Given a set S of n elements, we create a bit array A of length m as follows. First, we initialize each bit of A to 0, then for each element $x \in \mathcal{S}$, we use k hash functions to compute k hash values: $h_1(x), h_2(x), \cdots, h_k(x)$ where each hash value is in the range $[1, m]$. Second, for each $1 \leqslant i \leqslant k$, we let $A[h_i(x)] = 1$. The resulting bit array A is called the BF for set S. To query whether $y \in S$, we first use the same k hash functions to compute k hash values: $h_1(y), h_2(y), \cdots, h_k(y)$. Second, we check whether the corresponding k bits in A are all $1s$ (i.e., whether $A[h_1(y)] \wedge A[h_2(y)] \wedge \cdots, A[h_k(y)] = 1$ holds); if yes, then $y \in S$ may probably hold and we can further check whether $y \in S$; if no, then $y \in S$ definitely does not hold. The cases that the BF shows that $y \in S$ may hold but actually $y \notin S$ are called false positives (FP). The FP probability f can be calculated from n, k, and m. Thus, given a set of n elements and the required FP probability f, we can calculate the relationship between k and m. Based on the calculated relationship between k and m, we can properly trade off between space and speed: smaller m means smaller space, and smaller k means the smaller number of hash function calculations. In typical BF applications, as m is determined by the memory budget for the BF, with known values of n and f, we calculate the optimal value for the only unknown parameter k.

As set membership query is a fundamental operation in many applications [2], and BFs have the advantages of small memory consumption, fast query speed, and no false negatives, BFs have been widely used in networks, databases, data mining and analysis, and machine learning, *etc.* In terms of networks, BFs have applications in web caching [3], [4], [5], sensor networks [6], [7], [8], data center networks [9], [10], [11], cloud computing [12], [13], [14], and more [15], [16], [17]. In terms of databases, BFs have applications in key-value stores [18], [19], [20], [21], [22], [23], privacy-preserving record linkage [24], [25], [26], [27], [28], [29], block chain [30], [31], [32], and more [33], [34], [35]. In addition to the above, BFs also have applications that cannot be underestimated in data mining and analysis [36], [37], [38], [39], [40],

- *Zhuochen Fan, Zhipeng Huang, and Bin Cui are with the School of Computer Science and National Engineering Laboratory for Big Data Analysis Technology and Application, Peking University, Beijing 100871, China. E-mail: {fanzc, huangzpp, bin.cui}@pku.edu.cn.*
- *Gang Wen is with the School of Mathematical Sciences, Peking University, Beijing 100871, China. E-mail: jnwengang@pku.edu.cn.*
- *Yang Zhou is with the School of Engineering and Applied Sciences, Harvard University, Cambridge, MA 02138 USA. E-mail: yangzhou@g.harvard.edu.*
- *Qiaobin Fu is with Google Inc, Mountain View, CA 94043 USA. E-mail: qiaobinf@gmail.com.*
- *Tong Yang is with the School of Computer Science and National Engineering Laboratory for Big Data Analysis Technology and Application, Peking University, Beijing 100871, China, and also with the Peng Cheng Laboratory, Shenzhen 518066, China. E-mail: yangtongemail@gmail.com.*
- *Alex X. Liu is with Software Engineering Institute and the Chief Security Officer of the Midea Group, Foshan 528311, China. E-mail: alexliu@midea.com.*

Manuscript received 16 May 2021; revised 7 May 2022; accepted 6 August 2022. Date of publication 19 August 2022; date of current version 5 June 2023.
This work was supported in part by the Key-Area Research and Development Program of Guangdong Province under Grant 2020B0101390001, and in part by the National Natural Science Foundation of China (NSFC) under Grants U20A20179 and 61832001.
(Corresponding author: Tong Yang.)
Recommended for acceptance by G. Chen.
Digital Object Identifier no. 10.1109/TKDE.2022.3200045

[41], [42], and machine learning [43], [44], [45], [46], [47]. Most applications with set membership query can potentially be optimized using BFs [48].

Although BFs have been widely used in many applications, the fundamental issue of how to calculate FP probability remains elusive. Properly calculating the FP probability of BF is critical because it is used to calculate the optimal value of the important parameter k, the number of hash functions. In [1], Bloom gave a formula for calculating the FP probability with known parameters n, k, and m. Based on Bloom's formula, we can also easily compute the optimal value of the parameter k when m and n are known. This formula has been believed to be correct until 2008 when Prosenjit Bose et al. pointed out that Bloom's formula is flawed and gave a new FP formula [49]. Interestingly, two years later, Ken Christensen et al. pointed out that Bose's formula is also flawed and gave a new FP formula [50]. So far, it is believed that Christensen's formula is perfectly accurate. However, Both Bose's and Christensen's FP formulas are too complicated to calculate the optimal value of k from given values of n and m.

1.2 Main Contributions

While the conventional wisdom is to derive the optimal value of BF parameter k from the FP probability, in this paper, we propose the first approach to calculating the optimal k without any FP formula. We first observe that for a BF with m bits and n elements, if and only if its entropy is the largest, its false positive probability is the smallest, according to information entropy theory. Based on this observation, our approach is to derive a formula for calculating the optimal k by letting the entropy equal to 1. We also propose another method to calculate FP probability by deriving the left and right limit expressions of FP probability. We prove that when m goes to infinity, the left and right limits are the same, which is essentially the FP probability. Interestingly, our derived FP formula is the same as Bloom's formula in [1]. This deepens our understanding of Bloom's formula: it is perfectly accurate when m is infinitely large, and it is practically accurate when m is sufficiently large.

In summary, we make three key contributions in this paper. First, we propose an information theoretical approach to calculating the optimal value of BF parameter k without calculating FP probability. Second, we propose a new upper bound which is much more accurate than state-of-the-art. When m is infinitely large, our upper bound becomes the same as the lower bound. This result formally proves that Bloom's formula is practically accurate when m is sufficiently large. Third, we conducted experiments to validate our findings. In particular, we show that the error of Bloom's formula is negligibly small when m is large. Furthermore, we release our source code of Bloom Filters at GitHub [51]. The rest of this paper proceeds as follows. In Section 2, we introduce the evolutionary on FP probability. In Section 3, we show the derivation of the optimal number of hash functions using the information entropy theory. In Section 4, we present a new upper bound of the false positive probability of Bloom Filters. In Section 5, we derive the bounds of correct rate of Counting Bloom Filters (CBFs) [3] through our proposed formulas about Bloom Filters. In Section 6, we conduct experiments to evaluate the error of Bloom Filters. We conclude the paper in Section 7.

TABLE 1
Symbols Frequently Used in This Paper

Symbol	Description
$\mathcal{S}$	Set of elements
m	BF size
n	Number of elements in $\mathcal{S}$
k	Number of hash functions
k^*	Optimal number of hash functions
f	False positive probability
f_{bloom}	False positive probability calculated by Bloom
f_{bose}	False positive probability calculated by Bose
f_{christ}	False positive probability calculated by Christensen
f_{true}	True false positive probability of BF
FP	false positive
BF	Bloom Filter

2 Prior Art on BF False Positives

In this section, we review the prior art on calculating the false positive probability of Bloom Filters. Table 1 summarizes the notations used in this paper.

2.1 Bloom's False Positive Formula

In 1970, Bloom calculated the false positive probability of a Bloom Filter as follows [1]. Given a set $\mathcal{S}$ of elements, let n be the number of elements in $\mathcal{S}$, k be the number of hash functions, and m be the number of bits in the Bloom Filter A constructed from set $\mathcal{S}$. In querying an element x, the false positive happens when the Bloom Filter reports that $x \in \mathcal{S}$ (i.e., $A[h_i(x)] = 1$ holds for each $1 \leqslant i \leqslant k$), but actually $x \notin \mathcal{S}$. Consider an arbitrary bit $A[b]$ in A. For any element in $\mathcal{S}$ and any hash function h_i ($1 \leqslant i \leqslant k$), the probability that this element is not hashed to bit $A[b]$ by h_i is $1 - 1/m$. As $\mathcal{S}$ has n elements and each element is hashed k times, the probability of $A[b] = 0$ is p':

$$p' = \left(1 - \frac{1}{m}\right)^{kn} \tag{1}$$

Thus, the probability of $A[b] = 1$ is $1 - (1 - 1/m)^{kn}$. For any element $x \notin \mathcal{S}$, the probability that the false positive happens for x, i.e., the probability of $A[h_1(y)] \wedge A[h_2(y)] \wedge \cdots, A[h_k(y)] = 1$, is calculated as follows:

$$f_{bloom} = \left(1 - \left(1 - \frac{1}{m}\right)^{kn}\right)^k \tag{2}$$

which can be approximated by $(1 - e^{\frac{-nk}{m}})^k$.

2.2 Bose's Derivation

In 2008, Bose et al. pointed out that the last step in Bloom's derivation is flawed because for any element $x \notin \mathcal{S}$, the k events $A[h_1(x)] = 1, A[h_2(x)] = 1, \cdots, A[h_k(x)] = 1$ are not actually independent [49]. Although for each bit $A[h_i(x)]$ ($1 \leqslant i \leqslant k$), after inserting n elements into array A, the probability of $A[h_i(x)] = 1$ is $1 - (1 - 1/m)^{kn}$, for the probability of $A[h_1(y)] \wedge A[h_2(y)] \wedge \cdots, A[h_k(y)] = 1$ to be $(1 - (1 - 1/m)^{kn})^k$, the k events $A[h_1(x)] = 1, A[h_2(x)] = 1, \cdots, A[h_k(x)] = 1$ need to be independent. Observing the dependency of the k events $A[h_1(x)] = 1, A[h_2(x)] = 1, \cdots, A[h_k(x)] = 1$, Bose et al. derive the following false positive formula:

$$f_{bose} = \frac{1}{m^{k(n+1)}} \sum_{i=1}^{m} i^k i! \binom{m}{i} \left\{ \begin{matrix} kn \\ i \end{matrix} \right\} \tag{3}$$

where

$$\left\{ \begin{matrix} kn \\ i \end{matrix} \right\} = \frac{1}{i!} \sum_{j=0}^{i} (-1)^{i-j} \binom{i}{j} j^{kn} \tag{4}$$

Bose et al. derived asymptotically closed forms for the upper and lower bounds of the above formula:

$$f_{bloom} < f_{bose} \leqslant f_{bloom} \times \left(1 + \mathcal{O}\left(\frac{k}{p} \sqrt{\frac{\ln m - k \ln p}{m}} \right) \right) \tag{5}$$

where $p = 1 - p' = 1 - (1 - \frac{1}{m})^{kn}$. These bounds hold under the condition that

$$\frac{k}{p} \sqrt{\frac{\ln m - k \ln p}{m}} \leqslant c \tag{6}$$

for some constant $c < 1$. Bose et al. further showed that for $k \geqslant 2$, f_{bose} is strictly larger than f_{bloom}, and the lower bound converges to f_{bloom} when m becomes infinitely large.

2.3 Christensen's Derivation

In 2010, Christensen et al. pointed out that Bose's formula has a mistake that the term $(-1)^j$ should be $(-1)^{i-j}$, but the lower and upper bounds in Eq. (5) are correct. Christensen et al. derived the finally correct false positive formula for Bloom Filters as follows:

$$f_{christ} = \frac{m!}{m^{k(n+1)}} \sum_{i=1}^{m} \sum_{j=1}^{i} (-1)^{i-j} \frac{j^{kn} i^k}{(m-i)! j! (i-j)!} \tag{7}$$

Although Christensen's formula is perfectly accurate, it is not much useful in practice. First, given the Bloom Filter parameters n, m, and f, it is difficult to calculate the optimal k value as Christensen's formula does not give a closed form expression for calculating the optimal k. Second, given the Bloom Filter parameters n, m, and k, the algorithm by Christensen et al. takes $\mathcal{O}(knm)$ time to calculate the false positive probability f, which is time-consuming.

2.4 Grandi's Derivation

In 2018, Grandi proposed the γ-transform [52] approach to analyze the false positive probability [53], and further derived the false positive formula as follows:

$$f_{grandi} = \sum_{x=0}^{m} \left(\frac{x}{m} \right)^k \binom{m}{x} \sum_{j=0}^{x} (-1)^j \binom{x}{j} \left(\frac{x-j}{m} \right)^{kn} \tag{8}$$

where x represents the number of bits set to 1 in the Bloom Filter.

3 Computing the Optimal K

Traditionally, the optimal number of hash functions is derived through finding the extrema of the asymptotic formula of f_{bloom} given in Eq. (2) as follows:

$$k^*_{bloom} = \frac{m}{n} \ln 2 \tag{9}$$

However, as we have discussed, the underlying formula for FP probability is not fully correct. In this section, we first present the important theorems that correlate information entropy and the false positive rate of Bloom Filters. Then we propose a method of deriving the optimal value of k by minimizing the FP probability given the value of BF size m and number of inserted elements n.

3.1 Information Entropy Basis

Information Entropy. In information theory, *information entropy* is used to measure the uncertainty of a random variable. In this paper, we use *Shannon entropy* [54], which measures the value of the information contained in a variable. Entropy is typically measured in bits, nats, or bans [55]. For a variable with s events with the probabilities of p_1, p_2, $\cdots$, p_s. The information entropy E is defined as:

$$E = -\sum_{i=1}^{s} p_i log_2 \frac{1}{p_i} \tag{10}$$

Property of Information Entropy: For any variable or message, if its information entropy is not at the maximum, it can be compressed without information losses. For a random variable, the larger the uncertainty is, the bigger the information entropy is. Suppose an m-bit string variable is compressed to m'-bit string variable without loss of information, the entropy E of the Bloom Filter remains the same. However, the average information containing in each bit, which is $e = E/m$ will increase to $e' = E/m'$. Then, we have $me = m'e'$. According to the information entropy formula (Eq. 10), we can obtain the information entropy e' of one bit of a BF as follows, where p' is defined in Eq. (1).

$$e' = -(p' \log_2 p' + (1-p') \log_2 (1-p')) \tag{11}$$

We illustrate the relationship between entropy e' and p' in Fig. 1. We can observe that the entropy of an m-bit sequence reaches the maximum value of 1 when $p' = 0.5$, i.e., the probability of each bit in the BF being 1 (or 0) is 50%.

3.2 An Important Global Assumption

What must be said before our formal analysis is that we assume after compression without information loss, a Bloom Filter is still a Bloom Filter. In other words, we assume the properties of query and insertion of the Bloom Filter should be persisted. Of course that's a very strong assumption, and we have not given a rigorous proof. However, since the results of our analysis based on this assumption are in perfect agreement with our experiments, we state it here as an open question.

3.3 Relation Between False Positive Probability and Information Entropy

We now show the relation between the false positive probability and information entropy. We first present a lemma and some definitions.

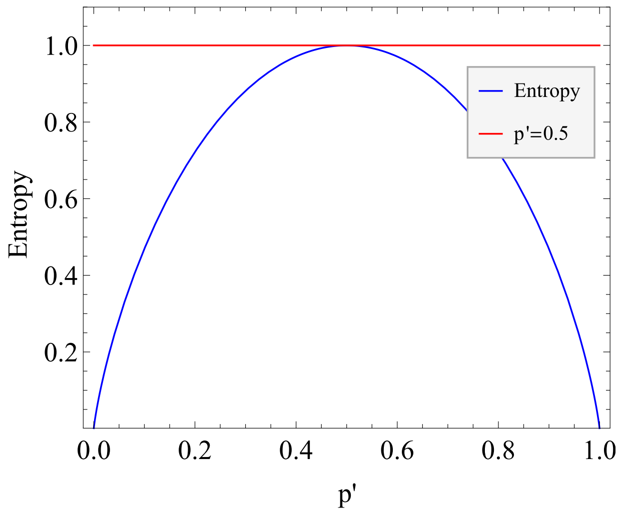


Fig. 1. Information entropy of a BF.

Lemma 1. *Given a BF, suppose n and k keep unchanged, when m becomes larger, the FP probability gets smaller.*

Proof. When n and k are fixed, for larger m, the probability of each bit being 1 in the BF becomes smaller, i.e., p' becomes smaller; thus, the FP probability gets smaller. □

Definition 1. Bloom Filter variable. *The false positive rate of Bloom Filters is determined by m, n, and k. For different n elements, the m-bit string varies. Thus, when the values of m, n, k are given, the m-bit string is a* random variable *(similar but slightly different from the mathematical definition of random variable).*[1] *When the n elements are given, the m-bit string is a random variable instance. Therefore, when the values of m, n, k are given, we call it a Bloom Filter Variable (BFR). Since it is a random variable, we can compute its information entropy.*

Definition 2. Equivalent Bloom Filter variables. *Given two Bloom Filters variables v_1 and v_2, for the same n elements, there are a pair of BFR instances. Given a set with n elements, if these pairs of BFR instances always report the same result: true or false for any input element, we say v_1 and v_2 are equivalent.*

Theorem 1. *Given a Bloom Filter* variable *v_1 with parameters m, n, and k, if its information entropy is not at the maximum, there must exist a smaller equivalent Bloom Filter variable v_2 with parameters m', n and k, where $m' < m$.*

Proof. For v_1, the parameters are m, n, k. Suppose its k hash functions are $h_1(\cdot), h_2(\cdot), \ldots, h_k(\cdot)$. Since the assumption is that the information entropy of v_1 is not at the maximum, according to the property of information entropy, v_1 can be compressed *without information loss*. After compression, suppose the new random variable has a length of $m'(m' < m)$, we name it v_2. Note that during the compression, the information value $m * e$ keeps unchanged; thus, the length of the compressed message has a minimum value. Here v_1 and v_2 are two variables consisting of bits, and we can treat them as two integer variables. We use $In(v_1)$ and $In(v_2)$ to represent the integer value of v_1 and v_2. Furthermore, we use $|In(v_1)|$ represents the length of v_1, then we have $|In(v_1)| = m$, $|In(v_2)| = m'$. Because we compress v_1 and get v_2, this can be regarded as a function $g(\cdot)$. In other words, $g(In(v_1)) = In(v_2)$. We can also obtain v_1 by equation $In(v_1) = g^{-1}(In(v_2))$.

At this stage, we consider the new Bloom Filter variable v_2, the parameters are m', n, and k. Note that we use k *different* hash functions, and the k hash functions are

$$\begin{aligned}
&g^{-1}(In(v_2)) \ll h_1(y) \gg (|g^{-1}(In(v_2))| - h_1(y) - 1),\\
&g^{-1}(In(v_2)) \ll h_2(y) \gg (|g^{-1}(In(v_2))| - h_2(y) - 1),\\
&\ldots\\
&g^{-1}(In(v_2)) \ll h_k(y) \gg (|g^{-1}(In(v_2))| - h_k(y) - 1)
\end{aligned} \tag{12}$$

where '$\ll$' represents the left shift operator, and '$\gg$' represents the right shift operator.[2]

Given an input element y, we can compute the above k values only using v_2 and $h_i(\cdot)$ without v_1. Then we need to prove that for any incoming element y, v_2 reports the same k-bit value. With formula 12, we use equation $v_1 = g^{-1}(In(v_2))$ and $m = |g^{-1}(In(v_2))|$. These k hash functions are simplified as

$$\begin{aligned}
&v_1 \ll h_1(y) \gg (m - h_1(y) - 1),\\
&v_1 \ll h_2(y) \gg (m - h_2(y) - 1),\\
&\ldots\\
&v_1 \ll h_k(y) \gg (m - h_k(y) - 1)
\end{aligned} \tag{13}$$

Here $1 \leqslant i \leqslant k$ and $0 \leqslant h_i(y) \leqslant m-1$. Here $v_1 \ll h_i(y) \gg (m - h_i(y) - 1)$ actually means the value of $h_i(y)$th bit of v_1. This is the same as the k hash functions as v_1. Therefore, v_1 and v_2 are equivalent. □

Theorem 2. *Given a Bloom Filter variable,* if and only if *its information entropy is at the maximum, its FP probability is at the minimum.*

Proof. First, we prove that if the FP probability is at the minimum, then its average information entropy Eq. (11) must be at the maximum. Given a Bloom Filter variable v_1 with parameters m, n, k. Since the assumption is that the average information entropy of v_1 is not at the maximum, according to Theorem I, there exists a smaller Bloom Filter variable v_2 with parameters m', n, k. Since v_2 and v_1 are equivalent, their FP probabilities are the same, we name it f. At this stage, we enlarge the size of v_2 a little from m' to m'', where $m' < m'' < m$. According to Lemma 1, we know the FP probability of v_2 becomes smaller than f. This means that for v_1, there exists a BF variable with a smaller size and a smaller FP probability. Therefore, the FP probability of v_1 is not at the minimum. This means that if its

1. Although we still use the phrase "random variable" for convenience, we want to point out that it is indeed a non-standard usage of the term "random variable." We want to say that m-bit strong is indeed a measurable function, mapping from the total sample space to $\mathbb{R}^m$. Random variable in mathematics is a term whose mapping value is restricted to $\mathbb{C}$.

2. The symbols '$\ll$' and '$\gg$' shift all the binary bits of a number to the left/right by several bits, respectively. On the premise that the number does not overflow, shifting left by x bits is equivalent to multiplying by 2 to the power of x, and shifting right by x bits is equivalent to dividing by 2 to the power of x. For example, shifting the integer 6 ('0110' in binary) to the left by 1 bit is equivalent to the integer 12 ('1100' in binary), and shifting the integer 6 to the right by 1 bit is equivalent to the integer 3 ('0011' in binary).

average information entropy is not at the maximum, then the FP probability is definitely not at the minimum. The contrapositive is that if the FP probability is at the minimum, then its average information entropy must be at the maximum.

Second, we prove that if its average information entropy is at the maximum, the FP probability is at the minimum. Given a Bloom Filter variable v_1 with parameters m, n, k. Since the assumption is that the FP probability of v_1 is not at the maximum, there exists an optimal Bloom Filter variable v_0 with parameters m, n, k', where $k' \neq k$, the average information entropy of v_0 is at the maximum. According to Eq. 11 and Fig. 1, the p' of v_0 is 0.5 whereas the p' of v_1 is not because they have different value of k. Therefore, the average information entropy of BF_1 is not at the maximum. This means if the FP probability is not at the minimum, its average information entropy must be not at the maximum. The contrapositive is that if its average information entropy is at the maximum, the FP probability is at the minimum. □

3.4 Computing the Optimal k

According to Theorem 2, when the average information entropy of the Bloom Filter variable is at the maximum, the FP probability is at the minimum. Recalling the definition of p' in Eq. (1), one can use this interpretation to find k^*, i.e., the optimal number of hash functions. From Fig. 1, we know that when p' is 0.5, E reaches the maximum value 1. By setting the value of p' to 0.5, we have

$$p' = \left(1 - \frac{1}{m}\right)^{k^* n} = 0.5 \tag{14}$$

Further, we have:

$$k^* = -\frac{\ln 2}{n} / \ln\left(1 - \frac{1}{m}\right) \tag{15}$$

This formula is very close to the formula of k^* obtained by Bloom. When x is very small, $\ln(1+x) \approx x$, and therefore $-1/\ln(1-\frac{1}{m}) \approx m$, resulting the same term as in Eq. (9).

Theorem 3. *Given any BF variable, when m and n are fixed, the FP probability f is a function of k, we represent it $f(k)$. Then $f(k)$ is a well-defined function which has only one minimum value.*

Proof. Given a Bloom Filter variable v_1 with parameters m, n, k_1, its entropy is E_1. Given another Bloom Filter variable v_2 with parameters m, n, k_2, its entropy is E_2. (1) For any $k_1 < k_2 \leqslant k^*$, according to Eqs. (1) and (11) and Fig. 1, we know $p'_1 < p'_2 \leqslant 0.5$. We compress v_1 to v_3 with parameters m_3, n, k_1. To make v_3's entropy equal to E_2, m_3 should be mE_1/E_2. In this case, the entropy of v_3 is equal to that of v_2. When the entropy of BF variables is less than 0.5, the same entropy leads to the same p'. In other words, $p'_3 = p'_2$. Because v_2 has more hash functions ($k_2 > k_1$), the FP probability of v_2 is smaller than that of v_3. While v_3 and v_1 have the same FP probability, therefore, the FP probability of v_2 is smaller than that of v_1. In other words, for any k ($k < k^*$) increasing, the FP probability of BFs decreases. (2) For any $k^* \leqslant k_1 < k_2$, according to Eqs. (1), (11), and Fig. 1, we know $p'_1 > p'_2 \geqslant 0.5$. Using the similar derivation, we can derive that the FP probability of v_2 is larger than that of v_1. According to the above two cases, we know that given any BF variable, when m and n are fixed, the FP probability f is a function of k, we represent it $f(k)$. So $f(k)$ is a *well-defined function*, which has a unique minimum value. □

4 Asymptotic Form of the FP Probability

In this section, we derive a new approach to computing the asymptotic form for the FP probability of BFs. The new derivation is based on *partitioned Bloom Filters* (pBF) that are used frequently to carry out parallel queries. Its underpinning principle is simple: the BF is divided into k even partitions, and each hash function only acts on one of the partitions, respectively. The probability that one bit of the BF array remains 0 after inserting n elements in the BF becomes the following as now each hash maps into $\frac{m}{k}$ separate bits.

$$p'_{partition} = \left(1 - \frac{k}{m}\right)^n \tag{16}$$

It is intuitive that the FP probability of partitioned BF is a little bigger than that of BF. Unfortunately, there is no strict proof. Here we show one proof method, which is based on the following Lemma.

Lemma 2. *For $m > 1,\ k > 1,\ n > 1, m > k$,*

$$\left(1 - \frac{k}{m}\right)^n < \left(1 - \frac{1}{m}\right)^{kn} \tag{17}$$

Proof. Directly application of Bernoulli inequality. □

The above lemma shows that $p'_{partition} < p'_{true}$ or equivalently $1 - p'_{partition} > 1 - p'_{true}$. Thus, we know that with the same parameters, the FP probability of the pBF will be larger than that of the standard BF f_{true}, i.e., $f_{partition} > f_{true}$. In addition, Bose's bounds in Eq. (5) state that the precise value of FP probability for a BF f_{true} is larger than f_{bloom}, i.e., $f_{true} > f_{bloom}$. Therefore, we have the following upper and lower bounds:

$$f_{partition} > f_{true} > f_{bloom} \tag{18}$$

For a partitioned BF, the probability that one bit of the array is still 0 p' is shown in Eq. (16). Different from standard Bloom Filters, for a partitioned Bloom Filter, the event $E(h_1 = 1), E(h_2 = 1), E(h_3 = 1), ..., E(h_{i-1} = 1)$ is independent of the event $E(h_{i-1} = 1)$, where $E(h_{i-1} = 1)$ means that the event that the position of $h_{i-1}(x)$ is 1 because each hash function is responsible for one partition, and has no impact on each other. Therefore, we have

$$f_{partition} = (1 - p'_{partition})^k = \left(1 - \left(1 - \frac{k}{m}\right)^n\right)^k \tag{19}$$

Then, the Eq. (18) becomes

$$\left(1 - \left(1 - \frac{k}{m}\right)^n\right)^k > f_{true} > \left(1 - \left(1 - \frac{1}{m}\right)^{nk}\right)^k \tag{20}$$

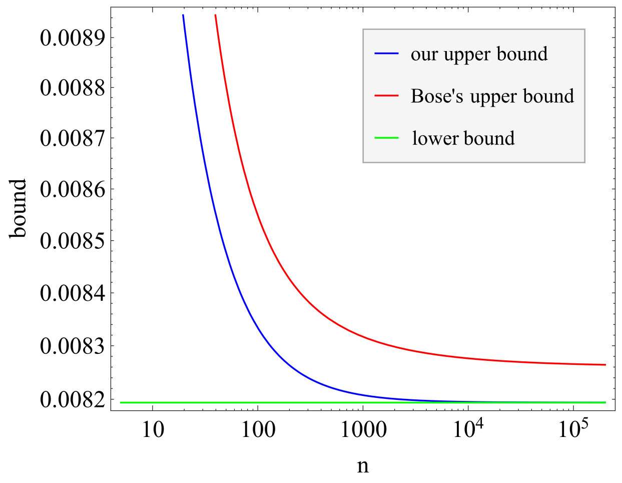


Fig. 2. Upper and lower bound for f_{true} for $k = 7$ and $m = 10n$.

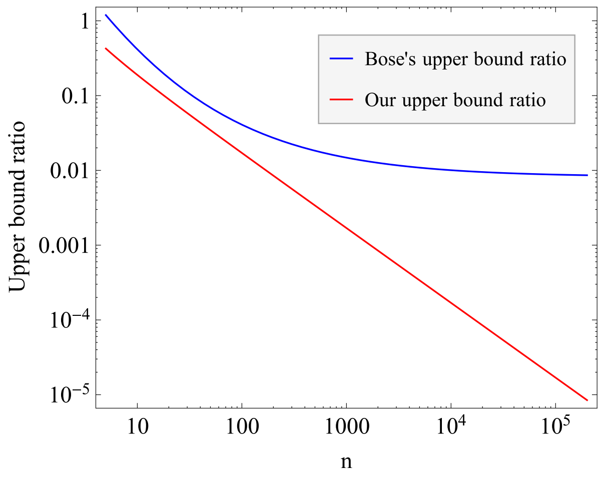


Fig. 3. Bounds error ratio for Bose's bound and the bound derived in this paper for $k = 7$ and $m = 10n$.

Then, we use the well known limit formula:

$$\lim_{x\to\infty}\left(1-\frac{1}{x}\right)^{-x}=e \tag{21}$$

Asymptotically, when m becomes large, we already know that f_{bloom} converges to the term in Eq. (2). Nevertheless, the upper bound has also an asymptotic behaviour as the following, which is the same term as the lower bound limit.

$$\lim_{m\to\infty}\left(1-\left(1-\frac{1}{m}\right)^{nk}\right)^{k}=\left(1-e^{-nk/m}\right)^{k} \tag{22}$$

Through the Sandwich Theorem (also known as squeeze theorem) we obtain the following equation, which is similarly to Christensen and Bose:

$$\lim_{m\to\infty} f_{true}=\left(1-e^{-nk/m}\right)^{k} \tag{23}$$

This means that when m is large, the Bloom's formula can be used with negligible error. However, we still need to evaluate what means m *being large*. We will do this by comparing the two bounds we have in hand: the one from Bose and the one we derived in this paper. We show in Fig. 2 that the two upper bounds along with the lower bound obtained for $k = 7$ and $m = 10n$ as a function of n, the number of elements inserted in the BF. As can be seen, the upper bound derived in this paper and the lower bound f_{bloom} converge relatively fast for $n = 9$, while the upper bound derived by Bose has a much slower convergence. We can see this better by looking at the behavior of the bounds error ratio β, defined as $\beta = \frac{upper\ bound - lower\ bound}{lower\ bound}$, for the two bounds in Fig. 3.

As can be seen, the gap between our derived upper bound and f_{bloom} is decreasing polynomially at a constant speed, while Bose's bound has a lower speed of convergence. In order to extend this observation, we show in Fig. 4 the evolution of the bounds error ratio for a BF with $m = 10000$, $n = 1000$ and varying k.

As expected, error involved with using f_{bloom} increases with the number of hash functions k increases. However, it can be seen that the convergence behavior of the bounds derived in this paper is much better than the one obtained by Bose.

5 Correct Rate of Counting Bloom Filters

The Counting Bloom Filter (CBF) [3], one widely used variant of standard Bloom Filter, replaces each bit with one counter, supporting estimating the frequency of each element in a multiset. Specifically, given a multiset $\mathcal{S}$ of n distinct elements with their corresponding frequencies, we create a counter array A of length m as follows. First, we initialize each counter of A to 0, then for each element $x \in \mathcal{S}$, we use k hash functions to compute k hash values: $h_1(x), h_2(x), \ldots, h_k(x)$ where each hash value is in the range $[1, m]$. Second, for each $1 \leqslant i \leqslant k$, we let $A[h_i(x)] = A[h_i(x)] + 1$. Let f_x be the frequency of element x in multiset $\mathcal{S}$. Therefore, the step of $A[h_i(x)] = A[h_i(x)] + 1 \text{for each} 1 \leqslant i \leqslant k$ will occur f_x times. The resulting counter array A is called the CBF for multiset $\mathcal{S}$. To query the frequency of an element y in multiset $\mathcal{S}$, we first use the same k hash functions to compute k hash values: $h_1(y), h_2(y), \ldots, h_k(y)$. Second, we report the minimum value of the k counters: $A[h_1(x)], A[h_2(x)], \ldots, A[h_k(x)]$ as the estimated frequency of this element. Obviously, the estimated frequency reported by the CBF is always larger than or equal to the real frequency for any element in multiset $\mathcal{S}$. The case that the estimated frequency from the CBF is equal to the real frequency for one element is called the correct case. The probability of such case happening is called the correct rate of the CBF ($\mathcal{C}_r$).

The calculation of the correct rate of CBFs can benefit from our derivation of the false positive probability of standard Bloom Filter. In querying an element x, the correct case happens when there exists at least one hashed counter (among $A[h_1(x)], A[h_2(x)], \ldots, A[h_k(x)]$) that is not hashed by any elements in multiset $\mathcal{S} \setminus \{x \cdot f_x\}$. The contrapositive is that the correct case does not happen when all the k hashed counters are also hashed by some elements in multiset $\mathcal{S} \setminus \{x \cdot f_x\}$. Consider an arbitrary counter $A[b]$ in A. For any distinct element in $\mathcal{S} \setminus \{x \cdot f_x\}$ and any hash function h_i ($1 \leqslant i \leqslant k$), the probability that this element is not hashed to counter $A[b]$ by h_i is $1 - 1/m$. As $\mathcal{S} \setminus \{x \cdot f_x\}$ has $n - 1$ distinct elements and each distinct element is hashed k times, the probability that $A[b]$ is not hashed by any element in multiset $\mathcal{S} \setminus \{x \cdot f_x\}$ is p_c:

$$p_c = \left(1-\frac{1}{m}\right)^{k(n-1)} \tag{24}$$

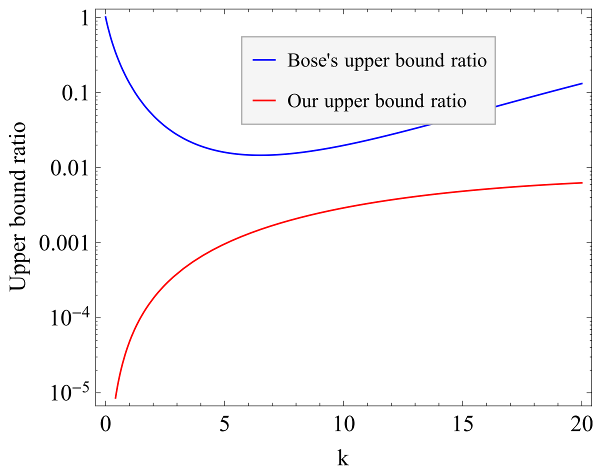


Fig. 4. Bounds error ratio for Bose's bound and the bound derived in this paper for $m = 10000$, $n = 1000$ and varying k.

Thus, the probability that $A[b]$ is hashed by some elements in multiset $\mathcal{S} \setminus \{x \cdot f_x\}$ is $1-(1-1/m)^{k(n-1)}$. The probability that all the k hashed counters are also hashed by some elements in multiset $\mathcal{S} \setminus \{x \cdot f_x\}$ is answered by f_{true} with element number of $n-1$. We denote $f_{true}|_{n-1}$ as f_{true} with element number of $n-1$, and get:

$$1 - \mathcal{C}_r = f_{true}|_{n-1} \Rightarrow \mathcal{C}_r = 1 - f_{true}|_{n-1} \tag{25}$$

Applying Eq. (20), we can get the upper and lower bounds of the $\mathcal{C}_r$ of CBFs:

$$1 - \left(1 - \left(1 - \frac{k}{m}\right)^{n-1}\right)^k < \mathcal{C}_r < 1 - \left(1 - \left(1 - \frac{1}{m}\right)^{(n-1)k}\right)^k \tag{26}$$

6 Experimental Results

In this section, we first validate our proposed formula of optimal k Eq. (15). Second, we compare our proposed upper bound of the FP probability of BFs Eq. (20) with Bose's upper bound Eq. (5). Third, we validate our proposed upper and lower bounds of the correct rate of CBFs Eq. (26).

6.1 Experimental Setup

Datasets: We use four kinds of datasets: 1) IP Trace Dataset; 2) Data Center Dataset; 3) Network Dataset; 4) Synthetic Dataset. The details are as follows.

1) IP Trace Dataset: The IP Trace Dataset contains anonymized IP traces collected in 2016 by CAIDA [56]. Each item contains a source IP address (sip, 4 bytes) and a destination IP address (dip, 4 bytes), 8 bytes in total. We treat each sip-dip pair as an ID.

2) Data Center Dataset: The Data center dataset [57] contains traces collected from the data centers in [58]. Each item (4 bytes) represents the ID of the trace.

3) Network Dataset: The network dataset contains users' posting history on the stack exchange website [59]. Each item (4 bytes) represents the ID of each user.

4) Synthetic Dataset: We generate a synthetic dataset that follows the Zipf [60] distribution using Web Polygraph [61], an open-source performance testing tool. The length of each item ID is 4 bytes, and the skewness of the dataset is 1.5.

The operations on the above datasets: For BF, we regard each distinct ID as an element in the aforementioned set. We use (part of) the first *100/1.5/0.17/0.23*M[3] distinct IDs to construct BFs, and query the next *300/4.5/0.51/0.69*M distinct IDs to get the empirical FP probabilities of these BFs. For CBF, we regard each distinct ID and its occurrence number as a distinct element and the corresponding frequency in the aforementioned multiset, respectively. We use (part of) the first *100/1.5/0.17/0.23*M distinct IDs and their occurrence numbers in the current trace to construct CBFs, and query their frequencies to get the empirical correct rates of these CBFs.

Implementation: We have implemented the standard Bloom filter in C++. We use the Bob Hash (obtained from the open source website [62]) with different initial seeds to implement the hash functions in BFs as recommended by literature [63]. All the implementation source code is made publicly available at GitHub[4].

6.2 Optimal k Formula Validation

6.2.1 Optimal k versus n

Figs. 5a, 5b, 5c, and 5d plot the empirically and theoretically optimal k with different n for m = 500M, 7.5M, 0.85M, 1.15M. *Our results show that the optimal k calculated from our new formula follows the empirically optimal k very well, regardless of the values of n.* We observe that the optimal k calculated from our new formula is very close to the one calculated from the formula obtained by Bloom.

6.2.2 Optimal k versus m

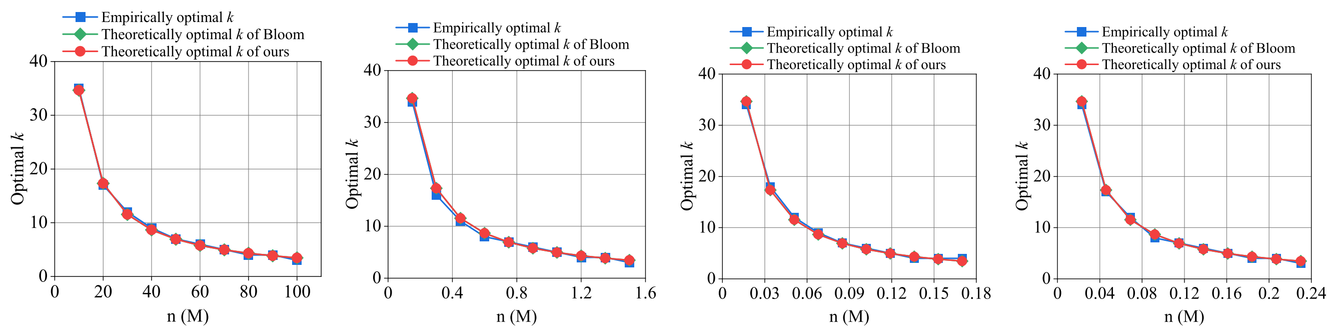
Fig. 6a, 6b, 6c, and 6d plot the empirically and theoretically optimal k with different m for n = 50M, 0.75M, 0.085M, 0.115M. *Our results show that the optimal k calculated from our new formula follows the empirically optimal k very well, regardless of the values of m.* We observe that the optimal k calculated from our new formula is very close to the one calculated from the formula obtained by Bloom, especially when m becomes larger.

6.3 Upper Bound Comparison

6.3.1 Upper Bound versus n

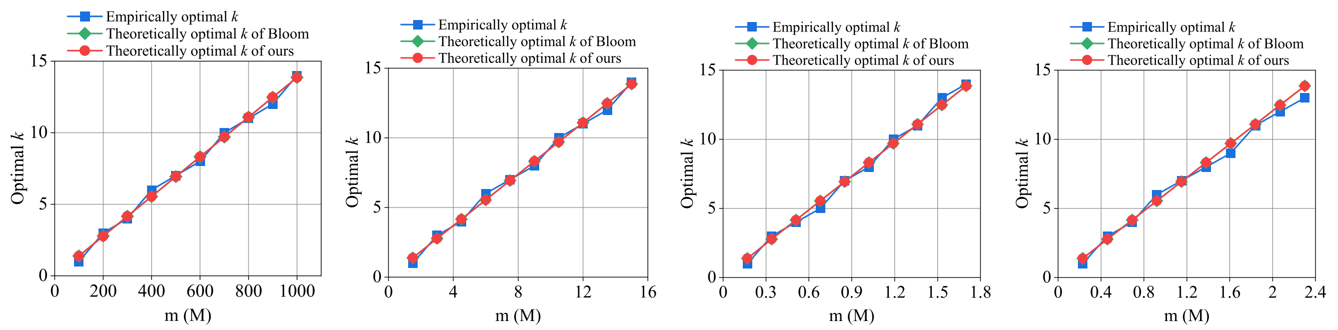
Fig. 7a, 7b, 7c, and 7d plot the empirical results, Bloom's theoretical results, Bose's upper bounds, and our upper bounds of FP probability with different n for m = 500M, 7.5M, 0.85M, 1.15M and $k = 6$. *Our results show that our upper bounds of FP probability follows the empirical FP probability very well, regardless of the values of n.* We find that all above four results almost coincide with each other, which demonstrates the tightness of bounds in Eqs. (5) and (20). To compare upper bound of Bose and ours more intuitively, in Figs. 8a, 8b, 8c, and 8d, we plot the bounds error ratios β, defined as $\beta = \frac{upper\ bound - lower\ bound}{lower\ bound}$, of these two upper bounds with different n for m = 500M, 7.5M, 0.85M, 1.15M and $k = 6$. *Our results show that the bounds error ratio of our upper bound is 23070.7 times lower than that of Bose's upper bound on average.* We find that our upper bound almost coincides with the lower bound, which demonstrates the superiority of our upper bound.

3. They correspond to the 4 datasets mentioned above, respectively.
4. https://github.com/pkufzc/Bloom-Error-TKDE



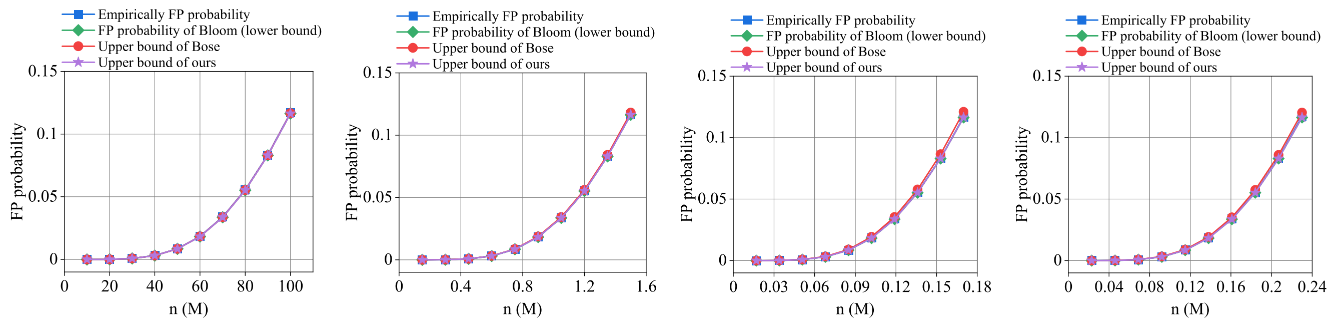
(a) IP trace (b) Data Center (c) Network (d) Synthetic

Fig. 5. Optimal k versus n for m = 500M, 7.5M, 0.85M, 1.15M.



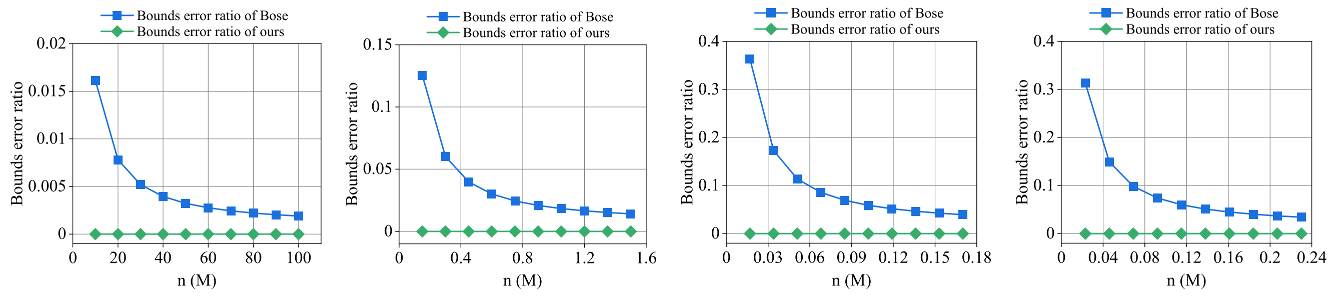
(a) IP trace (b) Data Center (c) Network (d) Synthetic

Fig. 6. Optimal k versus m for n = 50M, 0.75M, 0.085M, 0.115M.



(a) IP trace (b) Data Center (c) Network (d) Synthetic

Fig. 7. FP probability versus n for m = 500M, 7.5M, 0.85M, 1.15M and $k = 6$.



(a) IP trace (b) Data Center (c) Network (d) Synthetic

Fig. 8. Bounds error ratio versus n for m = 500M, 7.5M, 0.85M, 1.15M and $k = 6$.

6.3.2 *Upper Bound versus* m

Figs. 9a, 9b, 9c, and 9d plot the empirical results, Bloom's theoretical results, Bose's upper bounds, and our upper bounds of FP probability with different m for n = 50M, 0.75M, 0.085M, 0.115M and $k = 6$. *Our results show that our upper bounds of FP probability follows the empirical FP probability very well, regardless of the values of* m. Fig. 10a, 10b, 10c, and 10d plot the bounds error ratios of these two upper bounds with different m for n = 50M, 0.75M, 0.085M, 0.115M and $k = 6$. *Our results show that the bounds error ratio of our upper bound is 18838.9 times lower*

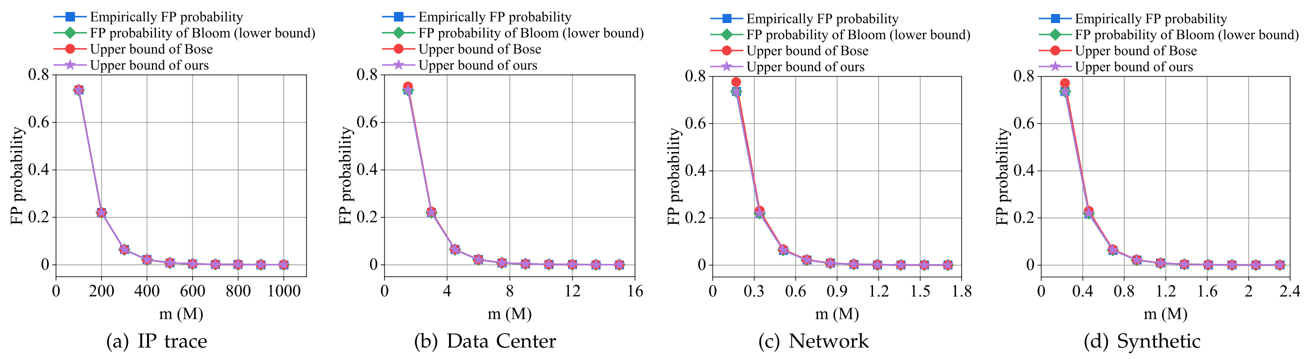


Fig. 9. FP probability versus m for n = 50M, 0.75M, 0.085M, 0.115M and $k = 6$.

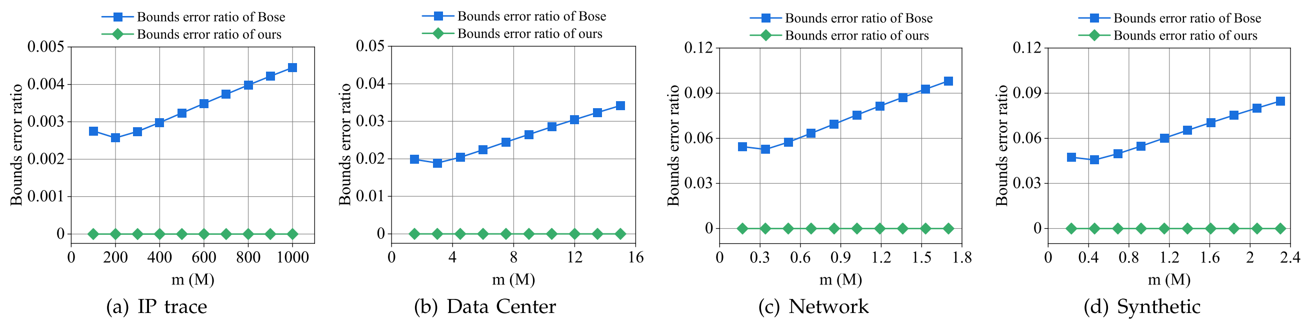


Fig. 10. Bounds error ratio versus m for n = 50M, 0.75M, 0.085M, 0.115M and $k = 6$.

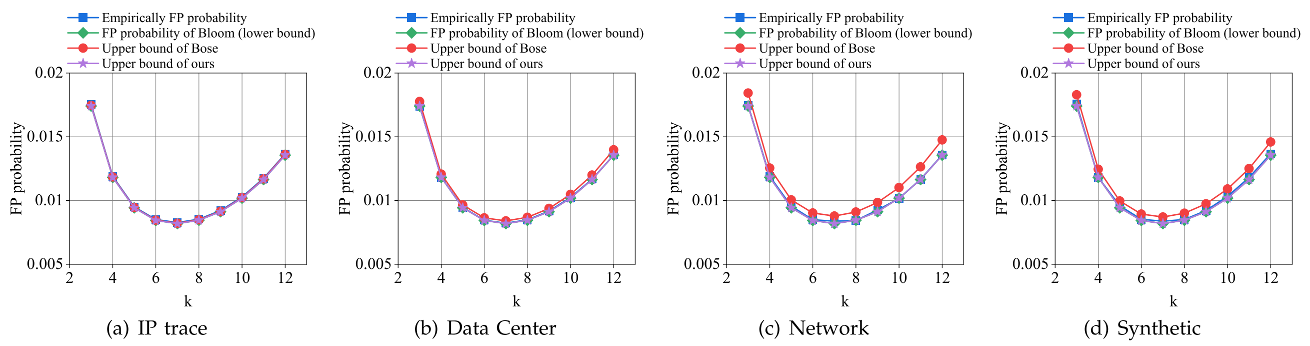


Fig. 11. FP probability versus k for n = 50M, 0.75M, 0.085M, 0.115M and m = 500M, 7.5M, 0.85M, 1.15M.

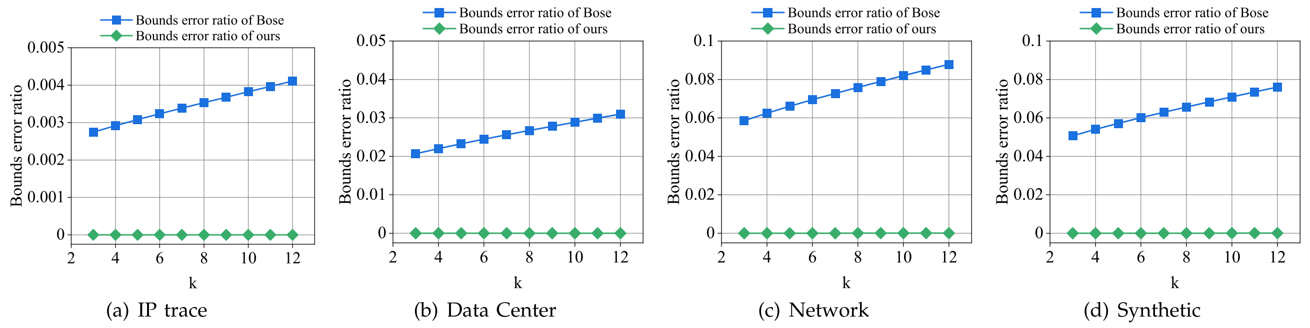
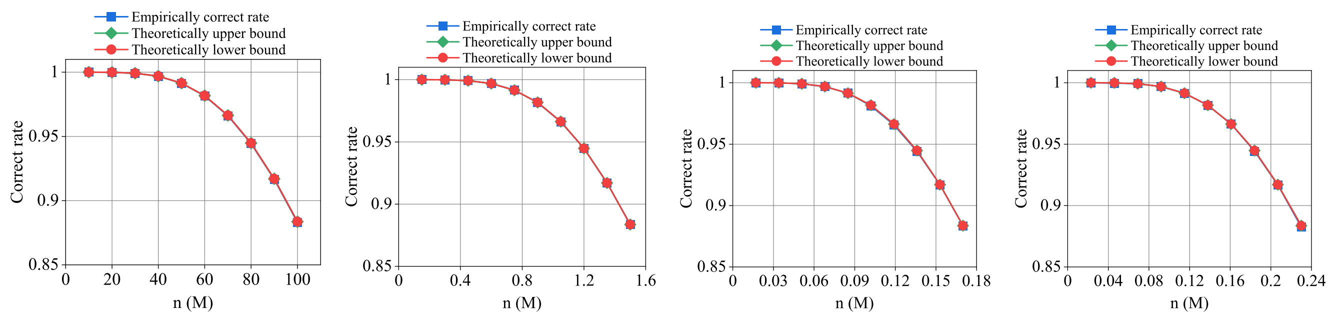


Fig. 12. Bounds error ratio versus k for n = 50M, 0.75M, 0.085M, 0.115M and m = 500M, 7.5M, 0.85M, 1.15M.

than that of Bose's upper bound on average. We find that our upper bound almost coincides with the lower bound, regardless of the values of m.

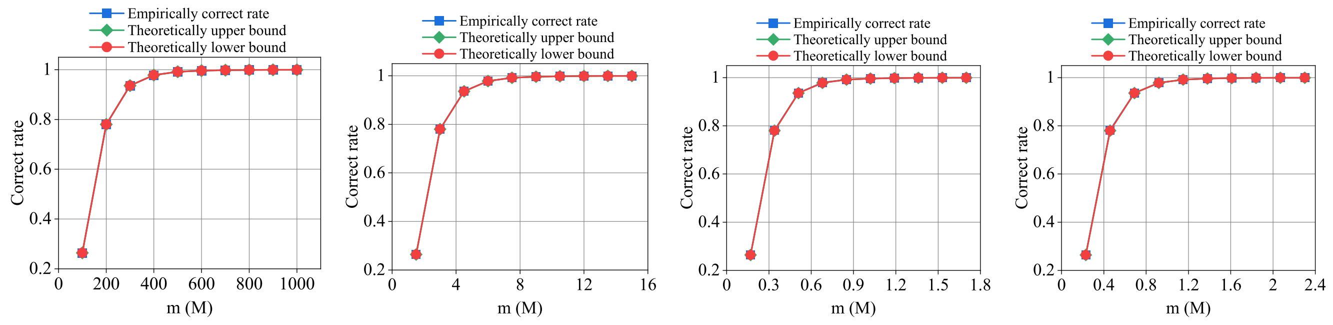
6.3.3 *Upper Bound versus k*

Figs. 11a, 11b, 11c, and 11d plot the empirical results, Bloom's theoretical results, Bose's upper bounds, and our upper bounds of FP probability with different k for n = 50M, 0.75M, 0.085M, 0.115M and m = 500M, 7.5M, 0.85M, 1.15M. *Our results show that our upper bounds of FP probability follows the empirical FP probability very well, regardless of the values of k.* Figs. 12a, 12b, 12c, and 12d plot the bounds error ratios of these two upper bounds with different k for n = 50M, 0.75M, 0.085M, 0.115M and m = 500M, 7.5M, 0.85M, 1.15M. *Our results show that the bounds error ratio of our upper*



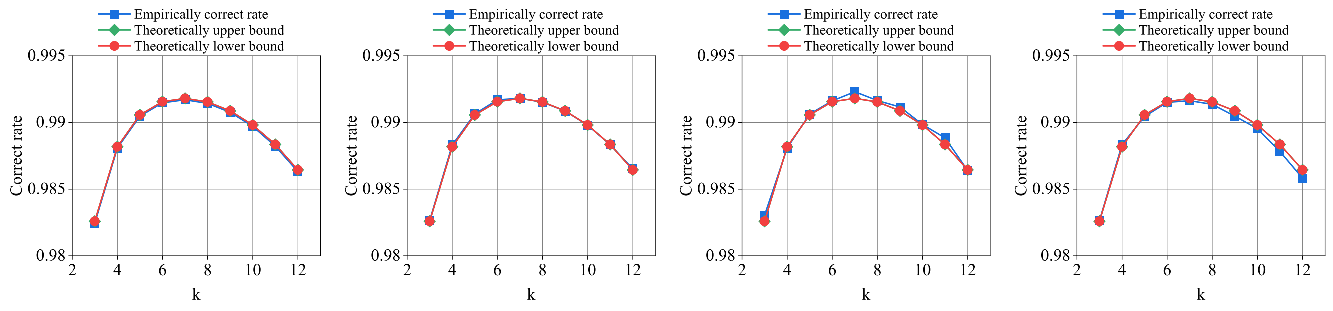
(a) IP trace (b) Data Center (c) Network (d) Synthetic

Fig. 13. Correct rate versus n for m = 500M, 7.5M, 0.85M, 1.15M and $k = 6$.



(a) IP trace (b) Data Center (c) Network (d) Synthetic

Fig. 14. Correct rate versus m for n = 50M, 0.75M, 0.085M, 0.115M and $k = 6$.



(a) IP trace (b) Data Center (c) Network (d) Synthetic

Fig. 15. Correct rate versus k for n = 50M, 0.75M, 0.085M, 0.115M and m = 500M, 7.5M, 0.85M, 1.15M.

bound is 14571.0 times lower than that of Bose's upper bound on average.

6.4 $\mathcal{C}_r$ Formula Validation

Figs. 13a, 13b, 13c, and 13d, Figs. 14a, 14b, 14c, and 14d, and Figs. 15a, 15b, 15c, and 15d plot the correct rates of CBFs with different values of n, m, and k, respectively. *Our results show that our lower and upper bounds of the correct rate of CBFs follow the empirical correct rates very well, regardless of the values of n, m, and k.*

7 Conclusion

In this paper, we discuss the evolutionary of the formula of Bloom Filter. Three formulas of false positive probability are presented: Bloom's formula, Bose's formula, and Christensen's formula. There is an error in the deduction process of Bloom's formula, and a minor error in Bose's formula. Christensen's formula is correct, but the false positive must be caculated using an iterative table-based algorithm with a time complexity of $O(knm)$. What is worse, it cannot deduce the optimal value of k. To compute the optimal value of k, we use information and entropy theory to deduce the exact formula of k. To compute the false positive probability, 1) For small m, Christensen's formula can be used; 2) For large m, we propose a new upper bound which is much more accurate than state-of-the-art. Fortunately, when m is infinitely large, our upper bound becomes the same as the lower bound, which is Bloom's formula. Besides, we derive the bounds of correct rate of Counting Bloom Filters through our proposed formulas about Bloom Filters. All the implementation source code is made publicly available at GitHub [51].

Acknowledgments

We would like to thank Siyuan Dong for his participation in the experiments, and Yuhan Wu and Ruwen Zhang for their participation in the discussions. We would like to thank the anonymous reviewers for their valuable suggestions and comments.

REFERENCES

[1] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Commun. ACM*, vol. 13, no. 7, pp. 422–426, 1970.

[2] M. Mitzenmacher, P. Reviriego, and S. Pontarelli, "OMASS: One memory access set separation," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 7, pp. 1940–1943, Jul. 2016.

[3] L. Fan, P. Cao, J. Almeida, and A. Z. Broder, "Summary cache: A scalable wide-area web cache sharing protocol," *IEEE ACM Trans. Netw.*, vol. 8, no. 3, pp. 281–293, Jun. 2000.

[4] M. Mitzenmacher, "Distributed, compressed bloom filter web cache server," US Patent 6,920,477, 2005.

[5] M. Yoon, "Aging bloom filter with two active buffers for dynamic sets," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 1, pp. 134–138, Jan. 2010.

[6] F. Ye, H. Luo, S. Lu, and L. Zhang, "Statistical en-route filtering of injected false data in sensor networks," *IEEE J. Sel. Areas Commun.*, vol. 23, no. 4, pp. 839–850, Apr. 2006.

[7] M. Luk, G. Mezzour, A. Perrig, and V. Gligor, "MiniSec: A secure sensor network communication architecture," in *Proc. IEEE ACM 6th Int. Symp. Inf. Proces. Sensor Netw.*, 2007, pp. 479–488.

[8] T. Chen, D. Guo, Y. He, H. Chen, X. Liu, and X. Luo, "A bloom filters based dissemination protocol in wireless sensor networks," *Ad Hoc Netw.*, vol. 11, no. 4, pp. 1359–1371, 2013.

[9] M. Yu, A. Fabrikant, and J. Rexford, "Buffalo: Bloom filter forwarding architecture for large organizations," in *Proc. ACM Conf. Emerg. Netw. Exp. Technol.*, 2009, pp. 313–324.

[10] D. Li, H. Cui, Y. Hu, Y. Xia, and X. Wang, "Scalable data center multicast using multi-class bloom filter," in *Proc. IEEE Int. Conf. Netw. Protoc.*, 2011, pp. 266–275.

[11] D. Li, Y. Li, J. Wu, S. Su, and J. Yu, "ESM: Efficient and scalable data center multicast routing," *IEEE ACM Trans. Netw.*, vol. 20, no. 3, pp. 944–955, Jun. 2012.

[12] A. Papadopoulos and D. Katsaros, "A-tree: Distributed indexing of multidimensional data for cloud computing environments," in *Proc. IEEE Int. Conf. Cloud Comput. Technol. Sci.*, 2011, pp. 407–414.

[13] V. Roussev, L. Wang, G. Richard, and L. Marziale, "A cloud computing platform for large-scale forensic computing," in *Proc. IFIP Adv. Inf. Commun. Technol.*, 2009, pp. 201–214.

[14] S. Xiong et al., "kBF: Towards approximate and bloom filter based key-value storage for cloud computing systems," *IEEE Trans. Cloud Comput.*, vol. 5, no. 1, pp. 85–98, Jan.–Mar. 2014.

[15] S. Dharmapurikar, P. Krishnamurthy, and D. E. Taylor, "Longest prefix matching using bloom filters," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 33, no. 4, pp. 201–212, 2003.

[16] F. Bonomi, M. Mitzenmacher, R. Panigrah, S. Singh, and G. Varghese, "Beyond bloom filters: From approximate membership checks to approximate state machines," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 36, no. 4, pp. 315–326, 2006.

[17] Y. Wang et al., "NameFilter: Achieving fast name lookup with low memory cost via applying two-stage bloom filters," in *Proc. IEEE Conf. Comput. Commun.*, 2013, pp. 95–99.

[18] B. Debnath, S. Sengupta, and J. Li, "Flashstore: High throughput persistent key-value store," *Proc. VLDB Endow.*, vol. 3, no. 1-2, pp. 1414–1425, 2010.

[19] Rocksdb - a persistent key-value store for fast storage environments. [Online]. Available: http://rocksdb.org/

[20] Y. Li, C. Tian, F. Guo, C. Li, and Y. Xu, "ElasticBF: Elastic bloom filter with hotness awareness for boosting read performance in large key-value stores," in *Proc. USENIX Annu. Tech. Conf.*, 2019, pp. 739–752.

[21] N. Dayan, M. Athanassoulis, and S. Idreos, "Optimal bloom filters and adaptive merging for LSM-trees," *ACM Trans. Database Syst.*, vol. 43, no. 4, pp. 1–48, 2018.

[22] S. Luo, S. Chatterjee, R. Ketsetsidis, N. Dayan, W. Qin, and S. Idreos, "Rosetta: A robust space-time optimized range filter for key-value stores," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2020, pp. 2071–2086.

[23] Y. Chai, Y. Chai, X. Wang, H. Wei, and Y. Wang, "Adaptive lower-level driven compaction to optimize LSM-tree key-value stores," *IEEE Trans. Knowl. Data Eng.*, vol. 34, no. 6, pp. 2595–2609, Jun. 2022.

[24] R. Schnell, T. Bachteler, and J. Reiher, "Privacy-preserving record linkage using bloom filters," *BMC Med. Inform. Decis. Mak.*, vol. 9, no. 1, pp. 1–11, 2009.

[25] E. A. Durham, M. Kantarcioglu, Y. Xue, C. Toth, M. Kuzu, and B. Malin, "Composite bloom filters for secure record linkage," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 12, pp. 2956–2968, Dec. 2014.

[26] D. Vatsalan and P. Christen, "Scalable privacy-preserving record linkage for multiple databases," in *Proc. ACM Int. Conf. Inf. Knowl. Manag.*, 2014, pp. 1795–1798.

[27] R. Schnell and C. Borgs, "Randomized response and balanced bloom filters for privacy preserving record linkage," in *Proc. IEEE Int. Conf. Data Min. Workshops*, 2016, pp. 218–224.

[28] P. Christen, R. Schnell, D. Vatsalan, and T. Ranbaduge, "Efficient cryptanalysis of bloom filters for privacy-preserving record linkage," in *Proc. Pacific-Asia Conf. Knowl. Discov. Data Min.*, 2017, pp. 628–640.

[29] P. Christen, T. Ranbaduge, D. Vatsalan, and R. Schnell, "Precise and fast cryptanalysis for bloom filter based privacy-preserving record linkage," *IEEE Trans. Knowl. Data Eng.*, vol. 31, no. 11, pp. 2164–2177, Nov. 2019.

[30] S. Jiang et al., "Privacy-preserving and efficient multi-keyword search over encrypted data on blockchain," in *Proc. IEEE Int. Conf. Blockchain*, 2019, pp. 405–410.

[31] T. Wang, W. Zhu, Q. Ma, Z. Shen, and Z. Shao, "Abacus: Address-partitioned bloom filter on address checking for uniqueness in IoT blockchain," in *Proc. IEEE ACM Int. Conf. Comput. Des. Dig. Tech. Pap.*, 2020, pp. 1–7.

[32] J. Han, M. Song, H. Eom, and Y. Son, "An efficient multi-signature wallet in blockchain using bloom filter," in *Proc. ACM Symp. Appl. Comput.*, 2021, pp. 273–281.

[33] B. Debnath, S. Sengupta, J. Li, D. J. Lilja, and D. H. Du, "Bloomflash: Bloom filter on flash-based storage," in *Proc. IEEE Int. Conf. Distrib. Comput. Syst.*, 2011, pp. 635–644.

[34] O. Papapetrou, E. Ioannou, and D. Skoutas, "Efficient discovery of frequent subgraph patterns in uncertain graph databases," in *Proc. IEEE Adv. Database Technol.*, 2011, pp. 355–366.

[35] H. Lang, T. Mühlbauer, F. Funke, P. A. Boncz, T. Neumann, and A. Kemper, "Data blocks: Hybrid OLTP and OLAP on compressed storage using both vectorization and compilation," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2016, pp. 311–326.

[36] K. Cheng, L. Xiang, and M. Iwaihara, "Time-decaying bloom filters for data streams with skewed distributions," in *Proc. IEEE Int. Workshop Res. Issues Data Eng.*, 2005, pp. 63–69.

[37] F. Deng and D. Rafiei, "Approximately detecting duplicates for streaming data using stable bloom filters," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2006, pp. 25–36.

[38] L. Qiu, Y. Li, and X. Wu, "Preserving privacy in association rule mining with bloom filters," *J. Intell. Inf. Syst.*, vol. 29, no. 3, pp. 253–278, 2007.

[39] Y. Tian, T. Zou, F. Ozcan, R. Goncalves, and H. Pirahesh, "Joins for hybrid warehouses: Exploiting massive parallelism in hadoop and enterprise data warehouses," in *Proc. Int. Conf. Extending Database Technol.*, 2015, pp. 373–384.

[40] H. Dai, M. Shahzad, A. X. Liu, and Y. Zhong, "Finding persistent items in data streams," *Proc. VLDB Endow.*, vol. 10, no. 4, pp. 289–300, 2016.

[41] Y. Peng, J. Guo, F. Li, W. Qian, and A. Zhou, "Persistent bloom filter: Membership testing for the entire history," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2018, pp. 1037–1052.

[42] J. Li et al., "WavingSketch: An unbiased and generic sketch for finding top-K items in data streams," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, 2020, pp. 1574–1584.

[43] M. M. Cisse, N. Usunier, T. Artieres, and P. Gallinari, "Robust bloom filters for large multilabel classification tasks," in *Proc. Adv. Neural Inf. Proces. Syst.*, 2013, pp. 1851–1859.

[44] M. Mitzenmacher, "A model for learned bloom filters and optimizing by sandwiching," in *Proc. Adv. Neural Inf. Proces. Syst.*, 2018, pp. 462–471.

[45] Q. Liu, L. Zheng, Y. Shen, and L. Chen, "Stable learned bloom filters for data streams," *Proc. VLDB Endow.*, vol. 13, no. 12, pp. 2355–2367, 2020.

[46] J. R. Anderson, Q. Huang, W. Krichene, S. Rendle, and L. Zhang, "Superbloom: Bloom filter meets transformer," *CoRR*, 2020. [Online]. Available: https://arxiv.org/abs/2002.04723

[47] R. Patgiri, A. Biswas, and S. Nayak, "deepBF: Malicious URL detection using learned bloom filter and evolutionary deep learning," *CoRR*, 2021. [Online]. Available: https://arxiv.org/abs/2103.12544

[48] D. Guo, Y. Liu, X. Li, and P. Yang, "False negative problem of counting bloom filter," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 5, pp. 651–664, May 2010.

[49] P. Bose et al., "On the false-positive rate of bloom filters," *Inf. Process. Lett.*, vol. 108, no. 4, pp. 210–213, 2008.

[50] K. Christensen, A. Roginsky, and M. Jimeno, "A new analysis of the false positive rate of a bloom filter," *Inf. Process. Lett.*, vol. 110, no. 21, pp. 944–949, 2010.

[51] Our open source website. [Online]. Available: https://github.com/pkufzc/Bloom-Error-TKDE
[52] F. Grandi, "The γ-transform: A new approach to the study of a discrete and finite random variable," *Int. J. Math. Models Appl. Sci*, vol. 9, pp. 624–635, 2015.
[53] F. Grandi, "On the analysis of bloom filters," *Inf. Process. Lett.*, vol. 129, pp. 35–39, 2018.
[54] C. E. Shannon, "A mathematical theory of communication," *Bell Syst. Tech. J.*, vol. 27, no. 3, pp. 379–423, 1948.
[55] L. Brillouin, "Science and information theory," *Dover Pub.*, vol. 2, 2004, Art. no. 293.
[56] The caida anonymized 2016 internet traces, 2016. [Online]. Available: http://www.caida.org/data/overview/
[57] The data center dataset, 2010. [Online]. Available: http://pages.cs.wisc.edu/tbenson/IMC10_Data.html
[58] T. Benson, A. Akella, and D. A. Maltz, "Network traffic characteristics of data centers in the wild," in *Proc. ACM SIGCOMM Internet Meas. Conf.*, 2010, pp. 267–280.
[59] The network dataset internet traces, 2014. [Online]. Available: http://snap.stanford.edu/data/
[60] D. M. Powers, "Applications and explanations of ZIPF's law," in *Proc. J. Conf. New Methods Lang. Process. Comput. Nat. Lang. Learn.*, 1998, pp. 151–160.
[61] A. Rousskov and D. Wessels, "High-performance benchmarking with web polygraph," *Softw. Pract Exper*, vol. 34, no. 2, pp. 187–211, 2004.
[62] Hash website, 1997. [Online]. Available: http://burtleburtle.net/bob/hash/evahash.html
[63] C. Henke, C. Schmoll, and T. Zseby, "Empirical evaluation of hash functions for multipoint measurements," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 3, pp. 39–50, 2008.

Zhuochen Fan is currently working toward the PhD degree with School of Computer Science, Peking University, advised by Tong Yang. His research interests include network big data, network measurements, streaming algorithms, Bloom Filters, and mobile computing.

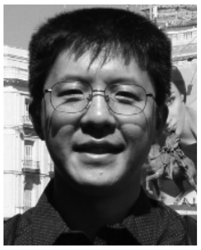
Gang Wen received the BS degree in mathematics from Peking University, in 2022. He is currently working toward the PhD degree majoring in statistics with Yale University. His research interests include network big data, network measurements, streaming algorithms, Bloom Filters, and deep learning theory. He has received the Excellent Academic Student Award and Scholarship for two years from Peking University.

Zhipeng Huang received the MS degree from Peking University, advised by Tong Yang. He is interested in applying machine learning to indexing, computer networks, and data stream processing.

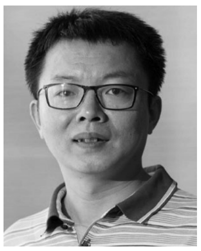
Yang Zhou received the graduated degree from Peking University, advised by Tong Yang. He is currently working toward the PhD degree with Harvard University, advised by Minlan Yu. He is broadly interested in streaming algorithms, networked systems, and data-intensive systems.

Qiaobin Fu received the BE degree from the Dalian University of Technology, in 2011, the MS degree from the University of Chinese Academy of Sciences, in 2014, and the PhD degree from Boston University, in 2020. Currently, he is working with Google on network-aware scheduling and cross-resource optimization. His research interests focus on computer networking, inter/intra-data center networks, and cloud computing.

Tong Yang (Member, IEEE) received the PhD degree in computer science from Tsinghua University, in 2013. He visited the Institute of Computing Technology, Chinese Academy of Sciences (CAS). Currently, he is an associate professor with the School of Computer Science, Peking University. His research interests include network big data, sketches, network measurements, Bloom Filters, IP lookups, KV stores, and hash tables. He published papers in SIGCOMM, NSDI, SIGKDD, SIGMOD, VLDB, ICDE, INFOCOM, USENIX ATC, TKDE, TC, TPDS, ToN, VLDB Journal, JSAC, etc. He is currently an associate editor for Knowledge and Information Systems.



Alex X. Liu (Fellow, IEEE) received the PhD degree in computer science from the University of Texas, Austin, in 2006, and is currently the president of the Software Engineering Institute and the Chief Security Officer of the Midea Group. Before that, he was the Chief Scientist of the Ant Group, and further before that, he was a professor of the Department of Computer Science and Engineering with Michigan State University. He received the IEEE & IFIP William C. Carter Award in 2004, a National Science Foundation CAREER award, in 2009, the Michigan State University Withrow Distinguished Scholar (Junior) Award, in 2011, and the Michigan State University Withrow Distinguished Scholar (Senior) Award, in 2019. He has served as an editor for IEEE/ACM Transactions on Networking and an Area Editor for Computer Communications. He is currently an associate editor for IEEE Transactions on Dependable and Secure Computing and IEEE Transactions on Mobile Computing. He has served as the TPC Co-Chair for ICNP 2014 and IFIP Networking 2019. He received Best Paper Awards from SECON-2018, ICNP-2012, SRDS-2012, and LISA-2010. His research interests focus on cybersecurity, cloud computing, dependable computing, and privacy-preserving computing. He is a Member of Academia Europaea, a Member of the European Academy of Sciences and Art, an IET Fellow, an AAIA Fellow, and an ACM Distinguished Scientist.



Bin Cui (Senior Member, IEEE) received the PhD from the National University of Singapore, in 2004. He is a professor and vice dean in the School of CS, Peking University. His research interests include database system architectures, query and index techniques, big data management, and mining. He is serving as vice chair of Techical Commettee on Database China Computer Federation (CCF) and Trustee Board Member of VLDB Endowment, is also in the Editorial Board of Distributed and Parallel Databases, Journal of Computer Science and Technology, and SCIENCE CHINA Information Sciences, and was an assocaite editor of *IEEE Transactions on Knowledge and Data Engineering (TKDE)* and *VLDB Journal*. He was awarded Microsoft Young professorship award (MSRA 2008), CCF Young Scientist award, (2009), Second Prize of Natural Science Award of MOE China, (2014), and appointed as Cheung Kong distinguished professor by MOE, in 2016.