2015 IEEE 56th Annual Symposium on Foundations of Computer Science

Hashing for statistics over k-partitions

Søren Dahlgaard, Mathias Bæk Tejs Knudsen, Eva Rotenberg, and Mikkel Thorup
Department of Computer Science
University of Copenhagen
{soerend,knudsen,roden,mthorup}@di.ku.dk

Abstract

In this paper we analyze a hash function for k-partitioning a set into bins, obtaining strong concentration bounds for standard algorithms combining statistics from each bin.

This generic method was originally introduced by Flajolet and Martin [FOCS'83] in order to save a factor $\Omega(k)$ of time per element over k independent samples when estimating the number of distinct elements in a data stream. It was also used in the widely used HyperLogLog algorithm of Flajolet et al. [AOFA'97] and in large-scale machine learning by Li et al. [NIPS'12] for minwise estimation of set similarity.

The main issue of k-partition, is that the contents of different bins may be highly correlated when using popular hash functions. This means that methods of analyzing the marginal distribution for a single bin do not apply. Here we show that a tabulation based hash function, mixed tabulation, does yield strong concentration bounds on the most popular applications of k-partitioning similar to those we would get using a truly random hash function. The analysis is very involved and implies several new results of independent interest for both simple and double tabulation, e.g. a simple and efficient construction for invertible bloom filters and uniform hashing on a given set.

Keywords

Hashing; k-partition; tabulation hashing; uniform hashing; concentration bounds; joint distributions; constant moments; peelability; invertible bloom filters.

I. Introduction

A useful assumption in the design of randomized algorithms and data structures is the free availability of fully random hash functions which can be computed in unit time. Removing this unrealistic assumption is the subject of a large body of work. To implement a hash-based algorithm, a concrete hash function has to be chosen. The space, time, and random choices made by this hash function affects the overall performance. *The generic goal is therefore to provide efficient constructions of hash functions that for important randomized algorithms yield probabilistic guarantees similar to those obtained assuming fully random hashing.*

To fully appreciate the significance of this program, we note that many randomized algorithms are very simple and popular in practice, but often they are implemented with too simple hash functions without the necessary guarantees. This may work very well in random tests, adding to their popularity, but the real world is full of structured data that could be bad for the hash function. This was illustrated in [1] showing how simple common inputs made linear probing fail with popular hash functions, explaining its perceived unreliability in practice. The problems disappeared when sufficiently strong hash functions were used.

In this paper, we consider the generic approach where a hash function is used to k-partition a set into bins. Statistics are computed on each bin, and then all these statistics are combined so as to get good concentration bounds. This approach was introduced by Flajolet and Martin [2] under the name *stochastic averaging* to estimate the number of distinct elements in a data stream. Today, a more popular estimator of this quantity is the HyperLogLog counter, which is also based on k-partitioning [3], [4]. These types of counters have found many applications, e.g., to estimate the neighbourhood function of a graph with all-distance sketches [5], [6].

Later it was considered by Li et al. [7], [8], [9] in the classic minwise hashing framework of Broder et al. for the very different application of set similarity estimation [10], [11], [12]. To our knowledge we are the first to address such statistics over a k-partitioning with practical hash functions.

We will use the example of MinHash for frequency estimation as a running example throughout the paper: suppose we have a fully random hash function applied to a set X of red and blue balls. We want to estimate the fraction f of red balls. The idea of the MinHash algorithm is to sample the ball with the smallest hash value. With a fully-random hash function, this is a uniformly random sample from X, and it is red with probability f. For better concentration, we may use k *independent repetitions*: we repeat the experiment k times with k independent hash functions. This yields a multiset S of k samples with replacement from X. The fraction of red balls in S concentrates around f and the error probability falls exponentially in k.

0272-5428/15 $31.00 © 2015 IEEE
DOI 10.1109/FOCS.2015.83

Consider now the alternative experiment based on *k-partitioning*: we use a single hash function, where the first $\lceil \lg k \rceil$ bits of the hash value partition X into k bins, and then the remaining bits are used as a *local hash value* within the bin. We pick the ball with the smallest (local) hash value in each bin. This is a sample S from X without replacement, and again, the fraction of red balls in the non-empty bins is concentrated around f with exponential concentration bounds. We note that there are some differences. We do get the advantage that the samples are without replacement, which means better concentration. On the other hand, we may end up with fewer samples if some bins are empty.

The big difference between the two schemes is that the second one runs $\Omega(k)$ times faster. In the first experiment, each ball participated in k independent experiments, but in the second one with k-partitioning, each ball picks its bin, and then only participates in the local experiment for that bin. Thus, essentially, we get k experiments for the price of one. Handling each ball, or key, in constant time is important in applications of high volume streams.

In this paper, we present the first realistic hash function for k-partitioning in these application. Thus we will get concentration bounds similar to those obtained with fully random hashing for the following algorithms:

- Frequency/similarity estimation as in our running example and as it is used for the machine learning in [7], [8], [9].
- Estimating distinct elements as in [2], [3].

Other technical developments include simpler hash functions for invertible Bloom filters, uniform hashing, and constant moment bounds.

For completeness we mention that the count sketch data structure of Charikar et al. [13] is also based on k-partitioning. However, for count sketches we can never hope for the kind of strong concentration bounds pursued in this paper as they are prevented by the presence of large weight items. The analysis in [13] is just based on variance for which 2-independent hashing suffices. Strong concentration bounds are instead obtained by independent repetitions.

A. Applications in linear machine learning

As mentioned, our running example with red and blue balls is mathematically equivalent to the classic application of minwise hashing to estimate the Jaccard similarity $J(X,Y) = |X \cap Y|/|X \cup Y|$ between two sets X and Y. This method was originally introduced by Broder et al. [10], [11], [12] for the AltaVista search engine. The red balls correspond to the intersection of X and Y and the blue balls correspond to the symmetric difference. The MinHash estimator is the indicator variable of whether the ball with the smallest hash value over both sets belongs to the intersection of the two sets. To determine this we store the smallest hash value from each set as a *sketch* and check if it is the same. In order to reduce the variance one uses k independent hash functions, known as $k\times$minwise. This method was later revisited by Li et al. [14], [15], [16]. By only using the b least significant bits of each hash value (for some small constant b), they were able to create efficient linear sketches, encoding set-similarity as inner products of sketch vectors, for use in large-scale learning. However, applying k hash functions to each element increases the sketch creation time by roughly a factor of k.

It should be noted that Bachrach and Porat [17] have suggested a more efficient way of maintaining k Minhash values with k different hash functions. They use k different polynomial hash functions that are related, yet pairwise independent, so that they can systematically maintain the Minhash for all k polynomials in $O(\log k)$ time per key assuming constant degree polynomials. There are two issues with this approach: It is specialized to work with polynomials and Minhash is known to have constant bias with constant degree polynomials [29], and this bias does not decay with independent repetitions. Also, because the experiments are only pairwise independent, the concentration is only limited by Chebyshev's inequality.

An alternative to $k\times$minwise when estimating set similarity with minwise sketches is bottom-k. In bottom-k we use one hash function and maintain the sketch as the keys with the k smallest hash values. This method can be viewed as sampling without replacement. Bottom-k has been proved to work with simple hash functions both for counting distinct elements [18] and for set similarity [19]. However, it needs a priority queue to maintain the k smallest hash values and this leads to a non-constant worst-case time per element, which may be a problem in real-time processing of high volume data streams. A major problem in our context is that we are not able to encode set-similarity as an inner product of two sketch vectors. This is because the elements lose their "alignment" – that is, the key with the smallest hash value in one set might have the 10th smallest hash value in another set.

Getting down to constant time per element via k-partitioning was suggested by Li et al. [7], [8], [9]. They use k-partitioned MinHash to quickly create small sketches of very high-dimensional indicator vectors. Each sketch consists of k hash values corresponding to the hash value of each bin. The sketches are then converted into sketch vectors that code similarity as inner products. Finally the sketch vectors are fed to a linear SVM for classifying massive data sets. The sketches are illustrated in Figure 1. Li et al. also apply this approach to near neighbour search using locality sensitive hashing as introduced in [20] (see also [21], [22]). When viewing the problems as red and blue balls, the canonical unbiased estimator uses the number $k^\star$ of non-empty bins, estimating f as:

$$\frac{\#\text{ of red bins}}{k^\star} \tag{1}$$

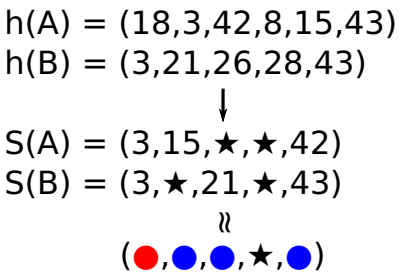

Figure 1. Example of k-partitioned sketches for two sets A and B. The hash values are from the set $\{0,\dots,49\}$ and $k=5$. The sketches $S(A)$ and $S(B)$ show the hash values for each bin and the $\star$ symbol denotes an empty bin. The corresponding interpretation as red and blue balls is shown below with a red ball belonging to the intersection and blue ball to the symmetric difference. Here $k^\star = 4$.

A major complication of this estimator is that we do not know in advance, which bins are jointly empty for two sketches (as illustrated in Figure 1). This means that there is no natural way of computing the estimator as an inner product of the two sketches. Shrivastava and Li [8], [9] suggest methods for dealing with this by assigning empty bins a value by copying from non-empty bins in different ways giving provably good bounds. It is important to note that when all bins are non-empty, all the estimators considered in [7], [8], [9] are identical to the estimator in (1) as $k^\star = k$ in this case.

We note that for our running example with red and blue balls, it would suffice to generate hash values on the fly, e.g., using a pseudo-random number generator, but in the real application of set similarity, it is crucial that when sets get processed separately, the elements from the intersection get the same hash value. Likewise, when we want to estimate distinct elements or do count sketches, it is crucial that the same element always gets the same hash value.

B. Technical challenge

Using a hash function function to k-partition n keys is often cast as using it to throw n balls into k bins, which is an important topic in randomized algorithms [23, Chapter 3] [24, Chapter 5]. However, when it comes to implementation via realistic hash functions, the focus is often only on the marginal distribution in a single bin. For example, with $k=n$, w.h.p., a given bin has load $O(\log n/\log\log n)$, hence by a union bound, w.h.p., the maximum load is $O(\log n/\log\log n)$. The high probability bound on the load of a given bin follows with an $O(\log n/\log\log n)$-independent hash function [25], but can also be obtained in other ways [26], [27].

However, the whole point in using a k-partition is that we want to average statistics over all k bins hoping to get strong concentration bounds, but this requires that the statistics from the k bins are not too correlated (even with full randomness, there is always some correlation since the partitioning corresponds to sampling without replacement, but this generally works in our favor).

To be more concrete, consider our running example with red and blue balls where Minhash is used to pick a random ball from each bin. The frequency f of red balls is estimated as the frequency of red balls in the sample. Using $O(\log k)$ independent hashing, we can make sure that the bias in the sample from any given bin is $1/k$ [28]. However, for concentration bounds on the average, we have to worry about two types of correlations between statistics of different bins. The first "local" correlation issue is if the local hashing within different bins is too correlated. This issue could conceivably be circumvented using one hash function for the k-partitioning itself, and then have an independent local hash function for each bin. The other "global" correlation issue is for the overall k-partitioning distribution between bins. It could be that if we get a lot of red balls in one bin, then this would be part of a general clustering of the red balls on a few bins (examples showing how such systematic clustering can happen with simple hash functions are given in [29]). This clustering would disfavor the red balls in the overall average even if the sampling in each bin was uniform and independent. This is an issue of non-linearity, e.g., if there are already more red than blue balls in a bin, then doubling their number only increases their frequency by at most $3/2$. As mentioned earlier we are not aware of any previous work addressing these issues with a less than fully random hash function, but for our running example it appears that a $O(k\log k)$-independent hash function will take care of both correlation issues (we will not prove this as we are going to present an even better solution).

Resource consumption: We are now going to consider the resource consumption by the different hashing schemes discussed above. The schemes discussed are summarized in Table I.

First we assume that the key universe is of size polynomial in the number n of keys. If not, we first do a standard universe reduction, applying a universal hash function [32] into an intermediate universe of size $u = n^{O(1)}$, expecting no collisions. We could now, in principle, have a fully random hash function over $[u]$.

Technique	Evaluation time	Space (words)
Fully random hashing	$O(1)$	$u = n^{O(1)}$
Fully random on n keys whp. [30]	$O(1)$	$(1 + o(1))n$
$\tilde{O}(k)$-independence [31]	$O(1)$	ku^{ε}
Mixed tabulation (this paper)	$O(1)$	$\tilde{O}(k) + u^{\varepsilon}$

Table I
RESOURCES OF HASHING TECHNIQUES. HERE, ε MAY BE CHOSEN AS AN ARBITRARILY SMALL POSITIVE CONSTANT.

We can get down to linear space using the construction of Pagh and Pagh (PP) [30]. Their hash function uses $O(n)$ words and is, w.h.p., fully random on any given set of n keys. However, using $O(n)$ space is still prohibitive in most applications as the main motivation of k-partitioning is exactly to create an estimator of size k when n is so big that we cannot store the set. Additionally, we may not know n in advance.

As indicated above, it appears that $\Theta(k \log k)$-independent hashing suffices for MinHash. For this we can use the recent construction of Christiani et al. [31]. Their construction gets $\Theta(k \log k)$-independence, w.h.p., in $O(1)$ time using space ku^{ε} for an arbitrarily small constant ε affecting the evaluation time. Interestingly, we use the same space if we want a $\Theta(\log k)$-independent hash function for each of the k bins. The construction of Thorup [33] gives independence $u^{\varepsilon} \gg \log k$ in $O(1)$ time using u^{ε} space. A lower bound of Siegel [34] shows that we cannot hope to improve the space in either case if we want fast hashing. More precisely, if we want q-independence in time $t < q$, we need space at least $q(u/q)^{1/t}$. Space $ku^{\Omega(1)}$ thus appears to be the best we can hope for with these independence based approaches.

C. k-partitions via mixed tabulation

In this paper we present and analyze a hash function, *mixed tabulation*, that for all the k-partitioning algorithms discussed above, w.h.p., gets concentration similar to that with fully random hash functions. The hashing is done in $O(1)$ time and $\tilde{O}(k) + u^{\varepsilon}$ space. If, say, $k = u^{\Omega(1)}$, this means that we hash in constant time using space near-linear in the number of counters. This is the first proposals of a hash function for statistics over k-partitions that has good theoretical probabilistic properties, yet does not significantly increase the amount of resources used by these popular algorithms. The hash function we suggest for k-partitioning, *mixed tabulation*, is an extension of simple tabulation hashing.

Simple tabulation: Simple tabulation hashing dates back to Zobrist [35]. The hash family takes an integer parameter $c > 1$, and we view a key $x \in [u] = \{0, \ldots, u-1\}$ as a vector of c characters $x_0, \ldots, x_{c-1} \in \Sigma = [u^{1/c}]$. The hash values are bit strings of some length r. For each character position i, we initialize a fully random table T_i of size $|\Sigma|$ with values from $\mathcal{R} = [2^r]$. The hash value of a key x is calculated as

$$h(x) = T_0[x_0] \oplus \cdots \oplus T_{c-1}[x_{c-1}] \ .$$

Simple tabulation thus takes time $O(c)$ and space $O(cu^{1/c})$. In our context we assume that c is a constant and that the character tables fit in fast cache (eg. for 64-bit keys we may pick $c = 4$ and have 16-bit characters. The tables T_i then take up 2^{16} words). Justifying this assumption, recall that with universe reduction, we can assume that the universe is of size $u = n^{O(1)}$. Now, for any desired constant $\varepsilon > 0$, we can pick $c = O(1)$ such that $\Sigma = u^{1/c} \leq n^{\varepsilon}$. We refer to the lookups $T_i[x_i]$ as *character lookups* to emphasize that we expect them to be much faster than a general lookups in memory. Pătraşcu and Thorup [27] found simple tabulation to be 3 times faster than evaluating a degree-2 polynomial over a prime field for the same key domain.

Pătraşcu and Thorup [27] analyzed simple tabulation assuming $c = O(1)$, showing that it works very well for common applications of hash function such as linear probing, cuckoo hashing and minwise independence. Note, however, that $O(\log n)$ independence was known to suffice for all these applications. We also note that simple tabulation fails to give good concentration for k-partitions: Consider the set $R = [2] \times [m/2]$ of m red balls and let B be some random set of blue balls. In this case the red balls hash into the same buckets in pairs with probability $1/k$, which will skew the estimate by a factor of 2 if, for instance, $|R|$ is relatively small.

Mixed tabulation: To handle k-partitions, we here propose and analyze a mix between simple tabulation defined above and the double tabulation scheme of [33]. In addition to c, mixed tabulation takes as a parameter an integer $d \geq 1$. We derive d extra characters using one simple tabulation function and compose these with the original key before applying an extra round of simple tabulation. Mathematically, we use two simple tabulation hash functions $h_1 : \Sigma^c \to \Sigma^d$ and $h_2 : \Sigma^{d+c} \to \mathcal{R}$ and define the hash function to be $h(x) \mapsto h_2(x \cdot h_1(x))$, where $\cdot$ denotes concatenation of characters. We call $x \cdot h_1(x)$ the *derived key* and denote this by $h_1^{\star}(x)$. Our mixed tabulation scheme is very similar to Thorup's

double tabulation [33] and we shall return to the relation in Section I-D. We note that we can implement this using just $c+d$ lookups if we instead store simple tabulation functions $h_{1,2} : \Sigma^c \to \Sigma^d \times \mathcal{R}$ and $h'_2 : \Sigma^d \to \mathcal{R}$, computing $h(x)$ by $(v_1, v_2) = h_{1,2}(x);\ h(x) = v_1 \oplus h'_2(v_2)$. This efficient implementation is similar to that of twisted tabulation [36], and is equivalent to the previous definition. In our applications, we think of c and d as a small constants, e.g. $c = 4$ and $d = 4$. We note that we need not choose Σ such that $|\Sigma|^c = u$. Instead, we may pick $|\Sigma| \geq u^{1/c}$ to be any power of two. A key x is divided into c characters x_i of $b = \lceil \lg u^{1/c} \rceil$ or $b-1$ bits, so $x_i \in [2^b] \subseteq \Sigma$. This gives us the freedom to use c such that $u^{1/c}$ is not a power of two, but it also allows us to work with $|\Sigma| \gg u^{1/c}$, which in effect means that the derived characters are picked from a larger domain than the original characters. Then mixed tabulation uses $O(c+d)$ time and $O(cu^{1/c}+d|\Sigma|)$ space. For a good balance, we will always pick c and $|\Sigma|$ such that $u^{1/c} \leq |\Sigma| \leq u^{1/(c-1)}$. In all our applications we have $c = O(1), d = O(1)$, which implies that the evaluation time is constant and that the space used is $\Theta(|\Sigma|)$.

Mixed tabulation in MinHash with k-partitioning: We will now analyze MinHash with k-partitioning using mixed tabulation as a hash function, showing that we get concentration bounds similar to those obtained with fully-random hashing. The analysis is based on two complimentary theorems. The first theorem states that for sets of size nearly up to $|\Sigma|$, mixed tabulation is fully random with high probability.

Theorem 1. *Let h be a mixed tabulation hash function with parameter d and let $X \subseteq [u]$ be any input set. If $|X| \leq |\Sigma|/(1+\Omega(1))$ then the keys of X hash independently with probability $1 - O(|\Sigma|^{1-\lfloor d/2 \rfloor})$.*

The second theorem will be used to analyze the performance for larger sets. It is specific to MinHash with k-partitioning, stating, w.h.p., that mixed tabulation hashing performs as well as fully random hashing with slight changes to the number of balls:

Theorem 2. *Consider a set of n_R red balls and n_B blue balls with $n_R + n_B > |\Sigma|/2$. Let $f = n_R/(n_R+n_B)$ be the fraction of red balls which we wish to estimate.*

Let $X^{\mathcal{M}}$ be the estimator of f from (1) *that we get using MinHash with k-partitioning using mixed tabulation hashing with d derived characters, where $k \leq |\Sigma|/(4d \log |\Sigma|)$.*

Let $\overline{X}^{\mathcal{R}}$ be the same estimator in the alternative experiment where we use fully random hashing but with $\lfloor n_R(1+\varepsilon) \rfloor$ red balls and $\lceil n_B(1-\varepsilon) \rceil$ blue balls where $\varepsilon = O\Big(\sqrt{\frac{\log |\Sigma| (\log\log |\Sigma|)^2}{|\Sigma|}}\Big)$. Then

$$\Pr\big[X^{\mathcal{M}} \geq (1+\delta)f\big] \leq \Pr\Big[\overline{X}^{\mathcal{R}} \geq (1+\delta)f\Big] + \tilde{O}\Big(|\Sigma|^{1-\lfloor d/2 \rfloor}\Big).$$

Likewise, for a lower bound, let $\underline{X}^{\mathcal{R}}$ be the estimator in the experiment using fully random hashing but with $\lceil n_R(1-\varepsilon) \rceil$ red balls and $\lfloor n_B(1+\varepsilon) \rfloor$ blue balls. Then

$$\Pr\big[X^{\mathcal{M}} \leq (1-\delta)f\big] \leq \Pr\big[\underline{X}^{\mathcal{R}} \leq (1-\delta)f\big] + \tilde{O}\Big(|\Sigma|^{1-\lfloor d/2 \rfloor}\Big).$$

To apply the above theorems, we pick our parameters k and Σ such that

$$k \leq \min\left\{ \frac{|\Sigma|}{\log|\Sigma| \, (\log\log|\Sigma|)^2}, \frac{|\Sigma|}{4d \log|\Sigma|} \right\} \tag{2}$$

Recall that we have the additional constraint that $|\Sigma| \geq u^{1/c}$ for some $c = O(1)$. Thus (2) is only relevant if want to partition into $k = u^{\Omega(1)}$ bins. It forces us to use space $\Theta(|\Sigma|) = \Omega(k \log k (\log\log k)^2)$.

With this setting of parameters, we run MinHash with k-partitioning over a given input. Let n_R and n_B be the number of red and blue balls, respectively. Our analysis will hold no matter which of the estimators from [7], [8], [9] we apply.

If $n_R + n_B \leq |\Sigma|/2$, we refer to Theorem 1. It implies that no matter which of estimators from [7], [8], [9] we apply, we can refer directly to the analysis done in [7], [8], [9] assuming fully random hashing. All we have to do is to add an extra error probability of $O(|\Sigma|^{1-\lfloor d/2 \rfloor})$.

Assume now that $n_R + n_B \geq |\Sigma|/2$. First we note that all bins are non-empty w.h.p. To see this, we only consider the first $|\Sigma|/2 \geq 2dk \log|\Sigma|$ balls. By Theorem 1, they hash fully randomly with probability $1 - O(|\Sigma|^{1-\lfloor d/2 \rfloor})$, and if so, the probability that some bin is empty is bounded by $k(1-1/k)^{2dk\log|\Sigma|} < k/|\Sigma|^{2d}$. Thus, all bins are non-empty with probability $1 - O(|\Sigma|^{1-\lfloor d/2 \rfloor})$.

Assuming that all bins are non-empty, all the estimators from [7], [8], [9] are identical to (1). This means that Theorem 2 applies no matter which of the estimators we use since the error probability $\tilde{O}\big(|\Sigma|^{1-\lfloor d/2 \rfloor}\big)$ absorbs the probability that some bin is empty. In addition, the first bound in (2) implies that $\varepsilon = O(1/\sqrt{k})$ (which is reduced to $o(1/\sqrt{k})$ if $\Sigma = \omega(k \log k (\log\log k)^2)$). In principle this completes the description of how close mixed tabulation brings us in performance to fully random hashing.

To appreciate the impact of ε, we first consider what guarantees we can give with fully random hashing. We are still assuming $n_R + n_B \geq |\Sigma|/2$ where $|\Sigma| \geq 4dk \log |\Sigma|$ as implied by (2), so the probability of an empty bin is bounded by $k(1-1/k)^{|\Sigma|/2} < |\Sigma|^{1-2d}$. Assume that all bins are non-emtpy, and let $f = n_R/(n_R+n_B)$ be the fraction of red balls. Then our estimator $X^{\mathcal{R}}$ of f is the fraction of red balls among k samples without replacement. In expectation we get fk red balls. For $\delta \leq 1$, the probability that the number of red balls deviates by more than δfk from fk is $2\exp(\Omega(\delta^2 fk))$. This follows from a standard application of Chernoff bounds without replacement [37]. The probability of a relative error $\left|X^{\mathcal{R}} - f\right|/f \geq t/\sqrt{fk}$ is thus bounded by $2e^{-\Omega(t^2)}$ for any $t \leq \sqrt{fk}$.

The point now is that $\varepsilon = O(1/\sqrt{k}) = O(1/\sqrt{fk})$. In the fully random experiments in Theorem 2, we replace n_R by $n'_R = (1\pm\varepsilon)n_R$ and n_B with $n'_B = (1\pm\varepsilon)n_B$. Then $X^{\mathcal{R}}$ estimates $f' = n'_R/(n'_R+n'_B) = (1\pm\varepsilon)f$, so we have $\Pr[\left|X^{\mathcal{R}} - f'\right|/f' \geq t/\sqrt{f'k}] \leq 2e^{-\Omega(t^2)}$. However, since $\varepsilon = O(1/\sqrt{k})$, this implies $\Pr[\left|X^{\mathcal{R}} - f\right|/f \geq t/\sqrt{fk}] \leq 2e^{-\Omega(t^2)}$ for any $t \leq \sqrt{fk}$. The only difference is that Ω hides a smaller constant. Including the probability of getting an empty bin, we get $\Pr[\left|X^{\mathcal{R}} - f\right| \geq t\sqrt{f/k}] \leq 2e^{-\Omega(t^2)} + |\Sigma|^{1-2d}$ for any $t \leq \sqrt{fk}$. Hence, by Theorem 2, $\Pr[\left|X^{\mathcal{M}} - f\right| \geq t\sqrt{f/k}] \leq 2e^{-\Omega(t^2)} + \tilde{O}\big(|\Sigma|^{1-\lfloor d/2\rfloor}\big)$ for any $t \leq \sqrt{fk}$.

Now if $n_B \leq n_R$ and $f \geq 1/2$, it gives better concentration bounds to consider the symmetric estimator $X_B^{\mathcal{M}} = 1 - X^{\mathcal{M}}$ for the fraction $f_B = 1 - f \leq f$ of blue balls. The analysis from above shows that $\Pr[\left|X_B^{\mathcal{M}} - f_B\right| \geq t\sqrt{f_B/k}] \leq 2e^{-\Omega(t^2)} + \tilde{O}\big(|\Sigma|^{1-\lfloor d/2\rfloor}\big)$ for any $t \leq \sqrt{f_B k}$. Here $\left|X_B^{\mathcal{M}} - f_B\right| = \left|X^{\mathcal{M}} - f\right|$, so we conclude that $\Pr[\left|X^{\mathcal{M}} - f\right| \geq t\sqrt{\min\{f,1-f\}/k}] \leq 2e^{-\Omega(t^2)} + \tilde{O}\big(|\Sigma|^{1-\lfloor d/2\rfloor}\big)$ for any $t \leq \sqrt{\min\{f,1-f\}/k}$. Thus we have proved:

Corollary 1. *We consider MinHash with k-partitioning using mixed tabulation with alphabet Σ and $c, d = O(1)$, and where k satisfies* (2)*. Consider a set of n_R and n_B red and blue balls, respectively, where $n_R+n_B > |\Sigma|/2$. Let $f = n_R/(n_R+n_B)$ be the fraction of red balls that we wish to estimate. Let $X^{\mathcal{M}}$ be the estimator of f we get from our MinHash with k-partitioning using mixed tabulation. The estimator may be that in* (1)*, or any of the estimators from [7], [8], [9]. Then for every $0 \leq t \leq \sqrt{\min\{f,1-f\}k}$,*

$$\Pr\Big[\left|X^{\mathcal{M}} - f\right| \geq t\sqrt{\min\{f,1-f\}/k}\Big] \leq 2e^{-\Omega(t^2)} + \tilde{O}\Big(|\Sigma|^{1-\lfloor d/2\rfloor}\Big)) \ .$$

The significance of having errors in terms of $1-f$ is when the fraction of red balls represent similarity as discussed earlier. This gives us much better bounds for the estimation of very similar sets.

The important point above is not so much the exact bounds we get in Corollary 1, but rather the way we translate bounds with fully random hashing to the case of mixed tabulation.

Mixed tabulation in distinct counting with k-partitioning: We can also show that distinct counting with k-partitioning using mixed tabulation as a hash function gives concentration bounds similar to those obtained with fully-random hashing. With less than $|\Sigma|/2$ balls, we just apply Theorem 1, stating that mixed tabulation is fully random with high probability. With more balls, we use the following analogue to Theorem 2:

Theorem 3. *Consider a set of $n > |\Sigma|/2$ balls. Let $X^{\mathcal{M}}$ be the estimator of n using either stochastic averaging [2] or HyperLogLog [3] over a k-partition with mixed tabulation hashing where $k \leq |\Sigma|/(4d\log|\Sigma|)$. Let $\overline{X}^{\mathcal{R}}$ be the same estimator in the alternative experiment where we use fully random hashing but with $\lfloor n(1+\varepsilon)\rfloor$ balls where $\varepsilon = O\Big(\sqrt{\frac{\log|\Sigma|(\log\log|\Sigma|)^2}{|\Sigma|}}\Big)$. Then*

$$\Pr\big[X^{\mathcal{M}} \geq (1+\delta)n\big] \leq \Pr\Big[\overline{X}^{\mathcal{R}} \geq (1+\delta)n\Big] + \tilde{O}\Big(|\Sigma|^{1-\lfloor d/2\rfloor}\Big)\,,$$

Likewise, for a lower bound, let $\underline{X}^{\mathcal{R}}$ be the estimator in the experiment using fully random hashing but with $\lceil n(1-\varepsilon)\rceil$ balls. Then

$$\Pr\big[X^{\mathcal{M}} \leq (1-\delta)n\big] \leq \Pr\big[\underline{X}^{\mathcal{R}} \leq (1-\delta)n\big] + \tilde{O}\Big(|\Sigma|^{1-\lfloor d/2\rfloor}\Big)\,.$$

Conceptually, the proof of Theorem 3 is much simpler than that of Theorem 2 since there are no colors. However, the estimators are harder to describe, leading to a more messy formal proof, which we do not have room for in this conference paper.

D. Techniques and other results

Our analysis of mixed tabulation gives many new insights into both simple and double tabulation. To prove Theorem 2 and Theorem 3, we will show a generalization of Theorem 1 proving that mixed tabulation behaves like a truly random hash function on fairly large sets with high probability, even when some of the output bits of the hash function are known. The exact statement is as follows.

Theorem 4. *Let $h = h_2 \circ h_1^\star$ be a mixed tabulation hash function. Let $X \subseteq [u]$ be any input set. Let $p_1, \ldots, p_b$ be any b bit positions, $v_1, \ldots, v_b \in \{0,1\}$ be desired bit values and let Y be the set of keys $x \in X$ where $h(x)_{p_i} = v_i$ for all i. If $\mathbf{E}[|Y|] = |X| \cdot 2^{-b} \leq |\Sigma|/(1+\Omega(1))$, then the remaining bits of the hash values in Y are completely independent with probability $1 - O(|\Sigma|^{1-\lfloor d/2 \rfloor})$.*

In connection with our k-partition applications, the specified output bits will be used to select a small set of keys that are critical to the final statistics, and for which we have fully random hashing on the remaining bits.

In order to prove Theorem 4 we develop a number of structural lemmas in Section III relating to key dependencies in simple tabulation. These lemmas provides a basis for showing some interesting results for simple tabulation and double tabulation, which we also include in this paper. These results are briefly described below.

Double tabulation and uniform hashing: In double tabulation [33], we compose two independent simple tabulation functions $h_1 : \Sigma^c \to \Sigma^d$ and $h_2 : \Sigma^d \to \mathcal{R}$ defining $h : \Sigma^c \to \mathcal{R}$ as $h(x) = h_2(h_1(x))$. We note that with the same values for c and d, double tabulation is a strict simplification of mixed tabulation in that h_2 is only applied to $h_1(x)$ instead of to $x \cdot h_1(x)$. The advantage of mixed tabulation is that we know that the "derived" keys $x \cdot h_1(x)$ are distinct, and this is crucial to our analysis of k-partitioning. However, if all we want is uniformity over a given set, then we show that the statement of Theorem 1 also holds for double tabulation.

Theorem 5. *Given an arbitrary set $S \subseteq [u]$ of size $|\Sigma|/(1+\Omega(1))$, with probability $1 - O(|\Sigma|^{1-\lfloor d/2 \rfloor})$ over the choice of h_1, the double tabulation function $h_2 \circ h_1$ is fully random over S.*

Theorem 5 should be contrasted by the main theorem from [33]:

Theorem 6 (Thorup [33])**.** *If $d \geq 6c$, then with probability $1 - o(|\Sigma|^{2-d/(2c)})$ over the choice of h_1, the double tabulation function $h_2 \circ h_1$ is $k = |\Sigma|^{1/(5c)}$-independent.*

The contrast here is, informally, that Theorem 5 is a statement about any one large set, Theorem 6 holds for all small sets. Also, Theorem 5 with $d = 4$ "derived" characters gets essentially the same error probability as Theorem 6 with $d = 6c$. Of course, with $d = 6c$, we are likely to get both properties with the same double tabulation function.

Siegel [34] has proved that with space $|\Sigma|$ it is impossible to get independence higher than $|\Sigma|^{1-\Omega(1)}$ with constant time evaluation. This is much less than the size of S in Theorem 5.

Theorem 5 provides an extremely simple $O(n)$ space implementation of a constant time hash function that is likely uniform on any given set S of size n. This should be compared with the corresponding linear space uniform hashing of Pagh and Pagh [30, §3]. Their original implementation used Siegel's [34] highly independent hash function as a subroutine. Dietzfelbinger and Woelfel [38] found a simpler subroutine that was not highly independent, but still worked in the uniform hashing from [30]. However, Thorup's highly independent double tabulation from Theorem 6 is even simpler, providing us the simplest known implementation of the uniform hashing in [30]. However, as discussed earlier, double tabulation uses many more derived characters for high independence than for uniformity on a given set, so for linear space uniform hashing on a given set, it is much faster and simpler to use the double tabulation of Theorem 5 directly rather than [30, §3]. We note that [30, §4] presents a general trick to reduce the space from $O(n(\lg n + \lg|\mathcal{R}|))$ bits downto $(1+\varepsilon)n\lg|\mathcal{R}| + O(n)$ bits, preserving the constant evaluation time. This reduction can also be applied to Theorem 5 so that we also get a simpler overall construction for a succinct dictionary using $(1+\varepsilon)n\lg|\mathcal{R}| + O(n)$ bits of space and constant evaluation time.

We note that our analysis of Theorem 4 does not apply to Pagh and Pagh's construction in [30], without strong assumptions on the hash functions used, as we rely heavily on the independence of output bits provided by simple tabulation.

Peelable hash functions and invertible bloom filters: Our proof of Theorem 5 uses Thorup's variant [33] of Siegel's notion of peelability [34]. The hash function h_1 is a fully peelable map of S if for every subset $Y \subseteq S$ there exists a key $y \in Y$ such that $h_1(y)$ has a unique output character. If h_1 is peelable over S and h_2 is a random simple tabulation hash function, then $h_2 \circ h_1$ is a uniform hash function over S. Theorem 5 thus follows by proving the following theorem.

Theorem 7. *Let $h : \Sigma^c \to \Sigma^d$ be a simple tabulation hash function and let X be a set of keys with $|X| \leq |\Sigma|/(1+\Omega(1))$. Then h is fully peelable on X with probability $1 - O(|\Sigma|^{1-\lfloor d/2 \rfloor})$.*

The peelability of h is not only relevant for uniform hashing. This property is also critical for the hash function in Goodrich and Mitzenmacher's Invertible Bloom Filters [39], which have found numerous applications in streaming and data bases [40], [41], [42]. So far Invertible Bloom Filters have been implemented with fully random hashing, but Theorem 7 states that simple tabulation suffices for the underlying hash function.

Constant moments: An alternative to Chernoff bounds in providing good concentration is to use bounded moments. We show that the kth moment of simple tabulation comes within a constant factor of that achieved by truly random hash functions for any constant k.

Theorem 8. *Let $h : [u] \to \mathcal{R}$ be a simple tabulation hash function. Let $x_0, \ldots, x_{m-1}$ be m distinct keys from $[u]$ and let $Y_0, \ldots, Y_{m-1}$ be any random variables such that $Y_i \in [0,1]$ is a function of $h(x_i)$ with mean $\mathbf{E}[Y_i] = p$ for all $i \in [m]$. Define $Y = \sum_{i\in[m]} Y_i$ and $\mu = \mathbf{E}[Y] = mp$. Then for any constant integer $k \geq 1$:*

$$\mathbf{E}\big[(Y-\mu)^{2k}\big] = O\left(\sum_{j=1}^{k} \mu^j\right) \ ,$$

where the constant in the O-notation is dependent on k and c.

E. Notation

Let $S \subseteq [u]$ be a set of keys. Denote by $\pi(S,i)$ the projection of S on the ith character, i.e. $\pi(S,i) = \{x_i | x \in S\}$. We also use this notation for keys, so $\pi((x_0,\ldots,x_{c-1}),i) = x_i$. A *position character* is an element of $[c] \times \Sigma$. Under this definition a key $x \in [u]$ can be viewed as a set of c position characters $\{(0,x_0),\ldots,(c-1,x_{c-1})\}$. Furthermore, for simple tabulation, we assume that h is defined on position characters as $h((i,\alpha)) = T_i[\alpha]$. This definition extends to sets of position characters in a natural way by taking the XOR over the hash of each position character. We denote the symmetric difference of the position characters of a set of keys $x_1,\ldots,x_k$ by

$$\bigoplus_{i=1}^{k} x_k \ .$$

We say that a set of keys $x_1,\ldots,x_k$ are *independent* if their corresponding hash values are independent. If the keys are not independent we say that they are *dependent*.

The *hash graph* of hash functions $h_1 : [u] \to \mathcal{R}_1, \ldots, h_k : [u] \to \mathcal{R}_k$ and a set $S \subseteq [u]$ is the graph in which each element of $\mathcal{R}_1 \cup \ldots \cup \mathcal{R}_k$ is a node, and the nodes are connected by the (hyper-)edges $(h_1(x),\ldots,h_k(x)), x \in S$. In the graph there is a one-to-one correspondence between keys and edges, so we will not distinguish between those.

F. Contents

The paper is structured as follows. In Section II we show how Theorem 4 can be used to prove Theorem 2 noting that the same argument can be used to prove Theorem 3. Sections III to V detail the proof of Theorem 4, which is the main technical part of the paper. Finally In Section VI we prove Theorem 8.

II. MinHash with mixed tabulation

In this section we prove Theorem 2. Theorem 3 can be proved using the same method. We will use the following lemma, which is proved at the end of this section.

Lemma 1. *Let h be a mixed tabulation hash function, $X \subset [u]$, and Y defined as in Theorem 4 such that $\mathbf{E}[|Y|] \in \left[\frac{|\Sigma|}{8}, \frac{|\Sigma|}{4}\right)$. Then with probability $1 - \tilde{O}\left(|\Sigma|^{1-\lfloor d/2 \rfloor}\right)$*

$$|Y| \in \mathbf{E}[|Y|] \cdot \left(1 \pm O\left(\sqrt{\frac{\log|\Sigma| \cdot (\log\log|\Sigma|)^2}{|\Sigma|}}\right)\right)$$

We are given sets R and B of n_R and n_B red and blue balls respectively. Recall that the hash value $h(x)$ of a key x is split into two parts: one telling which of the k bins x lands in (i.e. the first $\lceil \lg k \rceil$ bits) and the *local hash value* in $[0,1)$ (the rest of the bits).

Recall that $|R| + |B| > |\Sigma|/2$ and assume that $|B| \geq |R|$, as the other case is symmetric. For $C = R, B$, we define the set S_C to be the keys in C, for which the first ℓ_C bits of the local hash value are 0. We pick ℓ_C such that

$$\mathbf{E}[|S_C|] = 2^{-\ell_C}|C| \in \left(\frac{|\Sigma|}{8}, \frac{|\Sigma|}{4}\right] \ .$$

This is illustrated in Figure 2. We also define X to be the keys of R and B whose first ℓ_B bits of the local hash value are 0.

We only bound the probability $P = \Pr\big[X^{\mathcal{M}} \geq (1+\delta)f\big]$ and note that we can bound $\Pr\big[X^{\mathcal{M}} \leq (1-\delta)f\big]$ similarly. Consider also the alternative experiment $\overline{X}^{\mathcal{R}}$ as defined in the theorem. We let $\varepsilon = c_0 \cdot \sqrt{\frac{\log|\Sigma| \log\log|\Sigma|}{|\Sigma|}}$ for some large

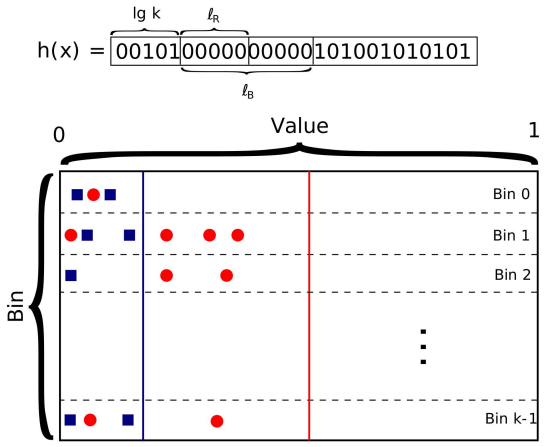

Figure 2. Illustration of the analysis for minwise hashing with mixed tabulation. Since there are more red than blue balls, ℓ_R is smaller than ℓ_B, illustrated by the blue vertical line being before the red one.

enough constant c_0. The set of $\lfloor(1+\varepsilon)\,|R|\rfloor$ and $\lceil(1-\varepsilon)\,|B|\rceil$ balls in this experiment is denoted R' and B' respectively. We define S'_R and S'_B to be the keys from R' and B' where the first ℓ_R and ℓ_B bits of the hash values are 0 respectively.

In order to do bound P we consider the following five *bad events*:

E_1: $|S_R| > |S'_R|$.
E_2: The remaining $\lg|\mathcal{R}| - \ell_R$ output bits are fully independent when restricting h to the keys of S_R.
E_3: $|S_B| < |S'_B|$.
E_4: The remaining $\lg|\mathcal{R}| - \ell_B$ output bits are fully independent when restricting h to the keys of X.
E_5: There exists a bin which contains no key from X.

We will show that $\Pr[E_i] = \tilde{O}\big(|\Sigma|^{1-\lfloor d/2\rfloor}\big)$ for $i = 1,\ldots,5$. For $i = 2,4$ this is an immediate consequence of Theorem 4. For $i = 1,3$ we use Lemma 1 and let c_0 be sufficiently large. For $i = 5$ we see that if E_3 and E_4 do not occur then the probability that there exist a bin with no balls from X is at most:

$$k\cdot\left(1-\frac{1}{k}\right)^{|\Sigma|/8\cdot(1-\varepsilon)} \leq k\cdot\exp\left(-\frac{|\Sigma|}{8k}(1-\varepsilon)\right) \leq k\cdot\exp\left(-\frac{d\log|\Sigma|}{2}(1-\varepsilon)\right) \leq O\Big(|\Sigma|^{1-d/2}\Big)$$

Hence by a union bound $\Pr[E_1\cup\ldots\cup E_5] = \tilde{O}\big(|\Sigma|^{1-\lfloor d/2\rfloor}\big)$ and:

$$P \leq \Pr\big[X^{\mathcal{M}} \geq (1+\delta)f \cap \neg E_1\cap\ldots\cap\neg E_5\big] + \tilde{O}\Big(|\Sigma|^{1-\lfloor d/2\rfloor}\Big) \tag{3}$$

Fix the ℓ_R bits of the hash values that decide S_R, S'_R such that $|S_R| = a, |S'_R| = a'$ and consider the probabilities

$$P_1 = \Pr\big[X^{\mathcal{M}} \geq (1+\delta)f \cap \neg E_1\cap\ldots\cap\neg E_5 \mid (|S_R| = a, |S'_R| = a')\big]$$
$$P_2 = \Pr\Big[\overline{X}^{\mathcal{R}} \geq (1+\delta)f \mid (|S_R| = a, |S'_R| = a')\Big]$$

We will now prove that $P_1 \leq P_2$. This is trivial when $a > a'$ since $P_1 = 0$ in this case so assume that $a \leq a'$. We define X' analogously to X and let $Y = X\cap S_R, Y' = X'\cap S'_R$. Now fix the joint distribution of (Y,Y') such that either E_2 or $|Y|\leq|Y'|$ with probability 1. We can do this without changing the marginal distributions of Y, Y' since if E_2 doesn't occur the probability that $|Y|\leq i$ is at most the probability that $|Y'|\leq i$ for any $i\geq 0$. Now we fix the $\ell_B - \ell_R$ bits of the hash values that decide X and X'. Unless E_2 or E_3 happens we know that $|Y|\leq|Y'|$ and $|S_B|\geq|S'_B|$. Now assume that none of the bad events happen. Then we must have that the probability that $X^{\mathcal{M}} \geq (1+\delta)f$ is no larger than the probability that $\overline{X}^{\mathcal{R}} \geq (1+\delta)f$. Since this is the case for any choice of the $\ell_B - \ell_R$ bits of the hash values that decide X and X' we conclude that $P_1 \leq P_2$. Since this holds for any a and a':

$$\Pr\big[X^{\mathcal{M}} \geq (1+\delta)f \cap \neg E_1\cap\ldots\cap\neg E_5\big] \leq \Pr\Big[\overline{X}^{\mathcal{R}} \geq (1+\delta)f\Big]$$

Inserting this into (3) finishes the proof.

A. Proof of Lemma 1

We only prove the upper bound as the lower bound is symmetric.

Let $p_1, \ldots, p_b$ and $v_1, \ldots, v_b$ be the bit positions and bit values respectively such that Y is the set of keys $x \in X$ where $h(x)_{p_i} = v_i$ for all i.

Let $n = |X|$, then $n2^{-b} \in I$, where $I = \left[\frac{|\Sigma|}{8}, \frac{|\Sigma|}{4}\right)$. Partition X into 2^b sets $X_0^0, \ldots, X_{2^b-1}^0$ such that $\left|X_i^0\right| \in I$ for all $i \in [2^b]$.

For each $j = 1, \ldots, b$ and $i \in [2^{b-j}]$ let X_i^j be the set of keys $x \in \bigcup_{k=2^j \cdot i}^{2^j \cdot (i+1)-1} X_k^0$ where $h(x)_{p_k} = v_k$ for $k = 1, \ldots, j$. Equivalently, X_i^j is the set of keys $x \in X_{2i}^{j-1} \cup X_{2i+1}^{j-1}$ where $h(x)_{p_j} = v_j$. We note that $\mathbf{E}\left[\left|X_i^j\right|\right] \in I$ and $X_0^b = Y$.

Let A_j be the event that there exists $i \in [2^{b-j}]$ such that when the bit positions $p_1, \ldots, p_{j-1}$ are fixed and the remaining bit positions of the keys in X_i^j do not hash independently. By Theorem 4 $\Pr[A_j] = O\left(2^{b-j}\left|\Sigma\right|^{1-\lfloor d/2 \rfloor}\right)$. Let $s_j = \sum_{i=0}^{2^{b-j}-1}\left|X_i^j\right|$.

Fix $j \in \{1, 2, \ldots, b\}$ and the bit positions $p_1, \ldots, p_{j-1}$ of h and assume that A_{j-1} does not occur. Fix i and say that X_i^{j-1} contains r keys and write $X_i^{j-1} = \{a_0, \ldots, a_{r-1}\}$. Let V_k be the random variable defined by $V_k = 1$ if $h(a_k)_{p_j} = b_j$ and $V_k = 0$ otherwise. Let $V = \sum_{k=0}^{r-1} V_k$. Then V has mean $\frac{r}{2}$ and is the sum of independent 0-1 variables so by Chernoff's inequality:

$$\Pr\left[V \geq \frac{r}{2} \cdot (1+\delta)\right] \leq e^{-\delta^2 \cdot r/6}$$

for every $\delta \in [0, 1]$. Letting $\delta = \sqrt{\frac{6d\log|\Sigma|}{r}}$ we see that with $\alpha = \sqrt{\frac{3}{2}d\log|\Sigma|}$:

$$\Pr\left[V \geq \frac{r}{2} + \sqrt{r} \cdot \alpha\right] \leq |\Sigma|^{-d}$$

We note that $V = \left|X_i^{j-1} \cap X_{\lfloor i/2 \rfloor}^j\right|$. Hence we can rephrase it as:

$$\Pr\left[\left|X_i^{j-1} \cap X_{\lfloor i/2 \rfloor}^j\right| \geq \frac{\left|X_i^{j-1}\right|}{2} + \sqrt{\left|X_i^{j-1}\right|} \cdot \alpha\right] \leq |\Sigma|^{-d}$$

Now unfix i. By a union bound over all i we see that with probability $\geq 1 - 2^{b-j+1}\left|\Sigma\right|^{-d}$ if A_{j-1} does not occur:

$$s_j \leq \sum_{i=0}^{2^{b-j+1}-1} \frac{\left|X_i^{j-1}\right|}{2} + \sqrt{\left|X_i^{j-1}\right|} \cdot \alpha \leq \frac{s_{j-1}}{2} + \sqrt{2^{b-j+1}s_{j-1}} \cdot \alpha \tag{4}$$

Since A_{j-1} occurs with probability $O\left(2^{b-j}\left|\Sigma\right|^{1-\lfloor d/2 \rfloor}\right)$ we see that (4) holds with probability $1 - O\left(2^{b-j}\left|\Sigma\right|^{1-\lfloor d/2 \rfloor}\right)$. Let $t_j = s_j 2^{-b+j-1}$. Then (4) can be rephrased as

$$t_j \leq t_{j-1} + \sqrt{t_{j-1}} \cdot \alpha \leq \left(\sqrt{t_{j-1}} + \frac{\alpha}{2}\right)^2$$

Note that in particular:

$$\sqrt{t_j} \leq \sqrt{t_{j-1}} + \frac{\alpha}{2} \tag{5}$$

Now assume that (4) holds for every $j = b'+1, \ldots, b$ for some parameter b' to be determined. This happens with probability $1 - O\left(2^{b-b'}\left|\Sigma\right|^{1-\lfloor d/2 \rfloor}\right)$. By (5) we see that $\sqrt{t_b} \leq \sqrt{t_{b'}} + \frac{b-b'}{2}\alpha$. Hence:

$$s_b \leq \left(\sqrt{s_{b'}2^{b'-b}} + \frac{b-b'}{\sqrt{2}}\alpha\right)^2 = s_{b'}2^{b'-b} + \sqrt{2^{b'-b+1}s_{b'}}(b-b')\alpha + \left(\frac{b-b'}{\sqrt{2}}\alpha\right)^2 \tag{6}$$

We now consider two cases, when $n \leq \Sigma\log^{2c}\Sigma$ and when $n > \Sigma\log^{2c}\Sigma$. First assume that $n \leq \Sigma\log^{2c}\Sigma$. Then we let $b' = 0$ and see that with probability $1 - \tilde{O}\left(|\Sigma|^{1-\lfloor d/2 \rfloor}\right)$:

$$|Y| = s_b \leq \mathbf{E}[|Y|] + \sqrt{2\mathbf{E}[|Y|]}b\alpha + \left(\frac{b}{\sqrt{2}}\alpha\right)^2 = \mathbf{E}[|Y|] + O\left(\sqrt{\frac{\log\Sigma\,(\log\log\Sigma)^2}{\Sigma}}\right)$$

Where we used that $b = O(\log\log\Sigma)$. This proves the claim when $n \leq \Sigma\log^{2c}\Sigma$.

Now assume that $n > \Sigma\log^{2c}\Sigma$. In this case we will use Theorem 9 below.

Theorem 9 (Pătraşcu and Thorup [27])**.** *If we hash n keys into $m \leq n$ bins with simple tabulation, then, with high probability (whp.)*[1]*, every bin gets $n/m + O(\sqrt{n/m}\log^c n)$ keys.*

Let $b' \geq 0$ be such that:

$$2^{-b'} = \Theta\left(\frac{\Sigma\cdot\log^{2c} n}{n}\right)$$

With $\gamma = \lfloor d/2\rfloor - 1$ in Theorem 9 we see that with probability $1 - O\big(|\Sigma|^{1-\lfloor d/2\rfloor}\big)$:

$$s_{b'} \leq 2^{-b'}n + O\Big(\sqrt{2^{-b'}n}\log^c n\Big) = 2^{-b'}n\cdot\left(1 + O\left(\sqrt{\frac{1}{\Sigma}}\right)\right) \tag{7}$$

By a union bound both (6) and (7) hold with probability $1 - \tilde{O}\big(|\Sigma|^{1-\lfloor d/2\rfloor}\big)$ and combining these will give us the desired upper bound. This concludes the proof when $n > \Sigma\log^{2c}\Sigma$.

III. Bounding dependencies

In order to proof our main technical result of Theorem 4 we need the following structural lemmas regarding the dependencies of simple tabulation.

Simple tabulation is not 4-independent which means that there exists keys $x_1,\ldots,x_4$, such that $h(x_1)$ is dependent of $h(x_2), h(x_3), h(x_4)$. It was shown in [27], that for every $X \subseteq U$ with $|X| = n$ there are at most $O(n^2)$ such dependent 4-tuples $(x_1,x_2,x_3,x_4) \in X^4$.

In this section we show that a similar result holds in the case of dependent k-tuples, which is one of the key ingredients in the proofs of the main theorems of this paper.

We know from [1] that if the keys $x_1,\ldots,x_k$ are dependent, then there exists a non-empty subset $I \subset \{1,\ldots,k\}$ such that

$$\bigoplus_{i\in I} x_i = \emptyset\ .$$

Following this observation we wish to bound the number of tuples which have symmetric difference $\emptyset$.

Lemma 2. *Let $X \subseteq U$ with $|X| = n$ be a subset. The number of $2t$-tuples $(x_1,\ldots,x_{2t}) \in X^{2t}$ such that*

$$x_1 \oplus \cdots \oplus x_{2t} = \emptyset$$

is at most $((2t-1)!!)^c n^t$, where $(2t-1)!! = (2t-1)(2t-3)\cdots 3\cdot 1$.

It turns out that it is more convenient to prove the following more general lemma.

Lemma 3. *Let $A_1,\ldots,A_{2t} \subset U$ be sets of keys. The number of $2t$-tuples $(x_1,\ldots,x_{2t}) \in A_1\times\cdots\times A_{2t}$ such that*

$$x_1 \oplus \cdots \oplus x_{2t} = \emptyset \tag{8}$$

is at most $((2t-1)!!)^c\prod_{i=1}^{2t}\sqrt{|A_i|}$.

Proof of Lemma 3: Let $(x_1,\ldots,x_{2t})$ be such a $2t$-tuple. Equation (8) implies that the number of times each position character appears is an even number. Hence we can partition $(x_1,\ldots,x_{2t})$ into t pairs $(x_{i_1},x_{j_1}),\ldots,(x_{i_t},x_{j_t})$ such that $\pi(x_{i_k},c-1) = \pi(x_{j_k},c-1)$ for $k = 1,\ldots,t$. Note that there are at $(2t-1)!!$ ways to partition the elements in such a way. This is illustrated in Figure 3.

We now prove the claim by induction on c. First assume that $c = 1$. We fix some partition $(x_{i_1},x_{j_1}),\ldots,(x_{i_t},x_{j_t})$ and count the number of $2t$-tuples which fulfil $\pi(x_{i_k},c-1) = \pi(x_{j_k},c-1)$ for $k = 1,\ldots,t$. Since $c = 1$ we have $x_{i_k},x_{j_k} \in A_{i_k}\cap A_{j_k}$. The number of ways to choose such a $2t$-tuple is thus bounded by:

$$\prod_{k=1}^{t}|A_{i_k}\cap A_{j_k}| \leq \prod_{k=1}^{t}\min\left\{|A_{i_k}|,|A_{j_k}|\right\} \leq \prod_{k=1}^{t}\sqrt{|A_{i_k}|\,|A_{j_k}|} = \prod_{k=1}^{2t}\sqrt{|A_k|}$$

And since there are $(2t-1)!!$ such partitions the case $c = 1$ is finished.

[1]With probability $1 - n^{-\gamma}$ for any $\gamma = O(1)$.

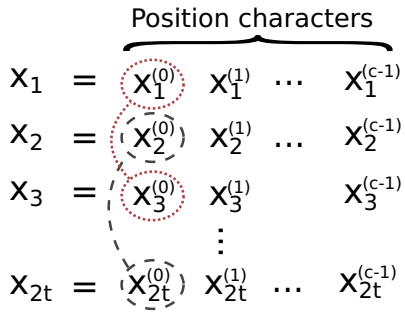

Figure 3. Pairing of the position characters of $2t$ keys. $x_1^{(0)}$ can be matched to $2t-1$ position characters, $x_2^{(0)}$ to $2t-3$, etc.

Now assume that the lemma holds when the keys have $< c$ characters. As before, we fix some partition $(x_{i_1}, x_{j_1}), \ldots, (x_{i_t}, x_{j_t})$ and count the number of $2t$-tuples which satisfy $\pi(x_{i_k}, c-1) = \pi(x_{j_k}, c-1)$ for all $k = 1, \ldots, t$. Fix the last position character $(a_k, c-1) = \pi(x_{i_k}, c-1) = \pi(x_{j_k}, c-1)$ for $k = 1, \ldots, t$, $a_k \in \Sigma$. The rest of the position characters from x_{i_k} is then from the set

$$A_{i_k}[a_k] = \{x \backslash (a_k, c-1) \mid (a_k, c-1) \in x, x \in A_{i_k}\}$$

By the induction hypothesis the number of ways to choose $x_1, \ldots, x_{2t}$ with this choice of $a_1, \ldots, a_t$ is then at most:

$$((2t-1)!!)^{c-1} \prod_{k=1}^{t} \sqrt{|A_{i_k}[a_k]|\,|A_{j_k}[a_k]|}$$

Summing over all choices of $a_1, \ldots, a_t$ this is bounded by:

$$\begin{aligned}
&((2t-1)!!)^{c-1} \sum_{a_1,\ldots,a_t \in \Sigma} \prod_{k=1}^{t} \sqrt{|A_{i_k}[a_k]|\,|A_{j_k}[a_k]|} \\
=&((2t-1)!!)^{c-1} \prod_{k=1}^{t} \sum_{a_k \in \Sigma} \sqrt{|A_{i_k}[a_k]|\,|A_{j_k}[a_k]|} \\
\leq&((2t-1)!!)^{c-1} \prod_{k=1}^{t} \sqrt{\sum_{a_k \in \Sigma} |A_{i_k}[a_k]|} \sqrt{\sum_{a_k \in \Sigma} |A_{j_k}[a_k]|} \qquad (9) \\
=&((2t-1)!!)^{c-1} \prod_{k=1}^{t} \sqrt{|A_{i_k}|}\sqrt{|A_{j_k}|} = ((2t-1)!!)^{c-1} \prod_{k=1}^{2t} \sqrt{|A_k|}
\end{aligned}$$

Here (9) is an application of Cauchy-Schwartz's inequality. Since there are $(2t-1)!!$ such partitions the conclusion follows. ∎

IV. Uniform hashing in constant time

This section is dedicated to proving Theorem 4. We will show the following more general theorem. This proof also implies the result of Theorem 7.

Theorem 10. *Let $h = h_2 \circ h_1^\star$ be a mixed tabulation hash function. Let $X \subset [u]$ be any input set. For each $x \in X$, associate a function $f_x : \mathcal{R} \to \{0,1\}$. Let $Y = \{x \in X \mid f_x(h(x)) = 1\}$ and assume $\mathbf{E}[|Y|] \leq |\Sigma|/(1+\varepsilon)$.*
Then the keys of $h_1^\star(Y) \subseteq \Sigma^{c+d}$ are peelable with probability $1 - O(|\Sigma|^{1-\lfloor d/2 \rfloor})$.

Here, we consider only the case when there exists a p such that $\Pr[f_x(z) = 1] = p$ for all x, when z is uniformly distributed in $\mathcal{R}$. In Section V we sketch the details when this is not the case. We note that the full proof uses the same ideas but is more technical.

The proof is structured in the following way: (1) We fix Y and assume the key set $h_1^\star(Y)$ is not independent. (2) With Y fixed this way we construct a *bad event*. (3) We unfix Y and show that the probability of a bad event occurring is low using a union bound. Each bad event consists of independent "sub-events" relating to subgraphs of the hash graph of $h_1(Y)$. These sub-events fall into four categories, and for each of those we will bound the probability that the event occurs.

First observe that if a set of keys S consists of independent keys, then the set of keys $h_1^\star(S)$ are also independent.

We will now describe what we mean by a *bad event*. We consider the hash function $h_1 : [u] \to \Sigma^d$ as d simple tabulation hash functions $h^{(0)}, \ldots, h^{(d-1)} : [u] \to \Sigma$ and define $G_{i,j}$ to be the hash graph of $h^{(i)}, h^{(j)}$ and the input set X.

Fix Y and consider some $y \in Y$. If for some i, j, the component of $G_{i,j}$ containing y is a tree, then we can perform a peeling process and observe that $h_1^\star(y)$ must be independent of $h_1^\star(Y \setminus \{y\})$. Now assume that there exists some $y_0 \in Y$ such that $h_1^\star(y_0)$ is dependent of $h_1^\star(Y \setminus \{y_0\})$, then y_0 must lie on a (possibly empty) path leading to a cycle in each of $G_{2i,2i+1}$ for $i \in [\lfloor d/2 \rfloor]$. We will call such a path and cycle a *lollipop*. Denote this lollipop by $y_0, y_1^i, y_2^i, \ldots, y_{p_i}^i$. For each such i we will construct a list L_i to be part of our bad event. Set $s \stackrel{def}{=} \lceil 2 \log_{1+\varepsilon} |\Sigma| \rceil$. The list L_i is constructed in the following manner: We walk along $y_1^i, \ldots, y_{p_i}^i$ until we meet an *obstruction*. Consider a key y_j^i. We will say that y_j^i is an obstruction if it falls into one of the following four cases as illustrated in Figure 4.

- A There exists some subset $B \subseteq \{y_0, y_1^i, \ldots, y_{j-1}^i\}$ such that $y_j^i = \bigoplus_{y \in B} y$.
- B If case **A** does not hold and there exists some subset $B \subseteq \{y_0, y_1^i, \ldots, y_{j-1}^i\} \cup L_0 \cup \ldots \cup L_{i-1}$ such that $y_j^i = \bigoplus_{y \in B} y$.
- C $j = p_i < s$ (i.e. y_j^i is the last key on the cycle). In this case y_j^i must share a node with either y_0 (the path of the lollipop is empty) or with two of the other keys in the lollipop.
- D $j = s$. In this case the keys $y_1^i, \ldots, y_s^i$ form a path keys independent from $L_0, \ldots, L_{i-1}$.

In all four cases we set $L_i = (y_1^i, \ldots, y_j^i)$ and we associate an attribute A_i. In case **A** we set $A_i = B$. In case **B** we set $A = (x^{(0)}, \ldots, x^{(c-1)})$, where $x^{(r)} \in B$ is chosen such that $\pi(y_j^i, r) = \pi(x^{(r)}, r)$. In **C** we set $A_i = z$, where z is the smallest value such that y_z^i shares a node with y_j^i, and in **D** we set $A_i = \emptyset$. Denote the lists by L and the types and attributes of the lists by T, A. We have shown, that if there is a dependency among the keys of $h_1^\star(Y)$, then we can find such a bad event (y_0, L, T, A).

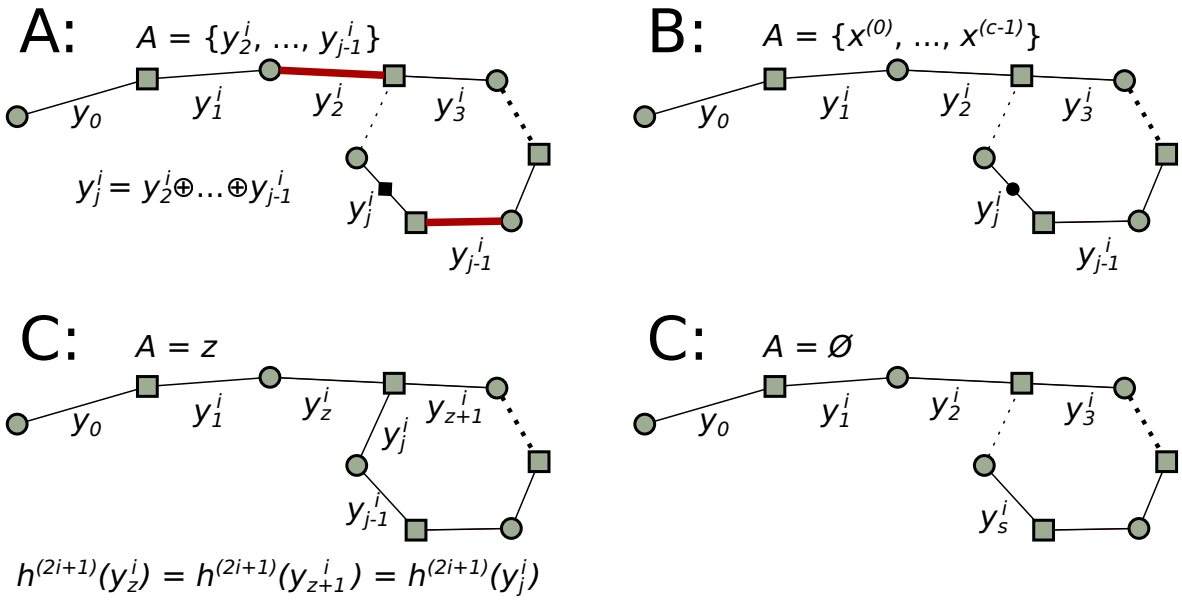

Figure 4. The four types of violations. Dependent keys are denoted by ■ and ●.

Now fix $y_0 \in X, l = (l_0, \ldots, l_{\lfloor d/2 \rfloor - 1})$. Let $F(y_0, l)$ be the event that there exists a quadruple (y_0, L, T, A) forming a bad event such that $|L_i| = l_i$. We use the shorthand $F = F(y_0, l)$. Let $F(y_0, L, T, A)$ denote the event that a given quadruple (y_0, L, T, A) occurs. Note that a quadruple (y_0, L, T, A) only occurs if some conditions are satisfied for h_1 (i.e. that the hash graph forms the lollipops as described earlier) and h_2 (i.e. that the keys of the lollipops are contained in Y). Let $F_1(y_0, L, T, A)$ and $F_2(y_0, L, T, A)$ denote the the event that those conditions are satisfied, respectively. Then

$$\begin{aligned} \Pr[F] &\leq \sum_{\text{bad event } L,\, T,\, A} \Pr[F(y_0, L, T, A)] \\ &= \sum_{\text{bad event } L,\, T,\, A} \Pr[F_2(y_0, L, T, A) | F_1(y_0, L, T, A)] \cdot \Pr[F_1(y_0, L, T, A)] \ . \end{aligned}$$

We note, that $F_1(y_0, L, T, A)$ consists of independent events for each $G_{2i,2i+1}$, for $i \in [\lfloor d/2 \rfloor]$. Denote these restricted events by $F_1^i(y_0, L, T, A)$.

For a fixed h_1 we can bound $\Pr[F_2(y_0, L, T, A)]$ in the following way: For each $i \in [\lfloor d/2 \rfloor]$ we choose a subset $V_i \subseteq L_i$ such that $S = \{y_0\} \bigcup_i V_i$ consists of independent keys. Since these keys are independent, so is $h_1^\star(S)$, so we can bound the probability that $S \subseteq Y$ by $p^{|S|}$. We can split this into one part for each i. Define

$$p_i \stackrel{def}{=} p^{|V_i|} \cdot \Pr\left[F_1^i(y_0, L_i, T_i, A_i)\right] \ .$$

We can then bound $\Pr[F] \leq p \cdot \prod_{i \in [\lfloor d/2 \rfloor]} p_i$.

We now wish to bound the probability p_i. Consider some $i \in [\lfloor d/2 \rfloor]$. We split the analysis into a case for each of the four types:

A Let $\Delta(y_0)$ be the number of triples $(a, b, c) \in X^3$ such that $y_0 \oplus a \oplus b \oplus c = \emptyset$. Note that the size of the attribute $|A_i| \geq 3$ must be odd. Consider the following three cases:

1) $|A_i| = 3, y_0 \in A_i$: We have y_0 is the $\oplus$-sum of three elements of L_i. The number of ways this can happen (i.e. the number of ways to choose L_i and A_i) is bounded by $l_i^3 n^{l_i - 3} \Delta(y_0)$ – The indices of the three summands can be chosen in at most l_i^3 ways, and the corresponding keys in at most $\Delta(y_0)$ ways. The remaining elements can be chosen in at most $n^{l_i - 3}$ ways.
2) $|A_i| \geq 5, y_0 \in A_i$: By Lemma 3 we can choose L_i and A_i in at most $l_i^{O(1)} \cdot n^{l_i - 5/2}$ ways.
3) $|A_i| \geq 3, y_0 \notin A_i$: By Lemma 3 we can choose L_i and A_i in at most $l_i^{O(1)} \cdot n^{l_i - 2}$ ways.

To conclude, we can choose L_i and A_i in at most

$$l_i^{O(1)} \cdot n^{l_i - 2} \cdot \left(1 + \frac{\Delta(y_0)}{n}\right)$$

ways. We can choose V_i to be L_i except for the last key. We note that $V_i \cup \{y_0\}$ form a path in $G_{2i,2i+1}$, which happens with probability $1/\left|\Sigma\right|^{l_i - 1}$ since the keys are independent. For type **A** we thus get the bound

$$\begin{aligned} p_i &\leq l_i^{O(1)} \cdot p^{l_i - 1} \cdot n^{l_i - 2} \cdot \left(1 + \frac{\Delta(y_0)}{n}\right) \cdot \frac{1}{\left|\Sigma\right|^{l_i - 1}} \leq l_i^{O(1)} \cdot \left(1 + \frac{\Delta(y_0)}{n}\right) \cdot \frac{1}{|\Sigma|} \cdot \frac{p}{(1+\varepsilon)^{l_i - 2}} \\ &\leq l^{O(1)} \cdot \left(1 + \frac{\Delta(y_0)}{n}\right) \cdot \frac{1}{|\Sigma|} \cdot \frac{1}{(1+\varepsilon)^{l_i/2}} \ . \end{aligned}$$

B All but the last key of L_i are independent and can be chosen in at most $n^{l_i - 1}$ ways. The last key is uniquely defined by A_i, which can be chosen in at most l^c ways (where $l = \sum_i l_i$), thus L_i and A_i can be chosen in at most $n^{l_i - 1} l^c$ ways. Define V_i to be all but the last key of L_i. The keys of $L_i \cup \{y_0\}$ form a path, and since the last key of L_i contains a position character not in V_i, the probability of this path occurring is exactly $1/\left|\Sigma\right|^{l_i}$, thus we get

$$p_i \leq l^c \cdot n^{l_i - 1} \cdot p^{l_i - 1} \cdot \frac{1}{\left|\Sigma\right|^{l_i}} \leq l^{O(1)} \cdot \frac{1}{|\Sigma|} \cdot \frac{1}{(1+\varepsilon)^{l_i - 1}} \leq l^{O(1)} \cdot \frac{1}{|\Sigma|} \cdot \frac{1}{(1+\varepsilon)^{l_i/2}} \ .$$

C The attribute A_i is just a number in $[l_i]$, and L_i can be chosen in at most n^{l_i} ways. We can choose $V_i = L_i$. $V_i \cup \{y_0\}$ is a set of independent keys forming a path leading to a cycle, which happens with probability $1/\left|\Sigma\right|^{l_i + 1}$, so we get the bound

$$p_i \leq l_i \cdot n^{l_i} \cdot p^{l_i} \cdot \frac{1}{\left|\Sigma\right|^{l_i + 1}} \leq l_i \cdot \frac{1}{|\Sigma|} \cdot \frac{1}{(1+\varepsilon)^{l_i}} \leq l^{O(1)} \cdot \frac{1}{|\Sigma|} \cdot \frac{1}{(1+\varepsilon)^{l_i/2}} \ .$$

D The attribute $A_i = \emptyset$ is uniquely chosen. L_i consists of s independent keys and can be chosen in at most n^s ways. We set $V_i = L_i$. We get

$$p_i \leq n^s \cdot p^s \cdot \frac{1}{\left|\Sigma\right|^s} \leq \frac{1}{(1+\varepsilon)^s} \leq \frac{1}{|\Sigma|} \cdot \frac{1}{(1+\varepsilon)^{l_i/2}} \ .$$

We first note, that there exists y_0 such that $\Delta(y_0) = O(n)$. We have just shown that for a specific y_0 and partition of the lengths $(l_0, \ldots, l_{\lfloor d/2 \rfloor})$ we get

$$\Pr[F] \leq p \cdot \left(l^{O(1)} \cdot \frac{1}{|\Sigma|}\right)^{\lfloor d/2 \rfloor} \cdot \frac{1}{(1+\varepsilon)^{l/2}} \ .$$

Summing over all partitions of the l_i's and choices of l gives

$$\sum_{l\geq 1} p \cdot l^{O(1)} \cdot |\Sigma|^{-\lfloor d/2 \rfloor} \cdot \frac{1}{(1+\varepsilon)^{l/2}} \leq O\left(p \cdot |\Sigma|^{-\lfloor d/2 \rfloor}\right) .$$

We have now bounded the probability for $y_0 \in X$ that $y_0 \in Y$ and y_0 is dependent on $Y \setminus \{y_0\}$. We relied on $\Delta(y_0) = O(n)$, so we cannot simply take a union bound. Instead we note that, if y_0 is independent of $Y \setminus \{y_0\}$ we can peel y_0 away and use the same argument on $X \setminus \{y_0\}$. This gives a total upper bound of

$$O\left(\sum_{y_0 \in X} p \cdot |\Sigma|^{-\lfloor d/2 \rfloor}\right) = O(|\Sigma|^{1-\lfloor d/2 \rfloor}) .$$

This finishes the proof. □

V. Uniform hashing with multiple probabilities

Here we present a sketch in extending the proof in Section IV. We only need to change the proof where we bound p_i. Define $p_x = \Pr[f_x(z) = 1]$ when z is uniformly distributed in $\mathcal{R}$. First we argue that cases **B**, **C** and **D** are handled in almost the exact same way. In the original proof we argued that for some size v we can choose $V_i, |V_i| = v$ in at most n^v ways and for each choice of V_i the probability that it is contained in Y is at most p^v, thus multiplying the upper bound by

$$n^v p^v = (\mathbf{E}\,|Y|)^v$$

For our proof we sum over all choices of V_i and add the probabilities that V_i is contained in Y getting the exact same estimate:

$$\sum_{V_i \in U, |V_i| = v} \left(\prod_{x \in V_i} p_x\right) \leq \left(\sum_{x \in U} p_x\right)^v = (\mathbf{E}\,|Y|)^v$$

The difficult part is to prove the claim in case **A**.

For all $i \geq 0$ we set

$$n_i = \left|\left\{x \in X \mid p_x \in \left(2^{-i-1}, 2^{-i}\right]\right\}\right| .$$

Now observe, that $\sum_{i\geq 0} n_i 2^{-i} \leq 2\,|\Sigma|\,/(1+\varepsilon) = O(|\Sigma|)$. Define $m_i = \sum_{j \leq i} n_j$, we then have:

$$\sum_{i\geq 0} m_i 2^{-i} = \sum_{i \geq 0} n_i \left(\sum_{j \geq i} 2^{-j}\right) = \sum_{i \geq 0} n_i 2^{-i+1} = O(|\Sigma|)$$

We let $X_i = \left\{x \in X \mid p_x > 2^{-i-1}\right\}$ and note that $m_i = |X_i|$. For each $y_0 \in X$ we will define $\Delta'(y_0)$ (analogously to $\Delta(y_0)$) in the following way:

$$\Delta'(y_0) = \sum_{a,b,c \in X} \min\{p_a p_b, p_b p_c, p_c p_a\}$$

where we only sum over triples (a, b, c) such that $y_0 \oplus a \oplus b \oplus c = \emptyset$. Analogously to the original proof we will show that there exists y_0 such that $\Delta'(y_0) \leq O(|\Sigma|)$. The key here is to prove that:

$$\sum_{y_0 \in X} \Delta'(y_0) = O\left(n\,|\Sigma|\right)$$

Now consider a 4-tuple (y_0, a, b, c) such that $y_0 \oplus a \oplus b \oplus c = \emptyset$. Let $i \geq 0$ be the smallest non-negative integer such that $b, c \in X_i$. Then:

$$\min\{p_a p_b, p_b p_c, p_c p_a\} \leq \min\{p_b, p_c\} \leq 2^{-i}$$

By 3 we see that for any i there are at most $O(nm_i)$ 4-tuples (y_0, a, b, c) such that $b, c \in X_i$. This gives the following bound on the total sum:

$$\sum_{y_0 \in X} \Delta'(y_0) \leq \sum_{i \geq 0} O(nm_i) \cdot 2^{-i} = O\left(n\,|\Sigma|\right)$$

Hence there exists y_0 such that $\Delta'(y_0) = O(|\Sigma|)$ and we can finish case **A.1** analogously to the original proof.

Now we turn to case **A.2** where $|A_i| \geq 5, y_0 \in A_0$. We will here only consider the case $|A_i| = 5$, since the other cases follow by the same reasoning. We will choose V_i to consist of all of $L_i \setminus A_i$ and 3 keys from A_i. We will write $A_i = \{a, b, c, d, e\}$ and find the smallest α, β, γ such that $a, b \in X_\alpha, c, d \in X_\beta, e \in X_\gamma$. Then:

$$\prod_{x \in V_i} p_x \leq \left(\prod_{x \in V_i \setminus A_i} p_x \right) 2^{-\alpha} 2^{-\beta} 2^{-\gamma}$$

When $a, b \in X_\alpha, c, d \in X_\beta, e \in X_\gamma$ we can choose a, b, c, d, e in at most $m_\alpha m_\beta \sqrt{m_\gamma}$ ways by Lemma 3. Hence, when we sum over all choices of V_i we get an upper bound of:

$$\left(\sum_{x \in X} p_x \right)^{l_i - 5} \left(\sum_{\alpha, \beta, \gamma \geq 0} m_\alpha m_\beta \sqrt{m_\gamma} 2^{-\alpha} 2^{-\beta} 2^{-\gamma} \right) = \left(\sum_{x \in X} p_x \right)^{l_i - 5} \left(\sum_{\alpha \geq 0} m_\alpha 2^{-\alpha} \right)^2 \left(\sum_{\alpha \geq 0} \sqrt{m_\alpha} 2^{-\alpha} \right)$$

Now we note that by Cauchy-Schwartz inequality:

$$\sum_{\alpha \geq 0} \sqrt{m_\alpha} 2^{-\alpha} \leq \sqrt{\sum_{\alpha \geq 0} 2^{-\alpha}} \sqrt{\sum_{\alpha \geq 0} m_\alpha 2^{-\alpha}} = O(\sqrt{|\Sigma|})$$

Hence we get a total upper bound of $O(|\Sigma|^{l_i - 5/2})$ and we can finish the proof in analogously to the original proof.

Case **A.3** is handled similarly to **A.2**.

VI. Constant moment bounds

This section is dedicated to proving Theorem 8.

Consider first Theorem 8 and let $k = O(1)$ be fixed. Define $Z_i = Y_i - p$ for all $i \in [m]$ and $Z = \sum_{i \in [m]} Z_i$. We wish to bound $\mathbf{E}\big[Z^{2k}\big]$ and by linearity of expectation this equals:

$$\mathbf{E}\big[Z^{2k}\big] = \sum_{r_0, \ldots, r_{2k-1} \in [m]^{2k}} \mathbf{E}\big[Z_{r_0} \cdots Z_{r_{2k-1}}\big]$$

Fix some $2k$-tuple $r = (r_0, \ldots, r_{2k-1}) \in [m]^{2k}$ and define $V(r) = \mathbf{E}\big[Z_{r_0} \cdots Z_{r_{2k-1}}\big]$. Observe, that if there exists $i \in [2k]$ such that x_{r_i} is independent of $(x_{r_j})_{j \neq i}$ then

$$V(r) = \mathbf{E}\big[Z_{r_0} \cdots Z_{r_{2k-1}}\big] = \mathbf{E}[Z_{r_i}]\, \mathbf{E}\left[\prod_{j \neq i} Z_{r_j}\right] = 0$$

The following lemma bounds the number of $2k$-tuples, r, for which $V(r) \neq 0$.

Lemma 4. *The number of $2k$-tuples r such that $V(r) \neq 0$ is $O(m^k)$.*

Proof: Fix $r \in [m]^{2k}$ and let $T_0, \ldots, T_{s-1}$ be all subsets of $[2k]$ such that $\bigoplus_{i \in T_j} x_{r_i} = \emptyset$ for $j \in [s]$. If $\bigcup_{j \in [s]} T_j \neq [2k]$ we must have $V(r) = 0$ as there exists some x_{r_i}, which is independent of $(x_{r_j})_{j \neq i}$. Thus we can assume that $\bigcup_{j \in [s]} T_j = [2k]$.

Now fix $T_0, \ldots, T_{s-1} \subseteq [2k]$ such that $\bigcup_{j \in [s]} T_j = [2k]$ and count the number of ways to choose $r \in [m]^{2k}$ such that $\bigoplus_{i \in T_j} x_{r_i} = \emptyset$ for all $j \in [s]$. Note that $T_0, \ldots, T_{s-1}$ can be chosen in at most $2^{2k} = O(1)$ ways, so if we can bound the number of ways to choose r by $O(m^k)$ we are done. Let $A_i = \bigcup_{j<i} T_j$ and $B_i = T_i \setminus A_i$ for $i \in [s]$. We will choose r by choosing $(x_{r_i})_{i \in B_0}$, then $(x_{r_i})_{i \in B_1}$, and so on up to $(x_{r_i})_{i \in B_{s-1}}$. When we choose $(x_{r_i})_{i \in B_j}$ we have already chosen $(x_{r_i})_{i \in A_j}$ and by Lemma 3 the number of ways to choose $(x_{r_i})_{i \in B_j}$ is bounded by:

$$((|T_j| - 1)!!)^c\, m^{|B_j|/2} = O\left(m^{|B_j|/2}\right)$$

Since $\bigcup_{j \in [s]} B_j = [2k]$ we conclude that the number of ways to choose r such that $V(r) \neq 0$ is at most $O(m^k)$. ∎

We note that since $|V(r)| \leq 1$ this already proves that

$$\mathbf{E}\big[Z^{2k}\big] \leq O(m^k)$$

Consider now any $r \in [m]^{2k}$ and let $f(r)$ denote the size of the largest subset $I \subset [2k]$ of independent keys $(x_{r_i})_{i \in I}$. We then have

$$\left| \mathbf{E}\left[\prod_{i \in [2k]} Z_{r_i} \right] \right| \leq \mathbf{E}\left[\left| \prod_{i \in [2k]} Z_{r_i} \right| \right] \leq \mathbf{E}\left[\left| \prod_{i \in I} Z_{r_i} \right| \right] \leq O\left(p^{f(r)}\right)$$

We now fix some value $s \in \{1, \ldots, 2k\}$ and count the number of $2k$-tuples r such that $f(r) = s$. We can bound this number by first choosing the s independent keys of I in at most m^s ways. For each remaining key we can write it as a sum of a subset of $(x_{r_i})_{i \in I}$. There are at most $2^s = O(1)$ such subsets, so there are at most $O(m^s)$ such $2k$-tuples r with $f(r) = s$.

Now consider the $O(m^k)$ $2k$-tuples $r \in [m]^{2k}$ such that $V(r) \neq 0$. For each $s \in \{1, \ldots, 2k\}$ there is $O(m^{\min\{k,s\}})$ ways to choose r such that $f(r) = s$. All these choices of r satisfy $V(r) \leq O(p^s)$. Hence:

$$\mathbf{E}\big[Z^{2k}\big] = \sum_{r \in [m]^{2k}} V(r) \leq \sum_{s=1}^{2k} O(m^{\min\{k,s\}}) \cdot O(p^s) = O\left(\sum_{s=1}^{k} (pm)^s \right) .$$

This finishes the proof of Theorem 8. □

A similar argument can be used to show the following theorem, where the bin depends on a query key q.

Theorem 11. *Let $h : [u] \to \mathcal{R}$ be a simple tabulation hash function. Let $x_0, \ldots, x_{m-1}$ be m distinct keys from $[u]$ and let $q \in [u]$ be a* query *key distinct from $x_0, \ldots, x_{m-1}$. Let $Y_0, \ldots, Y_{m-1}$ be any random variables such that $Y_i \in [0, 1]$ is a function of $(h(x_i), h(q))$ and for all $r \in \mathcal{R}$, $\mathbf{E}[Y_i \mid h(q) = r] = p$ for all $i \in [m]$. Define $Y = \sum_{i \in [m]} Y_i$ and $\mu = \mathbf{E}[Y] = mp$. Then for any constant integer $k \geq 1$:*

$$\mathbf{E}\big[(Y - \mu)^{2k}\big] \leq O\left(\sum_{j=1}^{k} \mu^j \right) ,$$

where the constant in the O-notation is dependent on k and c.

ACKNOWLEDGMENT

Søren Dahlgaard's, Mathias Bæk Tejs Knudsen's, and Mikkel Thorup's research is partly supported by Thorup's Advanced Grant DFF-0602-02499B from the Danish Council for Independent Research under the Sapere Aude research career programme. Mathias Bæk Tejs Knudsen's research is also partly supported by the FNU project AlgoDisc - Discrete Mathematics, Algorithms, and Data Structures.

REFERENCES

[1] M. Thorup and Y. Zhang, "Tabulation-based 5-independent hashing with applications to linear probing and second moment estimation," *SIAM Journal on Computing*, vol. 41, no. 2, pp. 293–331, 2012, announced at SODA'04 and ALENEX'10.

[2] P. Flajolet and G. N. Martin, "Probabilistic counting algorithms for data base applications," *Journal of Computer and System Sciences*, vol. 31, no. 2, pp. 182–209, 1985, announced at FOCS'83.

[3] P. Flajolet, Éric Fusy, O. Gandouet, and et al., "Hyperloglog: The analysis of a near-optimal cardinality estimation algorithm," in *In Analysis of Algorithms (AOFA)*, 2007.

[4] S. Heule, M. Nunkesser, and A. Hall, "Hyperloglog in practice: Algorithmic engineering of a state of the art cardinality estimation algorithm," in *Proceedings of the EDBT 2013 Conference*, 2013, pp. 683–692.

[5] P. Boldi, M. Rosa, and S. Vigna, "Hyperanf: Approximating the neighbourhood function of very large graphs on a budget," in *Proc. 20th WWW*. ACM, 2011, pp. 625–634.

[6] E. Cohen, "All-distances sketches, revisited: Hip estimators for massive graphs analysis," in *Proc. 33rd ACM Symposium on Principles of Database Systems*. ACM, 2014, pp. 88–99.

[7] P. Li, A. B. Owen, and C.-H. Zhang, "One permutation hashing," in *Proc. 26thAdvances in Neural Information Processing Systems*, 2012, pp. 3122–3130.

[8] A. Shrivastava and P. Li, "Densifying one permutation hashing via rotation for fast near neighbor search," in *Proc. 31th International Conference on Machine Learning (ICML)*, 2014, pp. 557–565.

[9] ——, "Improved densification of one permutation hashing," in *Proceedings of the Thirtieth Conference on Uncertainty in Artificial Intelligence, UAI 2014, Quebec City, Quebec, Canada, July 23-27, 2014*, 2014, pp. 732–741.

[10] A. Z. Broder, S. C. Glassman, M. S. Manasse, and G. Zweig, "Syntactic clustering of the web," *Computer Networks*, vol. 29, pp. 1157–1166, 1997.

[11] A. Z. Broder, M. Charikar, A. M. Frieze, and M. Mitzenmacher, "Min-wise independent permutations," *Journal of Computer and System Sciences*, vol. 60, no. 3, pp. 630–659, 2000, see also STOC'98.

[12] A. Z. Broder, "On the resemblance and containment of documents," in *Proc. Compression and Complexity of Sequences (SEQUENCES)*, 1997, pp. 21–29.

[13] M. Charikar, K. Chen, and M. Farach-Colton, "Finding frequent items in data streams," in *Proc. 29th International Colloquium on Automata, Languages and Programming (ICALP)*. Springer-Verlag, 2002, pp. 693–703.

[14] P. Li, A. C. König, and W. Gui, "b-bit minwise hashing for estimating three-way similarities," in *Proc. 24thAdvances in Neural Information Processing Systems*, 2010, pp. 1387–1395.

[15] P. Li and A. C. König, "b-bit minwise hashing," in *Proc. 19th WWW*, 2010, pp. 671–680.

[16] P. Li, A. Shrivastava, J. L. Moore, and A. C. König, "Hashing algorithms for large-scale learning," in *Proc. 25thAdvances in Neural Information Processing Systems*, 2011, pp. 2672–2680.

[17] Y. Bachrach and E. Porat, "Sketching for big data recommender systems using fast pseudo-random fingerprints," in *Proc. 40th International Colloquium on Automata, Languages and Programming (ICALP)*, 2013, pp. 459–471.

[18] Z. Bar-Yossef, T. S. Jayram, R. Kumar, D. Sivakumar, and L. Trevisan, "Counting distinct elements in a data stream," in *Proc. 6th International Workshop on Randomization and Approximation Techniques (RANDOM)*, 2002, pp. 1–10.

[19] M. Thorup, "Bottom-k and priority sampling, set similarity and subset sums with minimal independence," in *Proc. 45th ACM Symposium on Theory of Computing (STOC)*, 2013.

[20] P. Indyk and R. Motwani, "Approximate nearest neighbors: Towards removing the curse of dimensionality," in *Proc. 30th ACM Symposium on Theory of Computing (STOC)*, 1998, pp. 604–613.

[21] A. Andoni and P. Indyk, "Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions," *Communications of the ACM*, vol. 51, no. 1, pp. 117–122, 2008, see also FOCS'06.

[22] A. Andoni, P. Indyk, H. L. Nguyen, and I. Razenshteyn, "Beyond locality-sensitive hashing," in *Proc. 25th ACM/SIAM Symposium on Discrete Algorithms (SODA)*, 2014, pp. 1018–1028.

[23] R. Motwani and P. Raghavan, *Randomized algorithms*. Cambridge University Press, 1995.

[24] M. Mitzenmacher and E. Upfal, *Probability and computing - randomized algorithms and probabilistic analysis*. Cambridge University Press, 2005.

[25] M. N. Wegman and L. Carter, "New classes and applications of hash functions," *Journal of Computer and System Sciences*, vol. 22, no. 3, pp. 265–279, 1981, see also FOCS'79.

[26] L. E. Celis, O. Reingold, G. Segev, and U. Wieder, "Balls and bins: Smaller hash families and faster evaluation," in *Proc. 52nd IEEE Symposium on Foundations of Computer Science (FOCS)*, 2011, pp. 599–608.

[27] M. Pătraşcu and M. Thorup, "The power of simple tabulation-based hashing," *Journal of the ACM*, vol. 59, no. 3, p. Article 14, 2012, announced at STOC'11.

[28] P. Indyk, "A small approximately min-wise independent family of hash functions," *Journal of Algorithms*, vol. 38, no. 1, pp. 84–90, 2001, see also SODA'99.

[29] M. Pătraşcu and M. Thorup, "On the k-independence required by linear probing and minwise independence," in *Proc. 37th International Colloquium on Automata, Languages and Programming (ICALP)*, 2010, pp. 715–726.

[30] A. Pagh and R. Pagh, "Uniform hashing in constant time and optimal space," *SIAM J. Comput.*, vol. 38, no. 1, pp. 85–96, 2008.

[31] T. Christiani, R. Pagh, and M. Thorup, "From independence to expansion and back again," 2015, to appear.

[32] L. Carter and M. N. Wegman, "Universal classes of hash functions," *Journal of Computer and System Sciences*, vol. 18, no. 2, pp. 143–154, 1979, see also STOC'77.

[33] M. Thorup, "Simple tabulation, fast expanders, double tabulation, and high independence," in *FOCS*, 2013, pp. 90–99.

[34] A. Siegel, "On universal classes of extremely random constant-time hash functions," *SIAM Journal on Computing*, vol. 33, no. 3, pp. 505–543, 2004, see also FOCS'89.

[35] A. L. Zobrist, "A new hashing method with application for game playing," Computer Sciences Department, University of Wisconsin, Madison, Wisconsin, Tech. Rep. 88, 1970.

[36] M. Pătraşcu and M. Thorup, "Twisted tabulation hashing," in *Proc. 24th ACM/SIAM Symposium on Discrete Algorithms (SODA)*, 2013, pp. 209–228.

[37] R. J. Serfling, "Probability inequalities for the sum in sampling without replacement," *Annals of Statistics*, vol. 2, no. 1, pp. 39–48, 1974.

[38] M. Dietzfelbinger and P. Woelfel, "Almost random graphs with simple hash functions," in *Proc. 25th ACM Symposium on Theory of Computing (STOC)*, 2003, pp. 629–638.

[39] M. T. Goodrich and M. Mitzenmacher, "Invertible bloom lookup tables," in *2011 49th Annual Allerton Conference on Communication, Control, and Computing, Allerton Park & Retreat Center, Monticello, IL, USA, 28-30 September, 2011*, 2011, pp. 792–799.

[40] D. Eppstein, M. T. Goodrich, F. Uyeda, and G. Varghese, "What's the difference?: efficient set reconciliation without prior context," in *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 4. ACM, 2011, pp. 218–229.

[41] D. Eppstein and M. T. Goodrich, "Straggler identification in round-trip data streams via newton's identities and invertible bloom filters," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 23, no. 2, pp. 297–306, 2011.

[42] M. Mitzenmacher and G. Varghese, "Biff (bloom filter) codes: Fast error correction for large data sets," in *Information Theory Proceedings (ISIT), 2012 IEEE International Symposium on*. IEEE, 2012, pp. 483–487.