Less Hashing, Same Performance: Building a Better Bloom Filter[*]

Adam Kirsch[†] Michael Mitzenmacher[‡]

Harvard School of Engineering and Applied Sciences
{kirsch,michaelm}@eecs.harvard.edu

Abstract

A standard technique from the hashing literature is to use two hash functions $h_1(x)$ and $h_2(x)$ to simulate additional hash functions of the form $g_i(x) = h_1(x) + ih_2(x)$. We demonstrate that this technique can be usefully applied to Bloom filters and related data structures. Specifically, only two hash functions are necessary to effectively implement a Bloom filter without any loss in the asymptotic false positive probability. This leads to less computation and potentially less need for randomness in practice.

1 Introduction

A Bloom filter is a simple space-efficient randomized data structure for representing a set in order to support membership queries. Although Bloom filters allow false positives, the space savings often outweigh this drawback. The Bloom filter and its many variations have proven increasingly important for many applications (see, for instance, the survey [4]). As just a partial listing of examples, counting Bloom filters allow deletions as well as insertions of items [13], compressed Bloom filters are optimized to minimize space when transmitted [20], retouched Bloom filters trade off false positives and false negatives [10], Bloomier filters keep function values associated with set elements (thereby offering more than set membership) [5], count-min sketches [6] and multistage filters [12] track counts associated with items, and approximate concurrent state machines track the dynamically changing state of a changing set of items [2]. Although recently more complex but asymptotically better alternatives have been proposed (e.g. [3, 23]), the Bloom filter's simplicity, ease of use, and excellent performance make it a standard data structure that is and will continue to be of great use in many applications. For those who are not familiar with the Bloom filter, we review it below in Section 2. For now, it suffices to know that Bloom filters make use of multiple hash functions.

In this paper, we show that applying a standard technique from the hashing literature can simplify the implementation of Bloom filters significantly. The idea is the following: two hash functions $h_1(x)$ and $h_2(x)$ can simulate more than two hash functions of the form $g_i(x) = h_1(x) + ih_2(x)$. (See, for example, Knuth's discussion of open addressing with double hashing [18].) In our

[*]Preliminary versions of this work appeared in [17] and [16].

[†]Supported in part by an NSF Graduate Research Fellowship, NSF grants CCR-9983832 and CCR-0121154, and a grant from Cisco Systems.

[‡]Supported in part by NSF grants CCR-9983832 and CCR-0121154 and a grant from Cisco Systems.

context i will range from 0 up to some number $k-1$ to give k hash functions, and the hash values are taken modulo the size of the relevant hash table. We demonstrate that this technique can be usefully applied to Bloom filters and related data structures. Specifically, only two hash functions are necessary to effectively implement a Bloom filter without any increase in the asymptotic false positive probability. This leads to less computation and potentially less need for randomness in practice. Specifically, in query-intensive applications where computationally non-trivial hash functions are used (such as in [8, 9]), hashing can be a potential bottleneck in using Bloom filters, and reducing the number of required hashes can yield an effective speedup. This improvement was found empirically in the work of Dillinger and Manolios [8, 9], who suggested using the hash functions

$$g_i(x) = h_1(x) + ih_2(x) + i^2 \mod m,$$

where m is the size of the hash table.

Here we provide a full theoretical analysis that holds for a wide class of variations of this technique, justifies and gives insight into the previous empirical observations, and is interesting in its own right. In particular, our methodology generalizes the standard asymptotic analysis of a Bloom filter, exposing a new convergence result that provides a common unifying intuition for the asymptotic false positive probabilities of the standard Bloom filter and the generalized class of Bloom filter variants that we analyze in this paper. We obtain this result by a surprisingly simple approach; rather than attempt to directly analyze the asymptotic false positive probability, we formulate the initialization of the Bloom filter as a balls-and-bins experiment, prove a convergence result for that experiment, and then obtain the asymptotic false positive probability as a corollary.

We start by analyzing a specific, somewhat idealized Bloom filter variation that provides the main insights and intuition for deeper results. We then move to a more general setting that covers several issues that might arise in practice, such as when the size of the hash table is a power of two as opposed to a prime. Finally, we demonstrate the utility of this approach beyond the simple Bloom filter by showing how it can be used to reduce the number of hash functions required for Count-Min sketches [6], a variation of the Bloom filter idea used for keeping approximate counts of frequent items in data streams.

Before beginning, we note that Luecker and Molodowitch [19] and Schmidt and Siegel [25] have shown that in the setting of open addressed hash tables, the double hashing technique gives the same performance as uniform hashing. These results are similar in spirit to ours, but the Bloom filter setting is sufficiently different from that of an open addressed hash table that we do not see a direct connection. We also note that our use of hash functions of the form $g_i(x) = h_1(x) + ih_2(x)$ may appear similar to the use of pairwise independent hash functions, and that one might wonder whether there is any formal connection between the two techniques in the Bloom filter setting. Unfortunately, this is not the case; a straightforward modification of the standard Bloom filter analysis yields that if pairwise independent hash functions are used instead of fully random hash functions, then the space required to retain the same bound on the false positive probability increases by a constant factor. In contrast, we show that using the g_i's causes *no* increase in the false positive probability, so they can truly be used as a replacement for fully random hash functions.

2 Standard Bloom filters

We begin by reviewing the fundamentals of Bloom filters, based on the presentation of the survey [4], which we refer to for further details. A Bloom filter for representing a set $S = \{x_1, x_2, \ldots, x_n\}$ of n elements from a large universe U consists of an array of m bits, initially all set to 0. The filter uses k independent hash functions $h_1, \ldots, h_k$ with range $\{1, \ldots, m\}$, where it assumed that these hash functions map each element in the universe to a random number uniformly over the range. While the randomness of the hash functions is clearly an optimistic assumption, it appears to be suitable in practice [13, 24]. For each element $x \in S$, the bits $h_i(x)$ are set to 1 for $1 \leq i \leq k$. (A location can be set to 1 multiple times.) To check if an item y is in S, we check whether all $h_i(y)$ are set to 1. If not, then clearly y is not a member of S. If all $h_i(y)$ are set to 1, we assume that y is in S, and hence a Bloom filter may yield a *false positive*.

The probability of a false positive for an element not in the set, or the *false positive probability*, can be estimated in a straightforward fashion, given our assumption that hash functions are perfectly random. After all the elements of S are hashed into the Bloom filter, the probability that a specific bit is still 0 is

$$p' = (1 - 1/m)^{kn} \approx \mathrm{e}^{-kn/m}.$$

In this section, we generally use the approximation $p = \mathrm{e}^{-kn/m}$ in place of p' for convenience.

If ρ is the proportion of 0 bits after all the n elements are inserted in the table, then conditioned on ρ the probability of a false positive is

$$(1 - \rho)^k \approx (1 - p')^k \approx (1 - p)^k = \left(1 - \mathrm{e}^{-kn/m}\right)^k.$$

These approximations follow since $\mathbf{E}[\rho] = p'$, and ρ can be shown to be highly concentrated around p' using standard techniques. It is easy to show that the expression $\left(1 - \mathrm{e}^{-kn/m}\right)^k$ is minimized when $k = \ln 2 \cdot (m/n)$, giving a false positive probability f of

$$f = \left(1 - \mathrm{e}^{-kn/m}\right)^k = (1/2)^k \approx (0.6185)^{m/n}.$$

In practice, k must be an integer, and a smaller, sub-optimal k might be preferred since this reduces the number of hash functions that have to be computed.

This analysis provides us (roughly) with the probability that a single item $z \notin S$ gives a false positive. We would like to make a broader statement, that in fact this gives a false positive *rate*. That is, if we choose a large number of *distinct* elements not in S, the fraction of them that yield false positives is approximately f. This result follows immediately from the fact that ρ is highly concentrated around p', and for this reason, the false positive probability is sometimes called the *false positive rate*. As we will see, in our variations, it is not always as clear that the false positive probability acts like a false positive rate, and so we clearly distinguish between the two concepts. (Indeed, we think this clarification is a contribution of this paper.)

Before moving on, we note that sometimes Bloom filters are described slightly differently, with each hash function having a disjoint range of m/k consecutive bit locations instead of having one shared array of m bits. We refer to this variant as a *partitioned* Bloom filter. Repeating the analysis above, we find that in this case the probability that a specific bit is 0 is

$$\left(1 - \frac{k}{m}\right)^n \approx \mathrm{e}^{-kn/m},$$

and so, asymptotically, the performance is the same as the original scheme. In practice, however, the partitioned Bloom filter tends to perform slightly worse than the non-partitioned Bloom filter. This is explained by the observation that

$$\left(1 - \frac{1}{m}\right)^{kn} \geq \left(1 - \frac{k}{m}\right)^{n},$$

so partitioned filters tend to have more 1's than non-partitioned filters, resulting in larger false positive probabilities.

3 A Simple Construction Using Two Hash Functions

As an instructive example case, we consider a specific application of the general technique described in the introduction. We devise a Bloom filter that uses k fully random hash functions on some universe U of items, each with range $\{0, 1, 2, \dots, p-1\}$ for a prime p. Our hash table consists of $m = kp$ bits; each hash function is assigned a disjoint subarray of p bits in the filter, that we treat as numbered $\{0, 1, 2, \dots, p-1\}$. Our k hash functions will be of the form $g_i(x) = h_1(x) + ih_2(x) \bmod p$, where $h_1(x)$ and $h_2(x)$ are two independent, uniform random hash functions on the universe with range $\{0, 1, 2, \dots, p-1\}$, and throughout we assume that i ranges from 0 to $k-1$.

As with a standard partitioned Bloom filter, we fix some set $S \subseteq U$ and initialize the filter with S by first setting all of the bits to 0 and then, for each $x \in S$ and i, setting the $g_i(x)$-th bit of the i-th subarray to 1. For any $y \in U$, we answer a query of the form "Is $y \in S$?" with "Yes" if and only if the $g_i(y)$-th bit of the i-th subarray is 1 for every i. Thus, an item $z \notin S$ generates a false positive if and only if each of its hash locations in the array is also a hash location for some $x \in S$.

The advantage of our simplified setting is that for any two elements $x, y \in U$, exactly one of the following three cases occurs:

- $g_i(x) \neq g_i(y)$ for all i, or
- $g_i(x) = g_i(y)$ for exactly one i, or
- $g_i(x) = g_i(y)$ for all i.

That is, because we have partitioned the bit array into disjoint hash tables, each hash function can be considered separately. Moreover, by working modulo p, we have arranged that if $g_i(x) = g_i(y)$ for at least two values of i, then we must have $h_1(x) = h_1(y)$ and $h_2(x) = h_2(y)$, so all hash values are the same. This codifies the intuition behind our result: the most likely way for a false positive to occur is when each element in the Bloom filter set S collides with at most one array bit corresponding to the element generating the false positive; other events that cause an element to generate a false positive occur with vanishing probability. It is this intuition that motivates our analysis; in Section 4, we consider more general cases where other non-trivial collisions can occur.

Proceeding formally, we fix a set $S = \{x_1, x_2, \dots, x_n\}$ of n elements from U and another element $z \notin S$, and compute the probability that z yields a false positive. A false positive corresponds to the event $\mathcal{F}$ that for each i there is (at least) one j such that $g_i(z) = g_i(x_j)$. Obviously, one way this can occur is if $h_1(x_j) = h_1(z)$ and $h_2(x_j) = h_2(z)$ for some j. The probability of this event $\mathcal{E}$ is

$$\mathbf{Pr}(\mathcal{E}) = 1 - \left(1 - 1/p^2\right)^n = 1 - \left(1 - k^2/m^2\right)^n.$$

Notice that when $m/n = c$ is a constant and k is a constant, as is standard for a Bloom filter, we have $\mathbf{Pr}(\mathcal{E}) = o(1)$. Now since

$$\begin{aligned}\mathbf{Pr}(\mathcal{F}) &= \mathbf{Pr}(\mathcal{F} \mid \mathcal{E})\,\mathbf{Pr}(\mathcal{E}) + \mathbf{Pr}(\mathcal{F} \mid \neg\mathcal{E})\,\mathbf{Pr}(\neg\mathcal{E}) \\ &= \mathbf{Pr}(\mathcal{E}) + \mathbf{Pr}(\mathcal{F} \mid \neg\mathcal{E})\,\mathbf{Pr}(\neg\mathcal{E}) \\ &= o(1) + \mathbf{Pr}(\mathcal{F} \mid \neg\mathcal{E})(1 - o(1)),\end{aligned}$$

it suffices to consider $\mathbf{Pr}(\mathcal{F} \mid \neg\mathcal{E})$ to obtain the (constant) asymptotic false positive probability.

Conditioned on $\neg\mathcal{E}$ and $(h_1(z), h_2(z))$, the pair $(h_1(x_j), h_2(x_j))$ is uniformly distributed over the $p^2 - 1$ values in $V = \{0, \ldots, p-1\}^2 - \{(h_1(z), h_2(z))\}$. Of these, for each $i^* \in \{0, \ldots, k-1\}$, the $p - 1$ pairs in

$$V_{i^*} = \{(a, b) \in V \,:\, a \equiv i^*(h_2(z) - b) + h_1(z) \bmod p,\ b \not\equiv h_2(z) \bmod p\}$$

are the ones such that if $(h_1(x_j), h_2(x_j)) \in V_{i^*}$, then i^* is the unique value of i such that $g_i(x_j) = g_i(z)$. We can therefore view the conditional probability as a variant of a balls-and-bins problem. There are n balls (each corresponding to some $x_j \in S$), and k bins (each corresponding to some $i^* \in \{0, \ldots, k-1\}$). With probability $k(p-1)/(p^2-1) = k/(p+1)$ a ball lands in a bin, and with the remaining probability it is discarded; when a ball lands in a bin, the bin it lands in is chosen uniformly at random. What is the probability that all of the bins have at least one ball?

This question is surprisingly easy to answer. By the Poisson approximation and the fact that $p = m/k = cn/k$, the total number of balls that are not discarded has distribution $\mathrm{Bin}(n, k/(p+1)) \approx \mathrm{Po}(k^2/c)$, where $\mathrm{Bin}(\cdot,\cdot)$ and $\mathrm{Po}(\cdot)$ denote the binomial and Poisson distributions, respectively. Since each ball that is not discarded lands in a bin chosen at random, the joint distribution of the number of balls in the bins is asymptotically the same as the joint distribution of k independent $\mathrm{Po}(k/c)$ random variables, by a standard property of Poisson random variables. The probability that each bin has a least one ball now clearly converges to

$$\mathbf{Pr}(\mathrm{Po}(k/c) > 0)^k = \left(1 - \mathrm{e}^{-k/c}\right)^k = \left(1 - \mathrm{e}^{-kn/m}\right)^k,$$

which is the asymptotic false positive probability for a standard Bloom filter.

We make the above argument much more general and rigorous in Section 4, but for now we emphasize that we have actually characterized much more than just the false positive probability of our Bloom filter variant. In fact, we have characterized the asymptotic joint distribution of the number of items in S hashing to the locations used by some $z \notin S$ as being independent $\mathrm{Po}(k/c)$ random variables. Furthermore, from a technical perspective, this approach appears fairly robust. In particular, the above analysis uses only the facts that the probability that some $x \in S$ shares more than one of z's hash locations is $o(1)$, and that if some $x \in S$ shares exactly one of z's hash locations, then that hash location is nearly uniformly distributed over z's hash locations. These observations suggest that the techniques used in this section can be generalized to handle a much wider class of Bloom filter variants, and form the intuitive basis for the arguments in Section 4.

Now, as in Section 2, we must argue that the asymptotic false positive probability also acts like a false positive *rate*. Similar to the case for the standard Bloom filter, this fact boils down to a concentration argument. Once the set S is hashed, there is a set

$$B = \{(b_1, b_2) : h_1(z) = b_1 \text{ and } h_2(z) = b_2 \text{ implies } z \text{ gives a false positive}\}.$$

Conditioned on $|B|$, the probability of a false positive for any element in $U - S$ is $|B|/p^2$, and these events are independent. If we show that $|B|$ is concentrated around its expectation, it follows easily that the fraction of false positives in a set of distinct elements not in S is concentrated around the false positive probability.

A simple Doob martingale argument suffices (e.g. [21, Section 12.5]). Each hashed element of S can change the number of pairs in B by at most kp in either direction. This observation follows immediately from the fact that given any element x, its hash values $h_1(x)$ and $h_2(x)$, and some $i \in \{0, \dots, k-1\}$, there are exactly p solutions $(b_1, b_2) \in \{0, \dots, p-1\}^2$ to the equation

$$h_1(x) + ih_2(x) \equiv b_1 + ib_2 \pmod{p}.$$

By [21, Section 12.5], we now have that for any $\epsilon > 0$,

$$\mathbf{Pr}(|B - \mathbf{E}[B]| \geq \epsilon p^2) \leq 2 \exp\left[-2\epsilon^2 p^2 / nk^2\right].$$

It is now easy to derive the desired conclusion. We defer further details until Section 7, where we consider a similar but much more general argument.

As an aside, we remark that unlike the analysis of the standard Bloom filter in Section 2, here the fraction ρ of zeros in the Bloom filter array is not important for showing that the false positive probability acts like a false positive rate. However, it can be easily shown that ρ has essentially the same asymptotic expectation in this Bloom filter variation as for a standard Bloom filter, and that ρ is highly concentrated around its mean. (The same observations hold for the specific schemes in Section 5.)

4 A General Framework

In this section, we introduce a general framework for analyzing Bloom filter variants, such as the one examined in Section 3. We start with some new notation. For any integer ℓ, we define the set $[\ell] = \{0, 1, \dots, \ell - 1\}$ (note that this definition is slightly non-standard). We denote the support of a random variable X by $\mathrm{Supp}(X)$. For a multi-set M, we use $|M|$ to denote the number of distinct elements of M, and $\|M\|$ to denote the number of elements of M with multiplicity. For two multi-sets M and M', we define $M \cap M'$ and $M \cup M'$ to be, respectively, the intersection and union of M' as *multi-sets*. Furthermore, in an abuse of standard notation, we define the statement $i, i \in M$ as meaning that i is an element of M of multiplicity at least 2.

We are now ready to define the framework. As before, U denotes the universe of items and $S \subseteq U$ denotes the set of n items for which the Bloom filter will answer membership queries. We define a *scheme* to be a method of assigning hash locations to every element of U. Formally, a scheme is specified by a joint distribution of discrete random variables $\{H(u) : u \in U\}$ (implicitly parameterized by n), where for $u \in U$, $H(u)$ represents the multi-set of hash-locations assigned to u by the scheme. We do not require a scheme to be defined for every value of n, but we do insist that it be defined for infinitely many values of n, so that we may take limits as $n \to \infty$. For example, for the class of schemes discussed in Section 3, we think of the constants k and c as being fixed to give a particular scheme that is defined for those values of n such that $p \triangleq m/k$ is a prime, where $m \triangleq cn$. Since there are infinitely many primes, the asymptotic behavior of this scheme as $n \to \infty$ is well-defined and is the same as in Section 3, where we let m be a free parameter and analyzed the behavior as $n, m \to \infty$ subject to m/n and k being fixed constants, and m/k being prime.

Having defined the notion of a scheme, we may now formalize some important concepts with new notation (all of which is implicitly parameterized by n). We define H to be the set of all hash locations that can be assigned by the scheme (formally, H is the set of elements that appear in some multi-set in the support of $H(u)$, for some $u \in U$). For $x \in S$ and $z \in U - S$, define $C(x, z) = H(x) \cap H(z)$ to be the multi-set of hash collisions of x with z. We let $\mathcal{F}(z)$ denote the *false positive event* for $z \in U - S$, which occurs when each of z's hash locations is also a hash location for some $x \in S$.

In the schemes that we consider, $\{H(u) : u \in U\}$ will always be independent and identically distributed. In this case, $\mathbf{Pr}(\mathcal{F}(z))$ is the same for all $z \in U - S$, as is the joint distribution of $\{C(x, z) : x \in S\}$. Thus, to simplify the notation, we may fix an arbitrary $z \in U - S$ and simply use $\mathbf{Pr}(\mathcal{F})$ instead of $\mathbf{Pr}(\mathcal{F}(z))$ to denote the false positive probability, and we may use $\{C(x) : x \in S\}$ instead of $\{C(x, z) : x \in S\}$ to denote the joint probability distribution of the multi-sets of hash collisions of elements of S with z.

The main technical result of this section is the following key theorem, which is a formalization and generalization of the analysis of the asymptotic false positive probability in Section 3.

Theorem 4.1. *Fix a scheme. Suppose that there are constants λ and k and functions $\gamma_1(n) = o(1/n)$ and $\gamma_2(n) = o(1)$ such that:*

1. *$\{H(u) : u \in U\}$ are independent and identically distributed.*
2. *For $u \in U$, $\|H(u)\| = k$.*
3. *For $x \in S$,* $$\mathbf{Pr}(\|C(x)\| = i) = \begin{cases} 1 - \frac{\lambda}{n} + O(\gamma_1(n)) & i = 0 \\ \frac{\lambda}{n} + O(\gamma_1(n)) & i = 1 \\ O(\gamma_1(n)) & i > 1 \end{cases}.$$
4. *For $x \in S$, $\max_{i \in H} \left| \mathbf{Pr}(i \in C(x) \mid \|C(x)\| = 1,\ i \in H(z)) - \frac{1}{k} \right| = O(\gamma_2(n))$.*

Then $\lim_{n \to \infty} \mathbf{Pr}(\mathcal{F}) = \left(1 - \mathrm{e}^{-\lambda/k}\right)^k$.

Remark 1. It is not difficult to verify that the scheme analyzed in Section 3 satisfies the conditions of Theorem 4.1 with $\lambda = k^2/c$. However, more complicated schemes are not so amenable to a direct application of Theorem 4.1. Thus, after proving the theorem, we give a result that identifies another set of conditions that imply the hypotheses of Theorem 4.1 and are easier to verify.

Proof. For ease of exposition, we assign every element of $H(z)$ a unique number in $[k]$ (treating multiple instances of the same hash location as distinct elements). More formally, we define an arbitrary bijection f_M from M to $[k]$ for every multi-set $M \subseteq H$ with $\|M\| = k$ (where f_M treats multiple instances of the same hash location in M as distinct elements), and label the elements of $H(z)$ according to $f_{H(z)}$. This convention allows us to identify the elements of $H(z)$ by numbers $i \in [k]$, rather than hash locations $i \in H$.

For $i \in [k]$ and $x \in S$, define $X_i(x) = 1$ if $i \in C(x)$ and 0 otherwise, and define $X_i = \sum_{x \in S} X_i(x)$. Note that $i \in C(x)$ is an abuse of notation; what we really mean is $f^{-1}_{H(z)}(i) \in C(x)$, although we will continue using the former since it is much less cumbersome.

We show that $X^n \triangleq (X_0, \ldots, X_{k-1})$ converges in distribution to a vector $P \triangleq (P_0, \ldots, P_{k-1})$ of k independent Poisson random variables with parameter λ/k, as $n \to \infty$. To do this, we make use of moment generating functions. For a random variable R, the moment generating function of R is defined by $M_R(t) \triangleq \mathbf{E}[\exp(tR)]$. We show that for any $t_0, \ldots, t_k$,

$$\lim_{n \to \infty} M_{\sum_{i=0}^{k-1} t_i X_i}(t_k) = M_{\sum_{i=0}^{k-1} t_i P_i}(t_k),$$

which is sufficient by [1, Theorem 29.4 and p. 390], since

$$\begin{aligned} M_{\sum_{i=0}^{k-1} t_i P_i}(t_k) &= \mathbf{E}\left[\mathrm{e}^{t_k \sum_{i\in[k]} t_i P_i}\right] \\ &= \prod_{i\in[k]} \mathbf{E}\left[\mathrm{e}^{t_k t_i \mathrm{Po}(\lambda/k)}\right] \\ &= \prod_{i\in[k]} \sum_{j=0}^{\infty} \mathrm{e}^{-\lambda/k} \frac{\lambda^j}{k^j j!} \mathrm{e}^{t_k t_i j} \\ &= \prod_{i\in[k]} \mathrm{e}^{\frac{\lambda}{k}\left(\mathrm{e}^{t_k t_i} - 1\right)} \\ &= \mathrm{e}^{\frac{\lambda}{k}\left(\sum_{i\in[k]} \mathrm{e}^{t_k t_i} - 1\right)} < \infty, \end{aligned}$$

where the first step is just the definition of the moment generating function, the second step follows from independence of the $t_i P_i(\lambda_k)$'s, the third step is just the definition of the Poisson distribution, the fourth step follows from the Taylor series for e^x, and the fifth step is obvious.

Proceeding, we write

$$
\begin{aligned}
&M_{\sum_{i\in[k]} t_i X_i}(t_k)\\
&= M_{\sum_{i\in[k]} t_i \sum_{x\in S} X_i(x)}(t_k)\\
&= M_{\sum_{x\in S}\sum_{i\in[k]} t_i X_i(x)}(t_k)\\
&= \left(M_{\sum_{i\in[k]} t_i X_i(x)}(t_k)\right)^n\\
&= \Bigg(\mathbf{Pr}(\|C(x)\| = 0)\\
&\qquad + \sum_{j=1}^{k} \mathbf{Pr}(\|C(x)\| = j) \sum_{T\subseteq[k]:|T|=j} \mathbf{Pr}(C(x) = f_{H(z)}^{-1}(T) \mid \|C(x)\| = j)\mathrm{e}^{t_k \sum_{i\in T} t_i}\Bigg)^n\\
&= \left(1 - \frac{\lambda}{n} + \frac{\lambda}{n}\sum_{i\in[k]} \mathbf{Pr}(i \in C(x) \mid \|C(x)\| = 1)\mathrm{e}^{t_k t_i} + o(1/n)\right)^n\\
&= \left(1 - \frac{\lambda}{n} + \frac{\lambda}{n}\sum_{i\in[k]} \left(\frac{1}{k} + o(1)\right)\mathrm{e}^{t_k t_i} + o(1/n)\right)^n\\
&= \left(1 - \frac{\lambda}{n} + \frac{\lambda \sum_{i\in[k]} \mathrm{e}^{t_k t_i}}{kn} + o(1/n)\right)^n\\
&\to \mathrm{e}^{-\lambda + \frac{\lambda}{k}\sum_{i\in[k]} \mathrm{e}^{t_k t_i}} \qquad \text{as } n \to \infty\\
&= \mathrm{e}^{\frac{\lambda}{k}\left(\sum_{i\in[k]}\left(\mathrm{e}^{t_k t_i} - 1\right)\right)}\\
&= M_{\sum_{i\in[k]} t_i \mathrm{Po}_i(\lambda_k)}(t_k).
\end{aligned}
$$

The first two steps are obvious. The third step follows from the fact that the $H(x)$'s are independent and identically distributed (for $x \in S$) conditioned on $H(z)$, so the $\sum_{i\in[k]} t_i X_i(x)$'s are too, since each is a function of the corresponding $H(x)$. The fourth step follows from the definition of the moment generating function. The fifth and sixth steps follow from the assumptions on the distribution of $C(x)$ (in the sixth step, the conditioning on $i \in H(z)$ is implicit in our convention that associates integers in $[k]$ with the elements of $H(z)$). The seventh, eighth, and ninth steps are obvious, and the tenth step follows from a previous computation.

Now fix some bijection $g : \mathbb{Z}_{\geq 0}^k \to \mathbb{Z}_{\geq 0}$, and define $h : \mathbb{Z}_{\geq 0} \to \{0, 1\} : h(x) = 1$ if and only if every coordinate of $g^{-1}(x)$ is greater than 0. Since $\{X^n\}$ converges to P in distribution, $\{g(X^n)\}$ converges to $g(P)$ in distribution, because g is a bijection and X^n and P have discrete distributions. Skorohod's Representation Theorem [1, Theorem 25.6] now implies that there is some probability space where one may define random variables $\{Y_n\}$ and P', where $Y_n \sim g(X^n)$ and $P' \sim g(P)$, and $\{Y_n\}$ converges to P' almost surely. Of course, since the Y_n's only take integer values, whenever $\{Y_n\}$ converges to P', there must be some n_0 such that $Y_{n_0} = Y_{n_1} = P'$ for any $n_1 > n_0$, and so

$\{h(Y_n)\}$ trivially converges to $h(P')$. Therefore, $\{h(Y_n)\}$ converges to $h(P')$ almost surely, so

$$\begin{aligned}
\mathbf{Pr}(\mathcal{F}) &= \mathbf{Pr}(\forall i \in [k], X_i > 0) \\
&= \mathbf{E}[h(g(X^n))] \\
&= \mathbf{E}[h(Y_n)] \\
&\to \mathbf{E}[h(P')] \qquad \text{as } n \to \infty \\
&= \mathbf{Pr}(\mathrm{Po}(\lambda/k) > 0)^k \\
&= \left(1 - \mathrm{e}^{-\lambda/k}\right)^k,
\end{aligned}$$

where the fourth step is the only nontrivial one, and it follows from [1, Theorem 5.4]. $\square$

It turns out that the conditions of Theorem 4.1 can be verified very easily in many cases.

Lemma 4.1. *Fix a scheme. Suppose that there are constants λ and k and a function $\gamma(n) = o(1/n)$ such that:*

1. *$\{H(u) : u \in U\}$ are independent and identically distributed.*
2. *For $u \in U$, $\|H(u)\| = k$.*
3. *For $u \in U$, $\max_{i \in H} \left|\mathbf{Pr}(i \in H(u)) - \frac{\lambda}{kn}\right| = O(\gamma(n))$.*
4. *For $u \in U$, $\max_{i_1, i_2 \in H} \mathbf{Pr}(i_1, i_2 \in H(u)) = O(\gamma(n))$.*
5. *The set of all possible hash locations H satisfies $|H| = O(n)$.*

Then the conditions of Theorem 4.1 hold with $\gamma_1(n) = \gamma(n)$ and $\gamma_2(n) = n\gamma(n)$ (and the same values for λ and k).

Remark 2. Recall that, under our notation, the statement $i, i \in H(u)$ is true if and only if i is an element of $H(u)$ of multiplicity at least 2.

Proof. The proof is essentially just a number of applications of the first two Boole-Bonferroni inequalities (e.g. [22, Proposition C.2]). We adopt the convention introduced in the proof of Theorem 4.1 where the elements of $H(z)$ are identified by the integers in $[k]$. For $i \in [k]$, we continue to abuse notation and write $i \in H(x)$ as shorthand for $f^{-1}_{H(z)}(i) \in H(x)$ where doing so does not cause confusion.

The first two conditions of Theorem 4.1 are trivially satisfied. For the third condition, observe that for any $j \in \{2, \ldots, k\}$ and $x \in S$,

$$\begin{aligned}
\mathbf{Pr}(\|C(x)\| = j) &\le \mathbf{Pr}(\|C(x)\| > 1) \\
&= \mathbf{Pr}\left(\exists i_1 < i_2 \in [k] : f^{-1}_{H(z)}(i_1), f^{-1}_{H(z)}(i_2) \in H(x) \text{ or } \exists i \in H : i \in H(x), i, i \in H(z)\right) \\
&\le \mathbf{Pr}\left(\exists i_1 < i_2 \in [k] : f^{-1}_{H(z)}(i_1), f^{-1}_{H(z)}(i_2) \in H(x)\right) + \mathbf{Pr}\left(\exists i \in H : i \in H(x), i, i \in H(z)\right) \\
&= \mathbf{Pr}\left(\exists i_1 < i_2 \in [k] : i_1, i_2 \in H(x)\right) + \mathbf{Pr}\left(\exists i \in H : i \in H(x), i, i \in H(z)\right) \\
&\le \sum_{i_1 < i_2 \in [k]} \mathbf{Pr}(i_1, i_2 \in H(x)) + \sum_{i \in H} \mathbf{Pr}(i \in H(x))\, \mathbf{Pr}(i, i \in H(z)) \\
&\le \binom{k}{2} O(\gamma(n)) + |H| \left(\frac{\lambda}{kn} + O(\gamma(n))\right) O(\gamma(n)) \\
&= O(\gamma(n)) + O(n)O(\gamma(n)/n) \\
&= O(\gamma(n)),
\end{aligned}$$

and

$$\mathbf{Pr}(\|C(x)\| = 1) \le \mathbf{Pr}(|C(x)| \ge 1) \le \sum_{i \in [k]} \mathbf{Pr}(i \in H(x)) \le k\left(\frac{\lambda}{kn} + O(\gamma(n))\right) = \frac{\lambda}{n} + O(\gamma(n)),$$

and

$$\begin{aligned}
\mathbf{Pr}(\|C(x)\| \ge 1) &= \mathbf{Pr}\left(\bigcup_{i \in [k]} i \in H(x)\right) \\
&\ge \sum_{i \in [k]} \mathbf{Pr}(i \in H(x)) - \sum_{i_1 < i_2 \in [k]} \mathbf{Pr}(i_1, i_2 \in H(x)) \\
&\ge k\left(\frac{\lambda}{kn} + O(\gamma(n))\right) - k^2 O(\gamma(n)) \\
&= \frac{\lambda}{n} + O(\gamma(n)),
\end{aligned}$$

so

$$\begin{aligned}
\mathbf{Pr}(\|C(x)\| = 1) &= \mathbf{Pr}(\|C(x)\| \ge 1) - \mathbf{Pr}(\|C(x)\| > 1) \\
&\ge \frac{\lambda}{n} + O(\gamma(n)) - O(\gamma(n)) \\
&= \frac{\lambda}{n} + O(\gamma(n)).
\end{aligned}$$

Therefore,

$$\mathbf{Pr}(\|C(x)\| = 1) = \frac{\lambda}{n} + O(\gamma(n)),$$

and

$$\mathbf{Pr}(\|C(x)\| = 0) = 1 - \sum_{j=1}^{k} \mathbf{Pr}(\|C(x)\| = j) = 1 - \frac{\lambda}{n} + O(\gamma(n)).$$

We have now shown that the third condition of Theorem 4.1 is satisfied.

For the fourth condition, we observe that for any $i \in [k]$ and $x \in S$,

$$\mathbf{Pr}(i \in C(x), \|C(x)\| = 1) \leq \mathbf{Pr}(i \in H(x)) = \frac{\lambda}{kn} + O(\gamma(n)),$$

and

$$\begin{aligned}\mathbf{Pr}(i \in C(x), \|C(x)\| = 1) &= \mathbf{Pr}(i \in H(x)) - \mathbf{Pr}(i \in H(x), \|C(x)\| > 1)\\ &\geq \mathbf{Pr}(i \in H(x)) - \mathbf{Pr}(\|C(x)\| > 1)\\ &= \frac{\lambda}{kn} + O(\gamma(n)) - O(\gamma(n)),\end{aligned}$$

so

$$\mathbf{Pr}(i \in C(x), \|C(x)\| = 1) = \frac{\lambda}{kn} + O(\gamma(n)),$$

implying that

$$\mathbf{Pr}(i \in C(x) \mid \|C(x)\| = 1) = \frac{\mathbf{Pr}(i \in C(x), \|C(x)\| = 1)}{\mathbf{Pr}(\|C(x)\| = 1)} = \frac{\frac{\lambda}{kn} + O(\gamma(n))}{\frac{\lambda}{n} + O(\gamma(n))} = \frac{1}{k} + O(n\gamma(n)),$$

completing the proof (the conditioning on $i \in H(z)$ is once again implied by the convention that associates elements of $[k]$ with the hash locations in $H(z)$). □

5 Some Specific Schemes

We are now ready to analyze some specific schemes. In particular, we examine a natural generalization of the scheme described in Section 3, as well as the double hashing and enhanced double hashing schemes introduced in [8, 9]. In both of these cases, we consider a Bloom filter consisting of an array of $m = cn$ bits and k hash functions, where $c > 0$ and $k \geq 1$ are fixed constants. The nature of the hash functions depends on the particular scheme under consideration.

5.1 Partition Schemes

First, we consider the class of *partition schemes*, where the Bloom filter is defined by an array of m bits that is partitioned into k disjoint arrays of $m' = m/k$ bits (we require that m be divisible by k), and an item $u \in U$ is hashed to location

$$h_1(u) + ih_2(u) \bmod m'$$

of array i, for $i \in [k]$, where h_1 and h_2 are independent fully random hash functions with codomain $[m']$. Note that the scheme analyzed in Section 3 is a partition scheme where m' is prime (and so is denoted by p in Section 3).

Unless otherwise stated, henceforth we do all arithmetic involving h_1 and h_2 modulo m'. We prove the following theorem concerning partition schemes.

Theorem 5.1. *For a partition scheme,* $\lim_{n\to\infty} \mathbf{Pr}(\mathcal{F}) = \left(1 - \mathrm{e}^{-k/c}\right)^k$.

Proof. We show that the $H(u)$'s satisfy the conditions of Lemma 4.1 with $\lambda = k^2/c$ and $\gamma(n) = 1/n^2$. For $i \in [k]$ and $u \in U$, define $g_i(u) = (i, h_1(u) + ih_2(u))$ and $H(u) = (g_i(u) : i \in [k])$. That is, $g_i(u)$ is u's ith hash location, and $H(u)$ is the multi-set of u's hash locations. This notation is obviously consistent with the definitions required by Lemma 4.1.

Since h_1 and h_2 are independent and fully random, the first two conditions are trivial. The last condition is also trivial, since there are $m = cn$ possible hash locations. For the remaining two conditions, fix $u \in U$. Observe that for $(i, r) \in [k] \times [m']$,

$$\mathbf{Pr}((i, r) \in H(u)) = \mathbf{Pr}(h_1(u) = r - ih_2(u)) = 1/m' = (k^2/c)/kn,$$

and that for distinct $(i_1, r_1), (i_2, r_2) \in [k] \times [m']$, we have

$$\begin{aligned}
&\mathbf{Pr}((i_1, r_1), (i_2, r_2) \in H(u)) \\
&= \mathbf{Pr}(i_1 \in H(u))\, \mathbf{Pr}(i_2 \in H(u) \mid i_1 \in H(u)) \\
&= \frac{1}{m'} \mathbf{Pr}(h_1(u) = r_2 - i_2 h_2(u) \mid h_1(u) = r_1 - i_1 h_2(u)) \\
&= \frac{1}{m'} \mathbf{Pr}((i_1 - i_2) h_2(u) = r_1 - r_2) \\
&\le \frac{1}{m'} \cdot \frac{\gcd(|i_2 - i_1|, m')}{m'} \le \frac{k}{(m')^2} = O(1/n^2),
\end{aligned}$$

where the fourth step is the only nontrivial step, and it follows from the standard fact that for any $r, s \in [m]$, there are at most $\gcd(r, m)$ values $t \in [m]$ such that $rt \equiv s \bmod m$ (see, for example, [15, Proposition 3.3.1]). Finally, since it is clear that from the definition of the scheme that $|H(u)| = k$ for all $u \in U$, we have that for any $(i, r) \in [k] \times [m']$, $\mathbf{Pr}((i, r), (i, r) \in H(u)) = 0$. $\square$

5.2 (Enhanced) Double Hashing Schemes

Next, we consider the class of double hashing and enhanced double hashing schemes, which are analyzed empirically in [8, 9]. In these schemes, an item $u \in U$ is hashed to location

$$h_1(u) + ih_2(u) + f(i) \bmod m$$

of the array of m bits, for $i \in [k]$, where h_1 and h_2 are independent fully random hash functions with codomain $[m]$, and $f : [k] \to [m]$ is an arbitrary function. When $f(i) \equiv 0$, the scheme is called a *double hashing scheme*. Otherwise, it is called an *enhanced double hashing scheme (with f)*. We show that the asymptotic false positive probability for an (enhanced) double hashing scheme is the same as for a standard Bloom filter.

Theorem 5.2. *For any (enhanced) double hashing scheme,*

$$\lim_{n \to \infty} \mathbf{Pr}(\mathcal{F}) = \left(1 - \mathrm{e}^{-k/c}\right)^k.$$

Remark 3. The result holds for any choice of f. In fact, f can even be drawn from an arbitrary probability distribution over $[m]^{[k]}$, as long as it is drawn independently of the two random hash functions h_1 and h_2.

Proof. We proceed by showing that this scheme satisfies the conditions of Lemma 4.1 with $\lambda = k^2/c$ and $\gamma(n) = 1/n^2$. Since h_1 and h_2 are independent and fully random, the first two conditions trivially hold. The last condition is also trivial, since there are $m = cn$ possible hash locations.

Showing that the third and fourth conditions hold requires more effort. First, we need some notation. For $u \in U$, $i \in [k]$, define

$$\begin{aligned} g_i(u) &= h_1(u) + ih_2(u) + f(i) \\ H(u) &= (g_i(u) \,:\, i \in [k]). \end{aligned}$$

That is, $g_i(u)$ is u's ith hash location, and $H(u)$ is the multi-set of u's hash locations. This notation is obviously consistent with the definitions required by Lemma 4.1. Fix $u \in U$. For $r \in [m]$,

$$\mathbf{Pr}(\exists j \in [k] : g_j(u) = r) \leq \sum_{j \in [k]} \mathbf{Pr}(h_1(u) = r - jh_2(u) - f(j)) = \frac{k}{m}.$$

Furthermore, for any $j_1, j_2 \in [k]$ and $r_1, r_2 \in [m]$

$$\begin{aligned} \mathbf{Pr}(g_{j_1}(u) = r_1, g_{j_2}(u) = r_2) &= \mathbf{Pr}(g_{j_1}(u) = r_1)\,\mathbf{Pr}(g_{j_2}(u) = r_2 \mid g_{j_1}(u) = r_1) \\ &= \frac{1}{m}\mathbf{Pr}(g_{j_2}(u) = r_2 \mid g_{j_1}(u) = r_1) \\ &= \frac{1}{m}\mathbf{Pr}((j_1 - j_2)h_2(u) = r_1 - r_2 + f(j_2) - f(j_1)) \\ &\leq \frac{1}{m} \cdot \frac{\gcd(|j_1 - j_2|, m)}{m} \\ &\leq \frac{1}{m} \cdot \frac{k}{m} \\ &= \frac{k}{m^2} \\ &= O(1/n^2), \end{aligned}$$

where the fourth step is the only nontrivial step, and it follows from the standard fact that for any $r, s \in [m]$, there are at most $\gcd(r, m)$ values $t \in [m]$ such that $rt \equiv s \bmod m$ (see, for example, [15, Proposition 3.3.1]). Therefore, for $r \in [m]$,

$$\begin{aligned} \mathbf{Pr}(\exists j \in [k] : g_j(u) = r) &\geq \sum_{j \in [k]} \mathbf{Pr}(g_j(u) = r) - \sum_{j_1 < j_2 \in [k]} \mathbf{Pr}(g_{j_1}(u) = r, g_{j_2}(u) = r) \\ &\geq \frac{k}{m} - k^2 O(1/n^2) \\ &= \frac{k^2/c}{n} + O(1/n^2), \end{aligned}$$

implying that

$$\mathbf{Pr}(r \in H(u)) = \mathbf{Pr}(\exists j \in [k] : g_j(u) = r) = \frac{k^2/c}{n} + O(1/n^2),$$

so the third condition of Lemma 4.1 holds. For the fourth condition, fix any $r_1, r_2 \in [m]$. Then

$$\mathbf{Pr}(r_1, r_2 \in H(u)) \leq \sum_{j_1, j_2 \in [k]} \mathbf{Pr}(g_{j_1}(u) = r_1, g_{j_2}(u) = r_2) \leq k^2 O(1/n^2) = O(1/n^2),$$

completing the proof. □

6 Rate of Convergence

In the previous sections, we identified a broad class of non-standard Bloom filter schemes that have the same asymptotic false positive probability as a standard Bloom filter. For many applications, we would also like to know that these asymptotics kick in fairly quickly, for reasonable values of n. With these applications in mind, we provide an analysis of the rate of convergence in Theorem 4.1, and then apply that analysis to the specific schemes discussed in Section 5. Our results indicate that those schemes yield performance almost identical to that of a standard Bloom filter for a wide range of practical settings. Furthermore, in Section 8, we show the results of some simple experiments as further evidence of this fact.

The rate of convergence analysis proceeds along the following lines, where the underlying intuition is drawn from the analysis of the asymptotic false positive probability in Section 3, and we assume the hypotheses of Theorem 4.1. First, for each $x \in S$, we couple $\|C(x)\|$ with a $\mathrm{Bern}(\lambda/n)$ random variable B_x (where $\mathrm{Bern}(\cdot)$ denotes the Bernoulli distribution). (We specify exactly how to do the coupling later.) We then define a $\mathrm{Bin}(n, \lambda/n)$ random variable $B = \sum_{x \in S} B_x$. In the terminology of Section 3, each $x \in S$ represents a ball which is discarded if and only if $B_x = 0$, so B is the total number of balls that are not discarded. Next, conditioned on $T = \{x : B_x = 1\}$, for each $x \in T$, we couple the smallest element C_x of $f_{H(z)}(C(x))$ with a random variable T_x selected uniformly from $[k]$ (recall from the proof of Theorem 4.1 that $f_{H(z)}$ defines a correspondence between $H(z)$ and $[k]$). (In the asymptotically insignificant case where $\|C(x)\| = 0$, we simply define $C_x = -1$.) In the terminology of Section 3, T is the set of balls that are thrown into the bins, and for each $x \in T$, the random variable T_x represents the bin where it lands.

We can now bound the difference between the false positive probability (for a particular value of n) and the probability that every bin in $[k]$ is hit by at least one ball by the probability that at least one of the random variables just defined is different than the random variable to which it is coupled. Thus, we relate the true false positive probability to the same simple balls and bins experiment as in Section 3. Finally, as in Section 3, the asymptotics of the balls and bins experiment are easy to analyze; we just couple B with a $\mathrm{Po}(\lambda)$ random variable Y and bound $\mathbf{Pr}(B \neq Y)$. (This is because, for both the experiment where B balls are thrown (that is, not discarded) and the experiment where $\mathrm{Po}(\lambda)$ balls are thrown, each ball is placed in a bin that is chosen randomly from the k bins, so for each ball that is thrown in both experiments, the random variables indicating the bins where it lands in the two experiments can be trivially coupled.)

We now formalize the above argument.

Theorem 6.1. *Consider a scheme that satisfies the hypotheses of Theorem 4.1. Then*

$$\left|\mathbf{Pr}(\mathcal{F}) - \left(1 - \mathrm{e}^{-\lambda/k}\right)^k\right| = O(n\gamma_1(n) + \gamma_2(n) + 1/n).$$

Proof. With the above outline in mind, define the events

$$\mathcal{E}_1 = \{\forall x \in S \ \|C(x)\| = B_x\}$$
$$\mathcal{E}_2 = \{\forall x \in T \ C_x = T_x\}.$$

Let $\mathcal{F}'$ denote the event that in the experiment where Y balls are thrown randomly into k bins, each bin receives at least one ball. Since $Y \sim \mathrm{Po}(\lambda)$, a standard fact of Poisson random variables tells us that the joint distribution of number of balls in each bin is the same as the joint distribution of k independent $\mathrm{Po}(\lambda/k)$ random variables. Thus, $\mathbf{Pr}(\mathcal{F}') = (1-\exp[-\lambda/k])^k$. Letting $\mathbf{1}(\cdot)$ denote the indicator function, we write

$$\begin{aligned}
\left|\mathbf{Pr}(\mathcal{F}) - (1-\exp[-\lambda/k])^k\right| &= \left|\mathbf{Pr}(\mathcal{F}) - \mathbf{Pr}(\mathcal{F}')\right| \\
&= \left|\mathbf{E}\left[\mathbf{1}(\mathcal{F}) - \mathbf{1}(\mathcal{F}')\right]\right| \\
&\le \mathbf{E}\left[\left|\mathbf{1}(\mathcal{F}) - \mathbf{1}(\mathcal{F}')\right|\right] \\
&= \mathbf{Pr}(\mathbf{1}(\mathcal{F}) \neq \mathbf{1}(\mathcal{F}')) \\
&\le \mathbf{Pr}(\neg\mathcal{E}_1 \cup \neg\mathcal{E}_2 \cup (B \neq Y)) \\
&\le \mathbf{Pr}(\neg\mathcal{E}_1 \cup \neg\mathcal{E}_2) + \mathbf{Pr}(B \neq Y) \\
&= \mathbf{Pr}(\neg\mathcal{E}_1) + \mathbf{Pr}(\mathcal{E}_1)\,\mathbf{Pr}(\neg\mathcal{E}_2 \mid \mathcal{E}_1) + \mathbf{Pr}(B \neq Y)
\end{aligned}$$

where we have used the fact that if $\mathbf{1}(\mathcal{F}) \neq \mathbf{1}(\mathcal{F}')$, then some random variable in the coupling established above is not equal to the random variable to which it is coupled.

Before continuing, we must address the issue of actually constructing the couplings described above. Of course, our goal is to find a coupling so that random variables with similar distributions are likely to be equal in the probability space where they are coupled. Indeed, it follows from standard facts (see, for example, [14, Exercise 4.12.5]) that we can construct the couplings so that for any two coupled random variables X_1 and X_2, we have

$$\mathbf{Pr}(X_1 \neq X_2) = \frac{1}{2} \sum_{x \in \mathrm{Supp}(X_1) \cup \mathrm{Supp}(X_2)} |\,\mathbf{Pr}(X_1 = x) - \mathbf{Pr}(X_2 = x)|.$$

We perform all of the couplings in this way, except for the coupling of B and Y, which we defer until later in the proof.

Thus, for a fixed $x \in S$, we have $\mathbf{Pr}(\|C(x)\| \neq B_x) = O(\gamma_1(n))$, and so a union bound gives $\mathbf{Pr}(\neg\mathcal{E}_1) = O(n\gamma_1(n))$. Also, for any $x \in S$,

$$\mathbf{Pr}(C_x \neq T_x \mid x \in T, \mathcal{E}_1) \le kO(\gamma_2(n)) = O(\gamma_2(n)).$$

By a union bound (and the fact that $\{H(u) : u \in U\}$ are independent and identically distributed), we now have

$$\mathbf{Pr}(\exists x \in T \ : \ C_x \neq T_x \mid |T|, \mathcal{E}_1) \le |T| O(\gamma_2(n)).$$

Therefore,

$$\begin{aligned}
\mathbf{Pr}(\neg\mathcal{E}_2 \mid \mathcal{E}_1) &= \mathbf{E}\left[\mathbf{Pr}(\exists x \in T \ : \ f_{H(z)}(C_x) \neq T_x \mid |T|, \mathcal{E}_1) \mid \mathcal{E}_1\right] \\
&\le \mathbf{E}[|T| \mid \mathcal{E}_1] O(\gamma_2(n)) \\
&\le \frac{\mathbf{E}[|T|] O(\gamma_2(n))}{\mathbf{Pr}(\mathcal{E}_1)} \\
&= \frac{\lambda O(\gamma_2(n))}{\mathbf{Pr}(\mathcal{E}_1)} \\
&= \frac{O(\gamma_2(n))}{\mathbf{Pr}(\mathcal{E}_1)}.
\end{aligned}$$

Combining these results gives

$$\begin{aligned}\left|\mathbf{Pr}(\mathcal{F}) - (1-\exp[-\lambda/k])^k\right| &\le \mathbf{Pr}(\neg\mathcal{E}_1) + \mathbf{Pr}(\mathcal{E}_1)\,\mathbf{Pr}(\neg\mathcal{E}_2 \mid \mathcal{E}_1) + \mathbf{Pr}(B \neq Y) \\ &\le O(n\gamma_1(n)) + O(\gamma_2(n)) + \mathbf{Pr}(B \neq Y).\end{aligned}$$

It now suffices to specify the coupling between the $\mathrm{Bin}(n, \lambda/n)$ random variable B and the $\mathrm{Po}(\lambda)$ random variable Y. By applying standard facts (see, for example, [14, Section 4.12]), we can couple B and Y so that $\mathbf{Pr}(B \neq Y) \le \lambda^2/n$. Therefore,

$$\left|\mathbf{Pr}(\mathcal{F}) - \left(1 - \mathrm{e}^{-\lambda/k}\right)^k\right| = O(n\gamma_1(n) + \gamma_2(n) + 1/n),$$

completing the proof. □

We now use Theorem 6.1 to bound the rate of convergence in Theorems 5.1 and 5.2.

Theorem 6.2. *For any partition or (enhanced) double hashing scheme,*

$$\left|\mathbf{Pr}(\mathcal{F}) - \left(1 - \mathrm{e}^{-\lambda/k}\right)^k\right| = O(1/n).$$

Proof. In the proofs of Theorems 5.1 and 5.2, we show that all of these schemes satisfy the conditions of Lemma 4.1 with $\gamma(n) = 1/n^2$. Thus, by Lemma 4.1, all of these schemes satisfy the conditions of Theorem 4.1 with $\gamma_1(n) = 1/n^2$ and $\gamma_2(n) = 1/n$. Theorem 6.1 now gives the desired result. □

7 Multiple Queries

In the previous sections, we analyzed the behavior of $\mathbf{Pr}(\mathcal{F}(z))$ for some fixed z and moderately sized n. Unfortunately, this quantity is not directly of interest in most applications. Instead, one is usually concerned with certain characteristics of the distribution of the number of, say, $z_1, \ldots, z_\ell \in U - S$ for which $\mathcal{F}(z)$ occurs. In other words, rather than being interested in the probability that a particular false positive occurs, we are concerned with, for example, the fraction of distinct queries on elements of $U - S$ posed to the filter for which it returns false positives. Since $\{\mathcal{F}(z) : z \in U - S\}$ are not independent, the behavior of $\mathbf{Pr}(\mathcal{F})$ alone does not directly imply results of this form. This section is devoted to overcoming this difficulty.

Now, it is easy to see that in the schemes that we analyze here, once the hash locations for every $x \in S$ have been determined, the events $\{\mathcal{F}(z) : z \in U - S\}$ are independent and occur with equal probability. More formally, $\{\mathbf{1}(\mathcal{F}(z)) : z \in U - S\}$ are conditionally independent and identically distributed given $\{H(x) : x \in S\}$. Thus, conditioned on $\{H(x) : x \in S\}$, an enormous number of classical convergence results (e.g. the law of large numbers and the central limit theorem) can be applied to $\{\mathbf{1}(\mathcal{F}(z)) : z \in U - S\}$.

These observations motivate a general technique for deriving the sort of convergence results for $\{\mathbf{1}(\mathcal{F}(z)) : z \in U - S\}$ that one might desire in practice. First, we show that with high probability over the set of hash locations used by elements of S (that is, $\{H(x) : x \in S\}$), the random variables $\{\mathbf{1}(\mathcal{F}(z)) : z \in U - S\}$ are essentially independent $\mathrm{Bern}(p)$ random variables, for $p \triangleq \lim_{n\to\infty} \mathbf{Pr}(\mathcal{F})$. From a technical standpoint, this result is the most important in this section. Next, we show how to use that result to prove counterparts to the classical convergence theorems mentioned above that hold in our setting.

Proceeding formally, we begin with a critical definition.

Definition 7.1. Consider any scheme where $\{H(u) : u \in U\}$ are independent and identically distributed. Write $S = \{x_1, \ldots, x_n\}$. The *false positive rate* is defined to be the random variable

$$R = \mathbf{Pr}(\mathcal{F} \mid H(x_1), \ldots, H(x_n)).$$

The false positive rate gets its name from the fact that, conditioned on R, the random variables $\{\mathbf{1}(\mathcal{F}(z)) : z \in U - S\}$ are independent $\mathrm{Bern}(R)$ random variables. Thus, the fraction of a large number of queries on elements of $U - S$ posed to the filter for which it returns false positives is very likely to be close to R. In this sense, R, while a random variable, acts like a rate for $\{\mathbf{1}(\mathcal{F}(z)) : z \in U - S\}$.

It is important to note that in much of literature concerning standard Bloom filters, the false positive rate is not defined as above. Instead the term is often used as a synonym for the false positive probability. Indeed, for a standard Bloom filter, the distinction between the two concepts as we have defined them is unimportant in practice, since, as mentioned in Section 2, one can easily show that R is very close to $\mathbf{Pr}(\mathcal{F})$ with extremely high probability (see, for example, [20]). It turns out that this result generalizes very naturally to the framework presented in this paper, and so the practical difference between the two concepts is largely unimportant even in our very general setting. However, the proof is more complicated than in the case of a standard Bloom filter, and so we will be very careful to use the terms as we have defined them.

Theorem 7.1. *Consider a scheme where the conditions of Lemma 4.1 hold. Furthermore, assume that there is some function g and independent identically distributed random variables $\{V_u : u \in U\}$, each of which is uniformly distributed over some finite set V, such that for $u \in U$ we have $H(u) = g(V_u)$. Define*

$$\begin{aligned} p &= \left(1 - \mathrm{e}^{-\lambda/k}\right)^k \\ \Delta &= \max_{i \in H} \mathbf{Pr}(i \in H(u)) - \frac{\lambda}{nk} \quad (= o(1/n)) \\ \xi &= nk\Delta(2\lambda + k\Delta) \quad (= o(1)) \end{aligned}$$

Then for any $\epsilon = \epsilon(n) > 0$ with $\epsilon = \omega(|\mathbf{Pr}(\mathcal{F}) - p|)$, for n sufficiently large so that $\epsilon > |\mathbf{Pr}(\mathcal{F}) - p|$,

$$\mathbf{Pr}(|R - p| > \epsilon) \leq 2 \exp\left[\frac{-2n(\epsilon - |\mathbf{Pr}(\mathcal{F}) - p|)^2}{\lambda^2 + \xi}\right].$$

Furthermore, for any function $h(n) = o(\min(1/|\mathbf{Pr}(\mathcal{F}) - p|, \sqrt{n}))$, we have that $(R - p)h(n)$ converges to 0 in probability as $n \to \infty$.

Remark 4. Since $|\mathbf{Pr}(\mathcal{F}) - p| = o(1)$ by Lemma 4.1, we may take $h(n) = 1$ in Theorem 7.1 to conclude that R converges to p in probability as $n \to \infty$.

Remark 5. From the proofs of Theorems 5.1 and 5.2, it is easy to see that for both the partition and (enhanced) double hashing schemes, $\Delta = 0$, so $\xi = 0$ for both schemes as well.

Remark 6. We have added a new condition on the distribution of $H(u)$, but it trivially holds in all of the schemes that we discuss in this paper (since, for independent fully random hash functions h_1 and h_2, the random variables $\{(h_1(u), h_2(u)) : u \in U\}$ are independent and identically distributed, and $(h_1(u), h_2(u))$ is uniformly distributed over its support, which is finite).

Proof. The proof is essentially a standard application of Azuma's inequality to an appropriately defined Doob martingale. Specifically, we employ the technique discussed in [21, Section 12.5].

For convenience, write $S = \{x_1, \ldots, x_n\}$. For $h_1, \ldots, h_n \in \mathrm{Supp}(H(u))$, define

$$f(h_1, \ldots, h_n) = \mathbf{Pr}(\mathcal{F} \mid H(x_1) = h_1, \ldots, H(x_n) = h_n),$$

and note that $R = f(H(x_1), \ldots, H(x_n))$. Now consider some d such that for any $h_1, \ldots, h_j$, h'_j, $h_{j+1}, \ldots, h_n \in \mathrm{Supp}(H(u))$,

$$|f(h_1, \ldots, h_n) - f(h_1, \ldots, h_{j-1}, h'_j, h_{j+1}, \ldots, h_n)| \leq d.$$

Since the $H(x_i)$'s are independent, we may apply the result of [21, Section 12.5] to obtain

$$\mathbf{Pr}(|R - \mathbf{E}[R]| \geq \delta) \leq 2\mathrm{e}^{-2\delta^2/nd^2},$$

for any $\delta > 0$.

To find an appropriate and small choice for d, we write

$$\begin{aligned}
&|f(h_1, \ldots, h_n) - f(h_1, \ldots, h_{j-1}, h'_j, h_{j+1}, \ldots, h_n)| \\
&= |\,\mathbf{Pr}(\mathcal{F} \mid H(x_1) = h_1, \ldots, H(x_n) = h_n) \\
&\quad - \mathbf{Pr}(\mathcal{F} \mid H(x_1) = h_1, \ldots, H(x_{j-1}) = h_{j-1}, H(x_j) = h'_j, H(x_{j+1}) = h_{j+1}, \ldots H(x_n) = h_n)| \\
&= \frac{\left| |\{v \in V \,:\, g(v) \subseteq \bigcup_{i=1}^n h_i\}| - \left|\left\{v \in V \,:\, g(v) \subseteq \bigcup_{i=1}^n \left\{ \begin{array}{ll} h'_j & i = j \\ h_i & i \neq j \end{array} \right\}\right\}\right| \right|}{|V|} \\
&\leq \frac{\max_{v' \in V} |\,\{v \in V \,:\, |g(v) \cap g(v')| \geq 1\}\,|}{|V|} \\
&= \max_{M' \in \mathrm{Supp}(H(u))} \mathbf{Pr}(|H(u) \cap M'| \geq 1),
\end{aligned}$$

where the first step is just the definition of f, the second step follows from the definitions of V_u and g, the third step holds since changing one of the h_i's to some $M' \in \mathrm{Supp}(H(u))$ cannot change

$$\left|\left\{v \in V \,:\, g(v) \subseteq \bigcup_{i=1}^n h_i\right\}\right|$$

by more than

$$\left|\left\{v \in V \,:\, |g(v) \cap M'| \geq 1\right\}\right|,$$

and the fourth step follows from the definitions of V_u and g.

Now consider any fixed $M' \in \mathrm{Supp}(H(u))$, and let $y_1, \ldots, y_{|M'|}$ be the distinct elements of M'.

Recall that $\|M'\| = k$, so $|M'| \leq k$. Applying a union bound, we have that

$$\begin{aligned}
\mathbf{Pr}(|H(u) \cap M'| \geq 1) &= \mathbf{Pr}\left(\bigcup_{i=1}^{|M'|} y_i \in H(u)\right) \\
&\leq \sum_{i=1}^{|M'|} \mathbf{Pr}(y_i \in H(u)) \\
&\leq \sum_{i=1}^{|M'|} \frac{\lambda}{kn} + \Delta \\
&\leq \frac{\lambda}{n} + k\Delta.
\end{aligned}$$

Therefore, we may set $d = \frac{\lambda}{n} + k\Delta$ to obtain

$$\mathbf{Pr}(|R - \mathbf{E}[R]| > \delta) \leq 2\exp\left[\frac{-2n\delta^2}{\lambda^2 + \xi}\right],$$

for any $\delta > 0$. Since $\mathbf{E}[R] = \mathbf{Pr}(\mathcal{F})$, we write (for sufficiently large n so that $\epsilon > |\mathbf{Pr}(\mathcal{F}) - p|$)

$$\begin{aligned}
\mathbf{Pr}(|R - p| > \epsilon) &\leq \mathbf{Pr}(|R - \mathbf{Pr}(\mathcal{F})| > \epsilon - |\mathbf{Pr}(\mathcal{F}) - p|) \\
&\leq 2\exp\left[\frac{-2n(\epsilon - |\mathbf{Pr}(\mathcal{F}) - p|)^2}{\lambda^2 + \xi}\right].
\end{aligned}$$

To complete the proof, we see that for any constant $\delta > 0$,

$$\mathbf{Pr}(|R - p|h(n) > \delta) = \mathbf{Pr}(|R - p| > \delta/h(n)) \to 0 \quad \text{as } n \to \infty,$$

where the second step follows from the fact that $|\mathbf{Pr}(\mathcal{F}) - p| = o(1/h(n))$, so for sufficiently large n,

$$\begin{aligned}
\mathbf{Pr}(|R - p| > \delta/h(n)) &\leq 2\exp\left[\frac{-2n(\delta/h(n) - |\mathbf{Pr}(\mathcal{F}) - p|)^2}{\lambda^2 + \xi}\right] \\
&\leq 2\exp\left[-\frac{\delta^2}{\lambda^2 + \xi} \cdot \frac{n}{h(n)^2}\right] \\
&\to 0 \quad \text{as } n \to \infty,
\end{aligned}$$

and the last step follows from the fact that $h(n) = o(\sqrt{n})$. □

Since, conditioned on R, the events $\{\mathcal{F}(z) : z \in U - S\}$ are independent and each occur with probability R, Theorem 7.1 suggests that $\{\mathbf{1}(\mathcal{F}(z)) : z \in U - S\}$ are essentially independent $\text{Bern}(p)$ random variables. We formalize this idea in the next result, where we use Theorem 7.1 to prove versions of the strong law of large numbers, the weak law of large numbers, Hoeffding's inequality, and the central limit theorem.

Theorem 7.2. *Consider a scheme that satisfies the conditions of Theorem 7.1. Let $Z \subseteq U - S$ be countably infinite, and write $Z = \{z_1, z_2, \ldots\}$. Then we have:*

1.

$$\mathbf{Pr}\left(\lim_{\ell\to\infty}\frac{1}{\ell}\sum_{i=1}^{\ell}\mathbf{1}(\mathcal{F}_n(z_i)) = R_n\right) = 1.$$

2. For any $\epsilon > 0$, for n sufficiently large so that $\epsilon > |\,\mathbf{Pr}(\mathcal{F}) - p|$,

$$\mathbf{Pr}\left(\left|\lim_{\ell\to\infty}\frac{1}{\ell}\sum_{i=1}^{\ell}\mathbf{1}(\mathcal{F}_n(z_i)) - p\right| > \epsilon\right) \leq 2\exp\left[\frac{-2n(\epsilon - |\,\mathbf{Pr}(\mathcal{F}) - p|)^2}{\lambda^2+\xi}\right].$$

In particular, $\lim_{\ell\to\infty}\frac{1}{\ell}\sum_{i=1}^{\ell}\mathbf{1}(\mathcal{F}_n(z_i))$ converges to p in probability as $n\to\infty$.

3. For any function $Q(n)$, $\epsilon > 0$, and n sufficiently large so that $\epsilon/2 > |\,\mathbf{Pr}(\mathcal{F}) - p|$,

$$\mathbf{Pr}\left(\left|\frac{1}{Q(n)}\sum_{i=1}^{Q(n)}\mathbf{1}(\mathcal{F}_n(z_i)) - p\right| > \epsilon\right) \leq 2\mathrm{e}^{-Q(n)\epsilon^2/2} + 2\exp\left[\frac{-2n(\epsilon/2 - |\,\mathbf{Pr}(\mathcal{F}) - p|)^2}{\lambda^2+\xi}\right].$$

4. For any function $Q(n)$ with $\lim_{n\to\infty}Q(n) = \infty$ and $Q(n) = o(\min(1/|\,\mathbf{Pr}(\mathcal{F}) - p|^2, n))$,

$$\sum_{i=1}^{Q(n)}\frac{\mathbf{1}(\mathcal{F}_n(z_i)) - p}{\sqrt{Q(n)p(1-p)}} \to \mathrm{N}(0,1) \text{ in distribution as } n\to\infty.$$

Remark 7. By Theorem 6.2, $|\,\mathbf{Pr}(\mathcal{F}) - p| = O(1/n)$ for both the partition and double hashing schemes introduced in Section 5. Thus, for each of the schemes, the condition $Q(n) = o(\min(1/|\,\mathbf{Pr}(\mathcal{F}) - p|^2, n))$ in the fourth part of Theorem 7.2 becomes $Q(n) = o(n)$.

Proof. Since, given R_n, the random variables $\{\mathbf{1}(\mathcal{F}_n(z)) : z \in Z\}$ are conditionally independent $\mathrm{Bern}(R_n)$ random variables, a direct application of the strong law of large numbers yields the first item.

For the second item, we note that the first item implies that

$$\lim_{\ell\to\infty}\frac{1}{\ell}\sum_{i=1}^{\ell}\mathbf{1}(\mathcal{F}_n(z_i)) \sim R_n.$$

A direct application of Theorem 7.1 then gives the result.

For the third item, we write

$$\begin{aligned}
&\mathbf{Pr}\left(\left|\frac{1}{Q(n)}\sum_{i=1}^{Q(n)}\mathbf{1}(\mathcal{F}_n(z_i)) - p\right| > \epsilon\right)\\
&\leq \mathbf{Pr}\left(\left|\frac{1}{Q(n)}\sum_{i=1}^{Q(n)}\mathbf{1}(\mathcal{F}_n(z_i)) - R_n\right| > \epsilon/2 \mid |R_n - p| \leq \epsilon/2\right) + \mathbf{Pr}(|R_n - p| > \epsilon/2)\\
&\leq 2\mathrm{e}^{-Q(n)\epsilon^2/2} + 2\exp\left[\frac{-2n(\epsilon/2 - |\,\mathbf{Pr}(\mathcal{F}) - p|)^2}{\lambda^2+\xi}\right],
\end{aligned}$$

where the last step follows from Hoeffding's inequality and Theorem 7.1.

For the fourth item, we write

$$\sum_{i=1}^{Q(n)} \frac{\mathbf{1}(\mathcal{F}_n(z_i)) - p}{\sqrt{Q(n)p(1-p)}} = \sqrt{\frac{R_n(1-R_n)}{p(1-p)}} \left(\sum_{i=1}^{Q(n)} \frac{\mathbf{1}(\mathcal{F}_n(z_i)) - R_n}{\sqrt{Q(n)R_n(1-R_n)}} + (R_n - p)\sqrt{\frac{Q(n)}{R_n(1-R_n)}} \right).$$

By Slutsky's theorem, it suffices to show the following three items:

1. $R_n \to p$ in probability as $n \to \infty$,
2. $(R_n - p)\sqrt{Q(n)} \to 0$ in probability as $n \to \infty$, and
3. $$\sum_{i=1}^{Q(n)} \frac{\mathbf{1}(\mathcal{F}_n(z_i)) - R_n}{\sqrt{Q(n)R_n(1-R_n)}} \to N(0,1) \quad \text{in distribution as } n \to \infty.$$

The first item holds by Remark 4, and the second item holds by Theorem 7.1, since $\sqrt{Q(n)} = o(\min(1/|\mathbf{Pr}(\mathcal{F}) - p|, \sqrt{n}))$. The third item requires a little more work. First, we need a version of the central limit theorem that allows us to bound its rate of convergence.

Lemma 7.1. *[7, Theorem 24] Let $X_1, X_2, \ldots$ be independent random variables with some common distribution X, where $\mathbf{E}[X] = 0$ and $\mathbf{Var}[X] = 1$. For $n \geq 1$, let $Y_n = \sum_{i=1}^n X_i/\sqrt{n}$. Then there is some constant a such that for any $n \geq 1$ and $x \in \mathbb{R}$,*

$$|\mathbf{Pr}(Y_n \leq x) - \mathbf{Pr}(\mathrm{N}(0,1) \leq x)| \leq a\,\mathbf{E}\left[|X|^3\right]/\sqrt{n}.$$

Fix some constant $\epsilon > 0$ so that $I \triangleq [p-\epsilon, p+\epsilon] \subseteq (0,1)$, and let $b = \min_{x \in I} \sqrt{x(1-x)} > 0$. With Lemma 7.1 in mind, define

$$X_i(n) = \frac{\mathbf{1}(\mathcal{F}_n(z_i)) - R_n}{\sqrt{R_n(1-R_n)}}.$$

Since, given R_n, the random variables are independent $\mathrm{Bern}(R_n)$ random variables, Lemma 7.1 tells us that for any $x \in \mathbb{R}$,

$$\begin{aligned}
&\left| \mathbf{Pr}\left(\sum_{i=1}^{Q(n)} X_i(n)/\sqrt{Q(n)} \right) - \mathbf{Pr}(\mathrm{N}(0,1) \leq x) \right| \\
&\leq \mathbf{Pr}(|R_n - p| > \epsilon) + \left| \mathbf{Pr}\left(\sum_{i=1}^{Q(n)} \frac{X_i(n)}{\sqrt{Q(n)}} \leq x \;\middle|\; |R_n - p| \leq \epsilon \right) - \mathbf{Pr}(\mathrm{N}(0,1) \leq x) \right| \\
&\leq \mathbf{Pr}(|R_n - p| > \epsilon) + \frac{a(1/b)^3}{\sqrt{Q(n)}} \\
&\to 0 \quad \text{as } n \to \infty,
\end{aligned}$$

where the last step follows from Remark 4. □

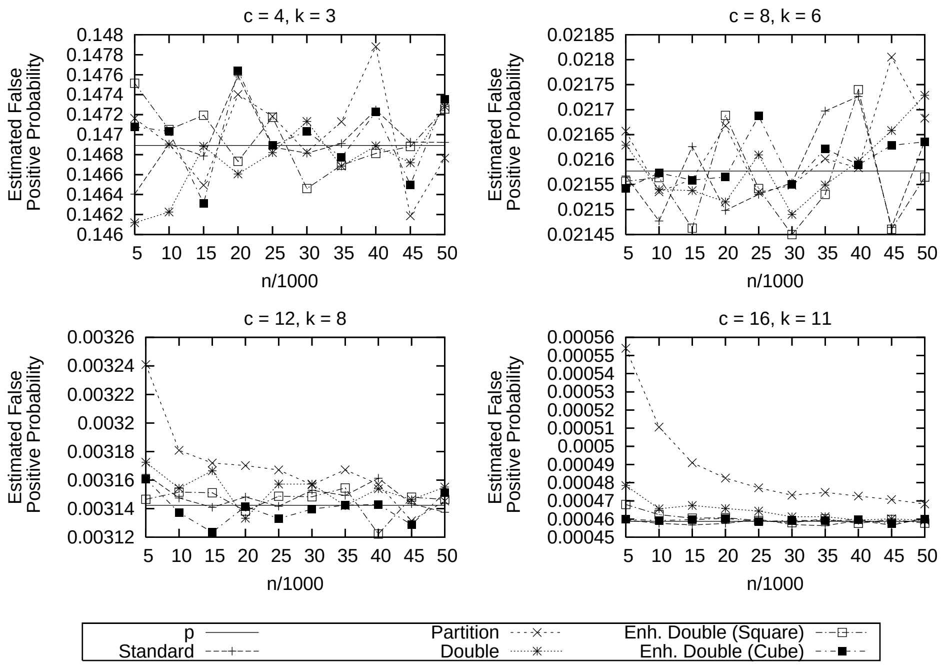

Figure 1: Estimates of the false positive probability for various schemes and parameters.

8 Experiments

In this section, we evaluate the theoretical results of the previous sections empirically for small values of n. We are interested in the following specific schemes: the standard Bloom filter scheme, the partition scheme, the double hashing scheme, and the enhanced double hashing schemes where $f(i) = i^2$ and $f(i) = i^3$.

For $c \in \{4, 8, 12, 16\}$, we do the following. First, compute the value of $k \in \{\lfloor c \ln 2 \rfloor, \lceil c \ln 2 \rceil\}$ that minimizes $p = (1 - \exp[-k/c])^k$. Next, for each of the schemes under consideration, repeat the following procedure 10,000 times: instantiate the filter with the specified values of n, c, and k, populate the filter with a set S of n items, and then query $\lceil 10/p \rceil$ elements not in S, recording the number Q of those queries for which the filter returns a false positive. We then approximate the false positive probability of the scheme by averaging the results over all 10,000 trials. Furthermore, we bin the results of the trials by their values for Q in order to examine the other characteristics of Q's distribution.

The results are shown in Figures 1 and 2. In Figure 1, we see that for small values of c, the different schemes are essentially indistinguishable from each other, and simultaneously have a false positive probability/rate close to p. This result is particularly significant since the filters that we are experimenting with are fairly small, supporting our claim that these schemes are useful even in settings with very limited space. However, we also see that for the slightly larger values of $c \in \{12, 16\}$, the partition scheme is no longer particularly useful for small values of n, while the other schemes are. This result is not particularly surprising; for large values of c and small values of n, the probability of a false positive can be substantially affected by the asymptotically vanishing probability that one element in the set can yield multiple collisions with an element not in the set, and this is somewhat larger in the partition scheme. Nevertheless, the difference is sufficiently

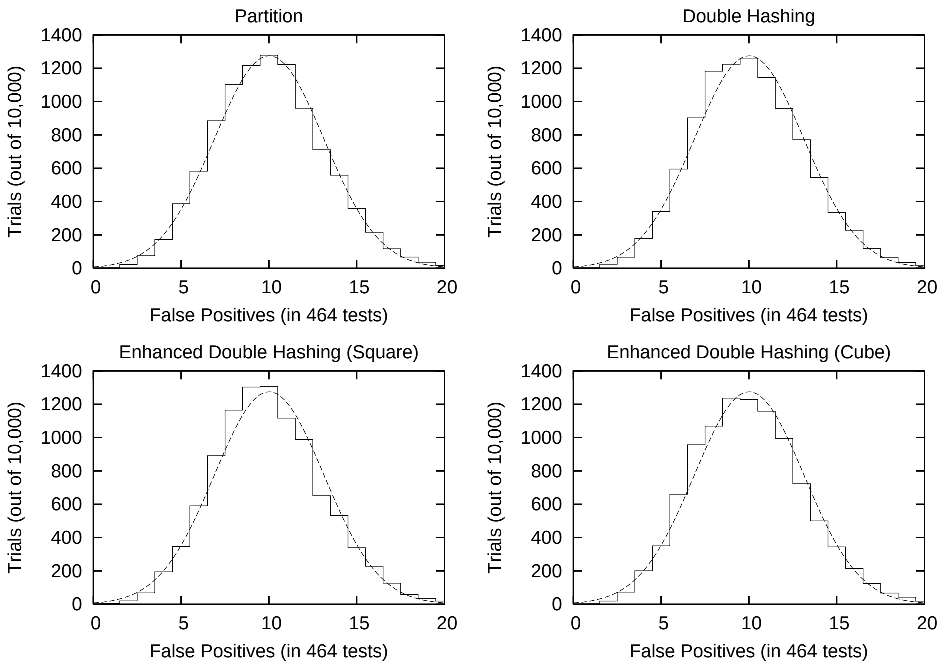

Figure 2: Estimate of distribution of Q (for $n = 5000$ and $c = 8$), compared with f.

small that the partition scheme might still be worthwhile in practice if parallelism is desired.

As an aside, Dillinger and Manolios [8, 9] observe that as c grows very large, various enhanced double hashing schemes (including triple hashing, where the g_i's use a third hash function with a coefficient that is quadratic in the index i) tend to perform slightly better than the regular double hashing scheme. Their results suggest that the difference is most likely due to differences in the constants in the rates of convergence of the various schemes. For the most part, this effect is not noticeable for the Bloom filter configurations that we consider, which are chosen to capture the typical Bloom filter setting where the false positive probability is small enough to be tolerable, but still non-negligible.

In Figure 2, we give histograms of the results from our experiments with $n = 5000$ and $c = 8$ for the partition and enhanced double hashing schemes. For this value of c, optimizing for k yields $k = 6$, so we have $p \approx 0.021577$ and $\lceil 10/p \rceil = 464$. In each plot, we compare the results to $f \triangleq 10,000\phi_{464p,464p(1-p)}$, where

$$\phi_{\mu,\sigma^2}(x) \triangleq \frac{\mathrm{e}^{-(x-\mu)^2/2\sigma^2}}{\sigma\sqrt{2\pi}}$$

denotes the density function of $\mathrm{N}(\mu, \sigma^2)$. As one would expect, given the central limit theorem result in the fourth part of Theorem 7.2, f provides a reasonable approximation to each of the histograms.

9 A Modified Count-Min Sketch

We now present a modification to the Count-Min sketch introduced in [6] that uses fewer hash functions in a manner similar to our improvement for Bloom filters, at the cost of a small space increase. We begin by reviewing the original data structure.

9.1 Count-Min Sketch Review

The following is an abbreviated review of the description given in [6]. A Count-Min sketch takes as input a stream of *updates* (i_t, c_t), starting from $t = 1$, where each *item* i_t is a member of a universe $U = \{1, \ldots, n\}$, and each *count* c_t is a positive number. (Extensions to negative counts are possible; we do not consider them here for convenience.) The state of the system at time T is given by a vector $\vec{a}(T) = (a_1(T), \ldots, a_n(T))$, where $a_j(T)$ is the sum of all c_t for which $t \leq T$ and $i_t = j$. We generally drop the T when the meaning is clear.

The Count-Min sketch consists of an array Count of width $w \triangleq \lceil \mathrm{e}/\epsilon \rceil$ and depth $d \triangleq \lceil \ln 1/\delta \rceil$: $\text{Count}[1, 1], \ldots, \text{Count}[d, w]$. Every entry of the array is initialized to 0. In addition, the Count-Min sketch uses d hash functions chosen independently from a pairwise independent family $\mathcal{H} : \{1, \ldots, n\} \to \{1, \ldots, w\}$.

The mechanics of the Count-Min sketch are extremely simple. Whenever an update (i, c) arrives, we increment $\text{Count}[j, h_j(i)]$ by c, for $j = 1, \ldots, d$. Whenever we want an estimate of a_i (called a *point query*), we compute

$$\hat{a}_i \triangleq \min_{j=1}^{d} \text{Count}[j, h_j(i)].$$

The fundamental result of Count-Min sketches is that for every i,

$$\hat{a}_i \geq a_i \quad \text{and} \quad \mathbf{Pr}(\hat{a}_i \leq a_i + \epsilon \|\vec{a}\|) \geq 1 - \delta,$$

where the norm is the L_1 norm. Surprisingly, this very simple bound allows for a number of sophisticated estimation procedures to be efficiently and effectively implemented on Count-Min sketches. The reader is once again referred to [6] for details.

9.2 Using Fewer Hash Functions

We now show how the improvements to Bloom filters discussed previously in this paper can be usefully applied to Count-Min sketches. Our modification maintains all of the essential features of Count-Min sketches, but reduces the required number of pairwise independent hash functions to $2\lceil (\ln 1/\delta)/(\ln 1/\epsilon) \rceil$. We expect that, in many settings, ϵ and δ will be related, so that only a constant number of hash functions will be required; in fact, in many such situations only two hash functions are required.

We describe a variation of the Count-Min sketch that uses just two pairwise independent hash functions and guarantees that

$$\hat{a}_i \geq a_i \quad \text{and} \quad \mathbf{Pr}(\hat{a}_i \leq a_i + \epsilon \|\vec{a}\|) \geq 1 - \epsilon.$$

Given such a result, it is straightforward to obtain a variation that uses $2\lceil (\ln 1/\delta)/(\ln 1/\epsilon) \rceil$ pairwise independent hash functions and achieves the desired failure probability δ: simply build $2\lceil (\ln 1/\delta)/(\ln 1/\epsilon) \rceil$ independent copies of this data structure, and always answer a point query with the minimum estimate given by one of those copies.

Our variation will use d tables numbered $\{0, 1, \ldots, d-1\}$, each with exactly w counters numbered $\{0, 1, \ldots, w-1\}$, where d and w will be specified later. We insist that w be prime. Just as in the original Count-Min sketch, we let $\text{Count}[j, k]$ denote the value of the kth counter in the jth table. We choose hash functions h_1 and h_2 independently from a pairwise independent family $\mathcal{H} : \{0, \ldots, n-1\} \to \{0, 1, \ldots, w-1\}$, and define $g_j(x) = h_1(x) + jh_2(x) \bmod w$ for $j = 0, \ldots, d-1$.

The mechanics of our data structure are the same as for the original Count-Min sketch. Whenever an update (i, c) occurs in the stream, we increment $\text{Count}[j, g_j(i)]$ by c, for $j = 0, \ldots, d-1$. Whenever we want an estimate of a_i, we compute

$$\hat{a}_i \triangleq \min_{j=0}^{d-1} \text{Count}[j, g_j(i)].$$

We prove the following result:

Theorem 9.1. *For the Count-Min sketch variation described above,*

$$\hat{a}_i \geq a_i \quad \textit{and} \quad \mathbf{Pr}(\hat{a}_i > a_i + \epsilon\|\vec{a}\|) \leq \frac{2}{\epsilon w^2} + \left(\frac{2}{\epsilon w}\right)^d.$$

In particular, for $w \geq 2\mathrm{e}/\epsilon$ and $\delta \geq \ln 1/\epsilon(1 - 1/2\mathrm{e}^2)$,

$$\hat{a}_i \geq a_i \quad \textit{and} \quad \mathbf{Pr}(\hat{a}_i > a_i + \epsilon\|\vec{a}\|) \leq \epsilon.$$

Proof. Fix some item i. Let A_i be the total count for all items z (besides i) with $h_1(z) = h_1(i)$ and $h_2(z) = h_2(i)$. Let $B_{j,i}$ be the total count for all items z with $g_j(i) = g_j(z)$, excluding i and items z counted in A_i. It follows that

$$\hat{a}_i = \min_{j=0}^{d-1} \text{Count}[j, g_j(i)] = a_i + A_i + \min_{j=0}^{d-1} B_{j,i}.$$

The lower bound now follows immediately from the fact that all items have nonnegative counts, since all updates are positive. Thus, we concentrate on the upper bound, which we approach by noticing that

$$\mathbf{Pr}(\hat{a}_i \geq a_i + \epsilon\|\vec{a}\|) \leq \mathbf{Pr}(A_i \geq \epsilon\|\vec{a}\|/2) + \mathbf{Pr}\left(\min_{j=0}^{d-1} B_{j,i} \geq \epsilon\|\vec{a}\|/2\right).$$

We first bound A_i. Letting $\mathbf{1}(\cdot)$ denote the indicator function, we have

$$\mathbf{E}[A_i] = \sum_{z \neq i} a_z \, \mathbf{E}[\mathbf{1}(h_1(z) = h_1(i) \wedge h_2(z) = h_2(i))] \leq \sum_{z \neq i} a_z/w^2 \leq \|\vec{a}\|/w^2,$$

where the first step follows from linearity of expectation and the second step follows from the definition of the hash functions. Markov's inequality now implies that

$$\mathbf{Pr}(A_i \geq \epsilon\|\vec{a}\|/2) \leq 2/\epsilon w^2.$$

To bound $\min_{j=0}^{d-1} B_{j,i}$, we note that for any $j \in \{0, \ldots, d-1\}$ and $z \neq i$,

$$\begin{aligned}
\mathbf{Pr}((h_1(z) \neq h_1(i) \vee h_2(z) \neq h_2(i)) \wedge g_j(z) = g_j(i)) &\leq \mathbf{Pr}(g_j(z) = g_j(i)) \\
&= \mathbf{Pr}(h_1(z) = h_1(i) + j(h_2(i) - h_2(z))) \\
&= 1/w,
\end{aligned}$$

so

$$\mathbf{E}[B_{j,i}] = \sum_{z \neq i} a_z \, \mathbf{E}[\mathbf{1}((h_1(z) \neq h_1(i) \vee h_2(z) \neq h_2(i)) \wedge g_j(z) = g_j(i))] \leq \|\vec{a}\|/w,$$

and so Markov's inequality implies that

$$\mathbf{Pr}(B_{j,i} \geq \epsilon\|\vec{a}\|/2) \leq 2/\epsilon w.$$

For arbitrary w, this result is not strong enough to bound $\min_{j=0}^{d-1} B_{j,i}$. However, since w is prime, each item z can only contribute to one $B_{k,i}$ (since if $g_j(z) = g_j(i)$ for two values of j, we must have $h_1(z) = h_1(i)$ and $h_2(z) = h_2(i)$, and in this case z's count is not included in any $B_{j,i}$). In this sense, the $B_{j,i}$'s are negatively dependent (specifically, they are negatively right orthant dependent) [11]. It follows that for any value v,

$$\mathbf{Pr}\left(\min_{j=0}^{d-1} B_{j,i} \geq v\right) \leq \prod_{j=0}^{d-1} \mathbf{Pr}(B_{j,i} \geq v).$$

In particular, we have that

$$\mathbf{Pr}\left(\min_{j=0}^{d-1} B_{j,i} \geq \epsilon\|\vec{a}\|/2\right) \leq (2/\epsilon w)^d,$$

so

$$\begin{aligned}\mathbf{Pr}(\hat{a}_i \geq a_i + \epsilon\|\vec{a}\|) &\leq \mathbf{Pr}(A_i \geq \epsilon\|\vec{a}\|/2) + \mathbf{Pr}\left(\min_{j=0}^{d-1} B_j, i \geq \epsilon\|\vec{a}\|/2\right) \\ &\leq \frac{2}{\epsilon w^2} + \left(\frac{2}{\epsilon w}\right)^d.\end{aligned}$$

And for $w \geq 2\mathrm{e}/\epsilon$ and $\delta \geq \ln 1/\epsilon(1 - 1/2\mathrm{e}^2)$, we have

$$\frac{2}{\epsilon w^2} + \left(\frac{2}{\epsilon w}\right)^d \leq \epsilon/2\mathrm{e}^2 + \epsilon(1 - 1/2\mathrm{e}^2) = \epsilon,$$

completing the proof. □

10 Conclusion

Bloom filters are simple randomized data structures that are extremely useful in practice. In fact, they are so useful that any significant reduction in the time required to perform a Bloom filter operation immediately translates to a substantial speedup for many practical applications. Unfortunately, Bloom filters are so simple that they do not leave much room for optimization.

This paper focuses on modifying Bloom filters to use less of the only resource that they traditionally use liberally: (pseudo)randomness. Since the only nontrivial computations performed by a Bloom filter are the constructions and evaluations of pseudorandom hash functions, any reduction in the required number of pseudorandom hash functions yields a nearly equivalent reduction in the time required to perform a Bloom filter operation (assuming, of course, that the Bloom filter is stored entirely in memory, so that random accesses can be performed very quickly).

We have shown that a Bloom filter can be implemented with only two pseudorandom hash functions without any increase in the asymptotic false positive probability. We have also shown that the asymptotic false positive probability acts, for all practical purposes and reasonable settings of a Bloom filter's parameters, like a false positive rate. This result has enormous practical significance, since the analogous result for standard Bloom filters is essentially the theoretical justification for their extensive use.

More generally, we have given a framework for analyzing modified Bloom filters, which we expect will be used in the future to refine the specific schemes that we analyzed in this paper. We also expect that the techniques used in this paper will be usefully applied to other data structures, as demonstrated by our modification to the Count-Min sketch.

Acknowledgments

We are very grateful to Peter Dillinger and Panagiotis Manolios for introducing us to this problem, providing us with advance copies of their work, and also for many useful discussions.

References

[1] P. Billingsley. Probability and Measure, Third Edition. John Wiley & Sons, 1995.

[2] F. Bonomi, M. Mitzenmacher, R. Panigrahy, S. Singh, and G. Varghese. Beyond Bloom filters: from approximate membership checks to approximate state machines. *Proceedings of the 2006 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, pp. 315-326, 2006.

[3] F. Bonomi, M. Mitzenmacher, R. Panigrahy, S. Singh, and G. Varghese. Bloom Filters via d-left Hashing and Dynamic Bit Reassignment. *Proceedings of the Allerton Conference on Communication, Control and Computing*, 2006.

[4] A. Broder and M. Mitzenmacher. Network Applications of Bloom Filters: A Survey. *Internet Mathematics*, 1(4):485-509, 2004.

[5] B. Chazelle, J. Kilian, R. Rubinfeld, and A. Tal. The Bloomier filter: an efficient data structure for static support lookup tables. *Proceedings of the 15th ACM/SIAM Symposium on Discrete Algorithms (SODA)*, pp 30-39, 2004.

[6] G. Cormode and S. Muthukrishnan. Improved Data Stream Summaries: The Count-Min Sketch and its Applications. DIMACS Technical Report 2003-20, 2003.

[7] H. Cramer. Random Variables and Probability Distributions, Third Edition. Cambridge University Press, 1970.

[8] P. C. Dillinger and P. Manolios. Bloom Filters in Probabilistic Verification. In *Proceedings of the 5th International Conference on Formal Methods in Computer-Aided Design (FMCAD)*, pp. 367-381, 2004.

[9] P. C. Dillinger and P. Manolios. Fast and Accurate Bitstate Verification for SPIN. In *Proceedings of the 11th International SPIN Workshop on Model Checking of Software (SPIN)*, pp. 57-75, 2004.

[10] B. Donnet, B. Baynat, and T. Friedman. Retouched Bloom Filters: Allowing Networked Applications to Flexibly Trade Off False Positives Against False Negatives. *arxiv, cs.NI/0607038*, 2006.

[11] D. P. Dubhashi and D. Ranjan. Balls and Bins: A Case Study in Negative Dependence. *Random Structures and Algorithms*, 13(2):99-124, 1998.

[12] C. Estan and G. Varghese. New Directions in Traffic Measurement and Accounting: Focusing on the Elephants, Ignoring the Mice. *ACM Transactions on Computer Systems*, 21(3):270-313, 2003.

[13] L. Fan, P. Cao, J. Almeida, and A. Z. Broder. Summary cache: a scalable wide-area Web cache sharing protocol. *IEEE/ACM Transactions on Networking*, 8(3):281-293, 2000.

[14] G. Grimmett and D. Stirzaker. Probability and Random Processes, Third Edition. Oxford University Press, 2001.

[15] K. Ireland and M. Rosen. A Classical Introduction to Modern Number Theory, Second Edition. Springer-Verlag, 1990.

[16] A. Kirsch and M. Mitzenmacher. Building a Better Bloom Filter. In *Proceedings of the 14th Annual European Symposium on Algorithms (ESA)*, 2006.

[17] A. Kirsch and M. Mitzenmacher. Building a Better Bloom Filter. Harvard University Computer Science Technical Report TR-02-05, 2005. `ftp://ftp.deas.harvard.edu/techreports/tr-02-05.pdf`.

[18] D. Knuth. The Art of Computer Programming, Volume 3: Sorting and Searching. Addison-Wesley, 1973.

[19] G. Lueker and M. Molodowitch. More analysis of double hashing. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing (STOC)*, pp. 354-359, 1988.

[20] M. Mitzenmacher. Compressed Bloom Filters. *IEEE/ACM Transactions on Networking*, 10(5):613-620, 2002.

[21] M. Mitzenmacher and E. Upfal. Probability and Computing: Randomized Algorithms and Probabilistic Analysis. Cambridge University Press, 2005.

[22] R. Motwani and P. Raghavan. Randomized Algorithms. Cambridge University Press, 1995.

[23] A. Pagh, R. Pagh, and S. S. Rao. An Optimal Bloom Filter Replacement. In *Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 823-829, 2005.

[24] M. V. Ramakrishna. Practical performance of Bloom filters and parallel free-text searching. *Communications of the ACM*, 32(10):1237-1239, 1989.

[25] J. P. Schmidt and A. Siegel. The analysis of closed hashing under limited randomness. In *Proceedings of the Twenty-Second Annual ACM Symposium on Theory of Computing (STOC)*, pp. 224-234, 1990.