Journal of Computer and System Sciences 84 (2017) 171–184

Contents lists available at ScienceDirect

Journal of Computer and System Sciences

www.elsevier.com/locate/jcss

d-k-min-wise independent family of hash functions ☆

Guy Feigenblat [a,b,*], Ely Porat [a], Ariel Shiftan [a]

[a] Department of Computer Science, Bar-Ilan University, Ramat Gan 52900, Israel
[b] IBM Haifa Research Lab, Haifa University Campus, Haifa, Israel

ARTICLE INFO

Article history:
Received 19 October 2015
Received in revised form 15 September 2016
Accepted 23 September 2016
Available online 11 October 2016

Keywords:
min-wise hash functions
Data streams
Windowed data streams
Similarity
Rarity

ABSTRACT

In this paper we introduce a general framework that exponentially improves the space, degree of independence, and time needed by *min-wise-based* algorithms. The authors, in SODA '11 [1], introduced an exponential time improvement for *min-wise-based* algorithms. Here we develop an alternative approach that achieves both exponential time and exponential space improvement. The new approach relaxes the need for approximately *min-wise* hash functions, hence getting around the $\Omega(\log \frac{1}{\epsilon})$ independence lower bound, by defining and constructing a *d-k-min-wise* independent family of hash functions; surprisingly, for most cases, only 8-wise independence is needed for the additional improvement. Furthermore, we discuss how this construction can be used to improve many *min-wise-based* algorithms. To our knowledge such definitions, for hash functions, were never previously studied or constructed. Finally, we show how to apply it for similarity and rarity estimation over data streams; other *min-wise-based* algorithms can be adjusted in the same way.

1. Introduction

Hash functions are the fundamental building blocks of many algorithms; they map values from one domain to another, usually from larger to smaller domains. Although they have been studied for many years, designing hash functions is still a hot topic in modern research. In a perfect world, we could use a truly random hash function, chosen randomly out of all the possible mappings.

Specifically, consider the domain of all hash functions $h : U \to U'$, where $|U| = u$ and $|U'| = u'$. As we need to map each of the u elements in the source into one of the u' possible mappings, the number of bits needed to maintain each function is $u \log(u')$. Since we often have a massive amount of data to process, this amount of space is not feasible; nevertheless, most algorithms do not really need such a high level of randomness, and can perform well enough with some relaxations. In such cases, one can use a much smaller domain of hash functions. A smaller domain implies lower space requirements at the price of a lower level of randomness.

As an illustrative example, the notion of a *2-wise independent* family of hash functions assures the independence of each pair of elements; only $2\log(u')$ bits are enough in order to choose and maintain such a function out of the family. Generally, a family of hash function is said to be *k-wise independent*, if selecting a hash function at random from the family guarantees that the hash values of any k elements are independent random variables; in this case $O(k \log(u'))$ bits are required. One

☆ This research was carried out as part of the PhD thesis of Guy Feigenblat under the supervision of Professor Ely Porat from Bar-Ilan University.

* Corresponding author.
E-mail addresses: feigeng@cs.biu.ac.il (G. Feigenblat), porately@cs.biu.ac.il (E. Porat), shiftaa@cs.biu.ac.il (A. Shiftan).

http://dx.doi.org/10.1016/j.jcss.2016.09.005
0022-0000/

of the main techniques to construct such a family [2] is to use a polynomial of degree k. Therefore, as the degree of independence increases, the number of bits needed to represent a function out of the family increases as well, hence more space is required. In addition the time takes to evaluate the hash function increases as well.

This work is focused on the area of min-hashing. One derivative of min-hashing is *min-wise* independent permutations, which were first introduced by Broder et al. and Mulmuley [3,4]. A family of **permutations** $F \subseteq S_u$ (where S_u is the symmetric group) is **min-wise independent** if, for any set $X \subseteq [u]$ (where $[u] = \{0, \ldots, u-1\}$) and any $x \in X$, where π is chosen uniformly at random in F, we have:

$$Pr[\min\{\pi(X)\} = \pi(x)] = \frac{1}{|X|}$$

Similarly, a family of **functions** $\mathcal{H} \in [u] \to [u]$ (where $[u] = \{0, \ldots, u-1\}$) is called **min-wise independent** if, for any $X \subseteq [u]$ and any $x \in X$, where h is chosen uniformly at random in $\mathcal{H}$, we have:

$$Pr_{h \in \mathcal{H}}[\min\{h(X)\} = h(x)] = \frac{1}{|X|}$$

Min-hashing is a widely-used tool for solving problems in computer science, such as estimating similarity [4–6], rarity [7], transitive closure [8], web page duplicate detection [9–12], sketching techniques [13,14], and other data mining problems [15–18].

One of the key properties of min-hashing is that it enables one to sample the universe of the elements being hashed; this is because each element, over the random choice of hash functions from the family, has equal probability of being mapped to the minimal value, regardless of the number of occurrences of the element. Thus, by maintaining the element with the minimal hash value over the input, one can sample the universe.

Similarity, estimation of data sets is a fundamental tool in mining data, often calculated using the Jaccard similarity coefficient, which is defined by $\frac{|A \cap B|}{|A \cup B|}$, where A and B are two data sets. By maintaining the minimal hash value over two sets of data inputs A and B, the probability of getting the same hash value is exactly $\frac{|A \cap B|}{|A \cup B|}$, which equals the Jaccard similarity coefficient [4–6,8].

Indyk [19] was the first to give a construction of a space efficient approximately *min-wise* independent family of hash functions; another construction was proposed by Saks et al. [20]. A family of functions $\mathcal{H} \subseteq [u] \to [u]$ is called **approximately min-wise independent**, or ϵ-*min-wise* independent, if, for any $X \subseteq [u]$, and for any $x \in X$, where h is chosen uniformly at random in $\mathcal{H}$, we have:

$$Pr_{h \in \mathcal{H}}[\min\{h(X)\} = h(x)] = \frac{1}{|X|}(1 \pm \epsilon)$$

where $\epsilon \in (0, 1)$ is the desired error bound and $O(\log(\frac{1}{\epsilon}))$ independence is needed. Pătraşcu and Thorup showed [21] that $\Omega(\log \frac{1}{\epsilon})$ independence is needed for maintaining an approximately *min-wise* function; therefore proving that Indyk's construction is optimal. Later, Pătraşcu and Thorup showed a technique to generate an approximately *min-wise* hash function using simple tabulation function, with some limitations on ϵ.

In SODA '11 [1], the authors defined and gave a construction for a family of **approximately k-min-wise** ***(ϵ-k-min-wise)*** **independent** hash functions:

A family of functions $\mathcal{H} \subseteq [u] \to [u]$ (where $[u] = \{0 \ldots u-1\}$) is called ϵ-k-min-wise independent if, for any $X \subseteq [u]$ and for any $Y \subset X$, $k = |Y|$, we have

$$\Pr_{h \in \mathcal{H}}\left[\max_{y \in Y} h(y) < \min_{z \in X - Y} h(z)\right] = \frac{1}{\binom{|X|}{|Y|}}(1 \pm \epsilon),$$

where the function h is chosen uniformly at random from $\mathcal{H}$ and $\epsilon \in (0, 1)$ is the error bound.

Their main theorem states [1, Theorem 1.1] that choosing uniformly at random from a $O(k \log\log \frac{1}{\epsilon} + \log \frac{1}{\epsilon})$-wise independent family of hash functions is approximately *k-min-wise* independence, where $k = |Y|$ and $\epsilon \in (0, 1)$. Formerly, most *min-wise-based* applications used k different approximately *min-wise* independent hash functions, i.e., they maintained k different samples. The authors [1] proposed to use only one approximately *k-min-wise* independent hash function to maintain the k samples by using the k minimal hash values. By definition, the k minimal elements are fully independent and thus the estimators' precision can be preserved. Furthermore, the use of this function exponentially reduced the running time of previous *min-wise-based* algorithms. The authors offered a general framework and gave examples of how to apply it for estimating the similarity, rarity, and entropy of random graphs. In this paper, we take it a step forward and exponentially reduce the space and degree of independence needed by *min-wise-based* algorithms.

Porat and Bachrach [22,23] proposed a general technique for constructing fingerprints of massive data streams: the heart of their method lies in using a specific family of pseudo-random hashes shown to be approximately *min-wise* independent, where only one bit is needed to be maintained per function, i.e., one can store just a single bit rather than the full element IDs. Post publication of the conference version of this paper [24], Thorup [25] showed that set similarity can be estimated

using only 2-independent and bottom-k sampling, with a different method. The advantage of our approach is that even-though it requires a bit more independence, it allows a flexibility in choosing the level of independence between the k sampled elements.

Both approximately *min-wise* and *k-min-wise* hash functions use a low degree of independence (and hence potentially low memory and runtime) and are therefore applicable for estimating various metrics in the data stream models. In the unbounded data stream model, we consider a stream in which elements arrive sequentially. Due to the size of the stream, only one pass over the data is allowed; furthermore, the storage available is poly-logarithmic in the size of the stream. In the windowed data stream model, we consider a predefined size window of size N over the stream, such that all the queries to the stream are related to elements in the current window. Similar to the unbounded streaming model, we are only allowed one pass over the data and the storage available is poly-logarithmic in the size of the window.

1.1. Our contribution

In this paper, we propose a new approach that 'closes the gap' and exponentially reduces the space, degree of independence, and the time needed by *min-wise-based* algorithms in addition to the exponential time improvement achieved in SODA '11 [1]. We do this by defining and constructing a small approximately *d-k-min-wise* independent family of hash functions. The construction shows the degree of independence needed to represent such a family, for given d and k. As will be discussed here, many *min-wise-based* estimators can be adjusted to use our construction, and exponentially reduce the space and degree of independence consumed.

First, we extend the notion of a *min-wise* independent family of hash functions by defining a *d-k-min-wise* independent family of hash functions. We then show a construction of such a family. Finally, as a usage example, we show how to apply those hash functions to the estimation of similarity and rarity over data streams.

Under our definition for a *d-k-min-wise* hash function, all subsets of size d of any fixed set X have an equal probability to have hash values among the minimal k values in X, where the probability is over the random choice of hash function from the family (the formal definition is given in section 2). The degree of independence and the space needed by our construction are constant. The lack of dependency on k is surprising, but the intuition behind that is the stability property of the k-th ranked element for large enough k. Hence, the randomness needed by the function is mainly for the independence of the d elements.

We argue that, for most applications, it is sufficient to use constant $d = 2$. This yields the need of only 8-wise independent hash functions, which can be implemented efficiently in practice. Our innovative approach gets around the $\Omega(\log \frac{1}{\epsilon})$ lower bound [21] of approximately *min-wise* functions, as our family, by definition, does not have to be approximately *min-wise* independent.

To utilize our construction, we propose a simple and general framework for the exponential space and degree of independence improvement of *min-wise-based* algorithms, such as in [7–18,26–28]. Formerly, *min-wise-based* algorithms used either k-independent approximately *min-wise* hash functions (which can be implemented using either [19] or [29]) or one approximately *k-min-wise* independent function [1] to sample k independent elements from the universe; even if we use tabulation [29], we would still need k independent instances of it, hence a multiplicative factor of $O(k)$ in independence, space, and time. The *k-min-wise* technique exponentially improved the time needed by *min-wise-based* applications; here, we take it a step forward by relaxing the need for k independent samples. We propose to use a much lesser degree of independence, specifically only a constant degree, and amplify the precision using probabilistic techniques. In comparison to the technique used by Porat and Bachrach [22,23], we use more space (as we maintain the elements' IDs), but our running time is better by more than a factor of $O(\log \frac{1}{\epsilon})$.

At a high level, we propose to use several independent, approximately *d-k-min-wise* independent functions, where each samples less than k elements (where k is the same as in *k-min-wise*). The elements sampled by each function are d-wise independent; therefore, we can use Chebyshev's inequality to bound the precision. Specifically, a pair-wise function is sufficient for applying Chebyshev, and this is why $d = 2$ should be used. By taking the median out of the independent samples, using the Chernoff bound, the precision is amplified. The above procedure does not change the algorithm itself, but only the way it samples; hence it is simple to adapt. This exponentially improves the space and the degree of independence (as it is constant for each function), while maintaining the exponential time improvement [1].

This approach can be applied in cases where the original estimators' values are numeric. In these cases, it is possible to take the average and median of the sampled values and aggregate them (as described above). The estimators' values in most of the applications we considered were indeed numeric, but for the other cases in which this technique cannot be applied, one can still achieve the exponential time improvement by using *k-min-wise* functions.

As illustrative examples, we propose algorithms which utilize the above framework for similarity and rarity; see Table 1.1 for a comparison of results.

1.2. Outline

In section 2, we define the notion of *d-k-min-wise* and approximately *d-k-min-wise* independent families. Later, in the first part of section 3, we give an outline and the intuition behind the approximately *d-k-min-wise* construction, which

Table 1.1
Similarity and rarity algorithms comparison in the unbounded data stream model. Time complexity is the expected per item observed, and space is given in words, in upper bounds, for the constant failure probability.

	[7,30]	[1]	This paper
Algorithm's hashing time	$\frac{1}{\epsilon^2}\log\frac{1}{\epsilon}$	$\log^2(\frac{1}{\epsilon^2}\log\log\frac{1}{\epsilon})$	$O(1)$
Additional algorithm's time	$\frac{1}{\epsilon^2}$	$\log\frac{1}{\epsilon^2}$	$O(1)$
Space for storing functions	$\frac{1}{\epsilon^2}\log\frac{1}{\epsilon}$	$\frac{1}{\epsilon^2}\log\log\frac{1}{\epsilon}$	$O(1)$
Space used by algorithm	$\frac{1}{\epsilon^2}$	$\frac{1}{\epsilon^2}$	$\frac{1}{\epsilon^2}$

is given in detail afterwards. In section 4, we present two usage examples of the framework. In section 5, we conclude and propose future work. Finally, the appendix contains lemmata needed for the completeness of the construction given in section 3.

2. Definitions and notations

d-k-min-wise independent families of hash functions are generalizations of *min-wise* and *k-min-wise* independent families of hash functions. Informally, under Definition 1 below, for any disjoint subsets of the domain X and Y, where $|X| > k \gg |Y| = d$, the elements of Y have an equal probability to have hash values among the minimal k values in $X \cup Y$. The probability is over the random choice of hash function from the family, i.e., all hash values of Y are less than the $k-d+1$ ranked hash value in X. This property measures the randomness of the family, as choosing a truly random function, obviously, satisfies the definition for $d = k = |X|$. In addition, we let $\Pr[\cdot]$ denote a fully random probability measure over $[u] \to [u]$, and let $\Pr_l[\cdot]$ denote any l-wise independent probability measure over the same domain, for $l \in \mathbb{N}$. Finally, let $t = k - d + 1$ and $m = n - d = |X|$.

For the rest of this paper, for any set X, we denote $MIN_k(X)$ to be the set of k smallest elements in X, and $RANK_k(X)$ to be the k-th element in X, where the elements are sorted by value. In addition, for any set X and hash function h, we denote $h(X)$ to be the set of all hash values of all elements in X. Finally, we denote $[u]$ to be the universe from which the elements are drawn; we choose $u \gg |X|$. We use these notations to define the following:

Definition 1. A family of functions $\mathcal{H} \subseteq [u] \to [u]$ (where $[u] = \{0 \dots u-1\}$) is called *d-k-min-wise* independent if, for any $X \subseteq [u]$, $|X| = n - d$ and for any $Y \subseteq [u]$, $|Y| = d$, $X \cap Y = \emptyset$, $d \leq k$, we have

$$\Pr_{h \in \mathcal{H}}[RANK_d(h(Y)) < RANK_{k-d+1}(h(X))] = \frac{\binom{k}{d}}{\binom{n}{d}}$$

where the function h is chosen uniformly at random from $\mathcal{H}$.

For cases where we allow a small error, the respective definition is:

Definition 2. A family of functions $\mathcal{H} \subseteq [u] \to [u]$ (where $[u] = \{0 \dots u-1\}$) is called approximately *d-k-min-wise* independent if, for any $X \subseteq [u]$, $|X| = n - d$ and for any $Y \subseteq [u]$, $|Y| = d$, $X \cap Y = \emptyset$, $d \leq k$, we have

$$\Pr_{h \in \mathcal{H}}[RANK_d(h(Y)) < RANK_{k-d+1}(h(X))] = \frac{\binom{k}{d}}{\binom{n}{d}}(1 \pm \epsilon)$$

where the function h is chosen uniformly at random from $\mathcal{H}$, and $\epsilon \in (0, 1)$ is the error bound.

3. The *d-k-min-wise* construction

3.1. Construction outline

In this section, we present the outline of the construction, which should give the reader the essence of the construction; subsequent sections delve into the full technical details. The construction shows the degree of independence needed to represent such a family, for given d and k.

The main intuition behind the construction is that high-enough ranked elements are relatively stable, as opposed to lower-ranked elements. As an example, consider the minimal element which is known to be not very stable; in contrast, we show that the probability of a highly-ranked element to deviate from its expected location decreases rapidly as the deviation increases. By definition, our goal is to show that every d elements have almost the same probability to be among the k minimal elements. By utilizing the stability property and using large enough k, we show that the amount of independence needed, i.e., l, is surprisingly only $O(d)$. The relationship between d, k and the amount of independence needed are given in the detailed construction section.

We start by showing that, in the fully random case, the probability for the hash values of any d elements to be within the k minimal values, is

$$\Pr[h(y_1), h(y_2), \ldots, h(y_d) < RANK_{k-d+1}(h(X))] = \frac{\binom{k}{d}}{\binom{n}{d}} = \frac{k}{n}\frac{k-1}{n-1}\cdots\frac{k-d+1}{n-d+1}$$

which yields that a totally random function is *d-k-min-wise* independent. We next need to find the amount of independence needed for any l-wise independent ($l \geq 1$) family of hash functions, in order to be close enough (within a multiplicative factor of some $\epsilon \in (0,1)$) to this property. In other words, for the chosen amount of independence, we will show that the difference between the random case and l-wise independent case is $\epsilon\frac{\binom{k}{d}}{\binom{n}{d}}$.

Specifically, we will divide the universe of elements into a set ϕ of non-overlapping blocks, b_i, for $i \in \mathbb{Z}$. We construct the blocks such that the boundaries of b_0 are roughly around the expected location of the $(k-d+1)$-th hash value in X. For each block, we will estimate the probability that the $(k-d+1)$-th hash value in X, i.e., $RANK_{k-d+1}(h(X))$, falls within this block's boundaries. Based on the complete probability formula, the probability $\Pr[h(y_1), h(y_2), \ldots, h(y_d) < RANK_{k-d+1}(h(X))]$ in the l-wise case is

$$\sum_{i\in\phi} \Pr_l[RANK_{k-d+1}(h(X)) \in b_i] \cdot \Pr_l[h(y_1), \ldots, h(y_d) \leq RANK_{k-d+1}(h(X)) \mid RANK_{k-d+1}(h(X)) \in b_i]$$

The construction yields a *d-k-min-wise* independent family if we show that the difference between the fully random case and the above is $\epsilon\frac{\binom{k}{d}}{\binom{n}{d}}$.

We aim to find the appropriate values of l and k sufficient to have the probability $\Pr_l[RANK_{k-d+1}(h(X)) \in b_i]$ decrease polynomially (roughly $\frac{1}{|i|^{d+1}}$) for each block b_i. Then, by adding to l other d degrees of independence, we can assume the elements of Y are randomly distributed, which is utilized to bound $\Pr_l[h(y_1), \ldots, h(y_d) \leq RANK_{k-d+1}(h(X))]$. Eventually, we use both to bound the difference and show that it is within the allowed error.

3.2. Construction in details

In this section, we provide a construction for an approximately *d-k-min-wise* independent family of hash functions. We use the notions defined in section 2 and divide the universe into a set ϕ of non-overlapping blocks, which will be defined in the next section.

Lemma 3.1.

$$\Pr[h(y_1), h(y_2), \ldots, h(y_d) < RANK_{k-d+1}(h(X))] = \frac{k}{n}\frac{k-1}{n-1}\cdots\frac{k-d+1}{n-d+1}$$

Proof. Consider n ordered elements divided into two groups — one of size $n-d$, and the other of size d. The number of possible locations of the d elements is $\binom{n}{d}$. There are $\binom{k}{d}$ possible locations in which the d elements are among the k smallest elements. Hence, the probability for the d element to be among the k-th smallest elements is:

$$\Pr[h(y_1), h(y_2), \ldots, h(y_d) < RANK_{k-d+1}(h(X))] = \frac{\binom{k}{d}}{\binom{n}{d}} = \frac{k}{n}\frac{k-1}{n-1}\cdots\frac{k-d+1}{n-d+1} \qquad \square$$

Since the blocks in ϕ are non-overlapping $\sum_{i\in\phi}\Pr_l[RANK_{k-d+1}(h(X)) \in b_i] = 1$, using Lemma 3.1, we get

$$\Pr[h(y_1), h(y_2), \ldots, h(y_d) < RANK_{k-d+1}(h(X))] = \frac{k}{n}\frac{k-1}{n-1}\cdots\frac{k-d+1}{n-d+1}\sum_{i\in\phi}\Pr_l[RANK_{k-d+1}(h(X)) \in b_i]$$

Lemma 3.2. *Let d, k, ϵ, n, and $h : [u] \to [u]$ be as in Definition 2, and denote*

$$\Delta = \sum_{i\in\phi}\Pr_l[RANK_{k-d+1}(h(X)) \in b_i] \times \left[\Pr_l[h(y_1), \ldots, h(y_d) \leq RANK_{k-d+1}(h(X)) \mid RANK_{k-d+1}(h(X)) \in b_i] - \frac{\binom{k}{d}}{\binom{n}{d}}\right]$$

Any family of l-wise independent hash functions is approximately d-k-min-wise independent if

$$-\epsilon\frac{\binom{k}{d}}{\binom{n}{d}} \leq \Delta \leq \epsilon\frac{\binom{k}{d}}{\binom{n}{d}}$$

Proof. Based on the complete probability formula, in the l-wise independent case

$$\Pr_l[h(y_1), h(y_2), \ldots, h(y_d) < RANK_{k-d+1}(h(X))] =$$
$$\sum_{i\in\phi}\Pr_l[RANK_{k-d+1}(h(X)) \in b_i]\cdot$$
$$\Pr_l[h(y_1), h(y_2), \ldots, h(y_d) < RANK_{k-d+1}(h(X)) \mid RANK_{k-d+1}(h(X)) \in b_i]$$

By definition, any family of l-wise independent hash functions is approximately *d-k-min-wise* independent if

$$\Pr_l[h(y_1), h(y_2), \ldots, h(y_d) < RANK_{k-d+1}(h(X))] = \frac{\binom{k}{d}}{\binom{n}{d}}(1\pm\epsilon)$$

which is satisfied if $-\epsilon\frac{\binom{k}{d}}{\binom{n}{d}} \leq \Delta \leq \epsilon\frac{\binom{k}{d}}{\binom{n}{d}}$. □

3.3. Blocks partitioning

We divide the universe $[0, |U|]$ into non-overlapping blocks. We construct the blocks around $\frac{t|U|}{m}$, where t is as defined in section 2:

$$\text{for } i\in\mathbb{Z},\ b_i = \left[(1+\epsilon(i-1))\tfrac{t|U|}{m}, (1+\epsilon i)\tfrac{t|U|}{m}\right),\ \text{e.g.,}$$
$$\ldots, b_{-1} = \left[(1-2\epsilon)\tfrac{t|U|}{m}, (1-\epsilon)\tfrac{t|U|}{m}\right),\ b_0 = \left[(1-\epsilon)\tfrac{t|U|}{m}, \tfrac{t|U|}{m}\right),\ b_1 = \left[\tfrac{t|U|}{m}, (1+\epsilon)\tfrac{t|U|}{m}\right), \ldots$$

By the blocks' partitioning and according to Definition 2, the expected number of hash values in X, below the upper boundary of block b_0, is t. This will be utilized for estimating the probability of any d elements in Y to be within the smallest k elements in $X\cup Y$ (below the t-th ranked element in X).

We refer to blocks b_i for $i>0$ as 'positive blocks' and for $i\leq 0$ as 'negative blocks'. For the rest of the paper, we ignore blocks which are outside $[0, |U|]$.

3.4. Bounding $\Pr_l[RANK_t(h(X)) \in b_i]$

We now bound the probability that the t-th ranked element's hash value in X falls into block b_i. We show that the probability decreases polynomially with the growth of $|i|$. For convenience, we separate the bound for the negative and positive blocks, and we specifically use 1 as an upper bound for the probabilities of b_0 and b_1. In addition, we bound $\Pr_l[RANK_t(h(X)) \in b_i]$, for $i>1$, with the probability of the t-th ranked element's hash value falling within any block greater than i, i.e., $\Pr_l\left[\cup_{j=i}^{\infty} RANK_t(h(X)) \in b_j\right]$. For $i<0$, we do a similar procedure, by bounding it with $\Pr_l\left[\cup_{j=-\infty}^{i} RANK_t(h(X)) \in b_j\right]$.

Lemma 3.3. *For $i>1$, constant d, $\epsilon\in(0,1)$, $k > d + 2\cdot 8^{\frac{2}{l}}\frac{(6l)^{1+\frac{1}{l}}}{\epsilon^2} - 1$ and $l = 2d+2$:*

$$\Pr_l[RANK_t(h(X)) \in b_i] \leq \Pr_l\left[\cup_{j=i}^{\infty} RANK_t(h(X)) \in b_j\right] \leq \frac{1}{(i-1)^{d+1}}$$

Proof. For block b_i, $X=\{x_1,\ldots,x_m\}$ we define Z_j to be the following indicator variable:

$$Z_j = \begin{cases} 1 & h(x_j) < (1+\epsilon(i-1))\frac{t|U|}{m} \\ 0 & otherwise \end{cases}$$

In addition, we define $Z=\sum_j Z_j$, and E_i to be the expected value of Z. Because Z is a sum of indicator variables, $E_i = (1+\epsilon(i-1))\frac{t}{m}(m) = t(1+\epsilon(i-1))$.

We use the above definitions to show that

$$\begin{aligned}\Pr_l[RANK_t(h(X)) \in b_i] &\leq \Pr_l\left[\cup_{j=i}^{\infty} RANK_t(h(X)) \in b_j\right] \\ &\leq \Pr_l[\text{number of hash values smaller than the lower boundary of block } b_i < t] \\ &\leq \Pr_l[Z < t] \\ &\leq \Pr_l[E_i - Z \geq E_i - t] \\ &\leq \Pr_l[|E_i - Z| \geq E_i - t] \\ &\leq \Pr_l[|Z - E_i| \geq t(1+\epsilon(i-1)) - t]\end{aligned}$$

Using Markov's inequality, as l is even:

$$\Pr_l[|Z - E_i| \geq t\epsilon(i-1)] \leq \frac{E(|Z-E_i|^l)}{[t\epsilon i]^l}$$

We use the following from Lemma A.1:

$$E(|Z-E_i|^l) \leq 8(6l)^{\frac{l+1}{2}}(E_i)^{\frac{l}{2}}$$

Thus,

$$\Pr_l[RANK_t(h(X)) \in b_i] \leq \frac{8(6l)^{\frac{l+1}{2}}(t(1+\epsilon(i-1)))^{\frac{l}{2}}}{[t\epsilon(i-1)]^l} = \frac{8(6l)^{\frac{l+1}{2}}(1+\epsilon(i-1))^{\frac{l}{2}}}{t^{\frac{l}{2}}[\epsilon(i-1)]^l}$$

Note that $t > 2\cdot 8^{\frac{2}{l}}\frac{(6l)^{1+\frac{1}{l}}}{\epsilon^2}$ for $t = k-d+1$, and proceed as follows:

$$\Pr_l[RANK_t(h(X)) \in b_i] \leq \frac{8(6l)^{\frac{l+1}{2}}(1+\epsilon(i-1))^{\frac{l}{2}}}{[2\cdot 8^{\frac{2}{l}}\frac{(6l)^{1+\frac{1}{l}}}{\epsilon^2}]^{\frac{l}{2}}[\epsilon(i-1)]^l} = \frac{8(6l)^{\frac{l+1}{2}}(1+\epsilon(i-1))^{\frac{l}{2}}}{2^{\frac{l}{2}}\cdot 8\cdot\frac{(6l)^{\frac{l+1}{2}}}{\epsilon^l}[\epsilon(i-1)]^l} =$$

$$\frac{(1+\epsilon(i-1))^{\frac{l}{2}}}{2^{\frac{l}{2}}(i-1)^l} = \frac{(1+\epsilon(i-1))^{\frac{l}{2}}}{(2(i-1)^2)^{\frac{l}{2}}} = \left(\frac{1}{2(i-1)^2} + \frac{\epsilon}{2(i-1)}\right)^{\frac{l}{2}} \leq \left(\frac{1}{i-1}\right)^{\frac{l}{2}}$$

As $l = 2d+2$ is defined, we get:

$$\Pr_l[RANK_t(h(X)) \in b_i] \leq \left(\frac{1}{i-1}\right)^{d+1} \qquad \square$$

The proof for the matching lemma for negative blocks is similar and thus is deferred to the appendix.

Lemma 3.4. *For $i > 0$, constant d, $\epsilon \in (0,1)$, $k > d + 2\cdot 8^{\frac{2}{l}}\frac{(6l)^{1+\frac{1}{l}}}{\epsilon^2} - 1$ and $l = 2d+2$:*

$$\Pr_l[RANK_t(h(X)) \in b_{-i}] \leq \Pr_l\left[\cup_{j=-\infty}^{-i} RANK_t(h(X)) \in b_j\right] \leq \frac{1}{i^{d+1}}$$

Proof. See appendix. $\square$

3.5. Bounding Δ

In this section, we prove the upper and lower bounds of Lemma 3.2, i.e., that $-\epsilon\frac{\binom{k}{d}}{\binom{n}{d}} \leq \Delta \leq \epsilon\frac{\binom{k}{d}}{\binom{n}{d}}$.

Lemma 3.5. *For constant d, $\epsilon \in (0,1)$ (specifically $d < \frac{1}{\epsilon}$), $k = O(\frac{1}{\epsilon^2})$, $l \geq 2d+2$, and using $(l+d)$ independence:*

$$|\Delta| \leq 6\cdot 2^d d^{\frac{d+1}{2}}\epsilon\frac{\binom{k}{d}}{\binom{n}{d}}$$

Proof.

$$|\Delta| = \left| \sum_{i=-\infty}^{\infty} \Pr_{l+d}[RANK_t(h(X)) \in b_i] \times \left[\Pr_{l+d}[h(y_1), h(y_2), \ldots, h(y_d) \leq RANK_t(h(X)) \mid RANK_t(h(X)) \in b_i] - \frac{\binom{k}{d}}{\binom{n}{d}} \right] \right|$$

Using d degrees of independence (out of $l+d$) for $h(y_1), h(y_2), \ldots, h(y_d)$, we can assume the elements of Y are randomly distributed:

$$\begin{aligned}
|\Delta| &= \left| \sum_{i=-\infty}^{\infty} \Pr_{l}[RANK_t(h(X)) \in b_i] \left[(\frac{t}{m})^d (1+\epsilon i)^d - \frac{\binom{k}{d}}{\binom{n}{d}} \right] \right| \\
&= \left| \sum_{i=-\infty}^{0} \left(\Pr_{l}\left[\cup_{j=-\infty}^{i} RANK_t(h(X)) \in b_j \right] - \Pr_{l}\left[\cup_{j=-\infty}^{i-1} RANK_t(h(X)) \in b_j \right] \right) \right. \\
&\quad \left. + \sum_{i=1}^{\infty} \left(\Pr_{l}\left[\cup_{j=i}^{\infty} RANK_t(h(X)) \in b_j \right] - \Pr_{l}\left[\cup_{j=i+1}^{\infty} RANK_t(h(X)) \in b_j \right] \right) \right] \times \\
&\qquad \left[(\frac{t}{m})^d (1+\epsilon i)^d - \frac{\binom{k}{d}}{\binom{n}{d}} \right] \Bigg|
\end{aligned}$$

By expanding the sum, and factoring out the probabilities for each i, the $\frac{\binom{k}{d}}{\binom{n}{d}}$ factors are eliminated in all cases excluding the tails. In addition, note that the probabilities are always non-negative, and therefore we can remove the absolute function surrounding them. We now get the following:

$$\begin{aligned}
|\Delta| \leq & \sum_{i=-\infty}^{-1} \Pr_{l}\left[\cup_{j=-\infty}^{i} RANK_t(h(X)) \in b_j \right] \left| (\frac{t}{m})^d (1+\epsilon i)^d - (\frac{t}{m})^d (1+\epsilon(i+1))^d \right| \\
& + \Pr_{l}\left[\cup_{j=-\infty}^{0} RANK_t(h(X)) \in b_j \right] \left| (\frac{t}{m})^d - \frac{\binom{k}{d}}{\binom{n}{d}} \right| \\
& + \Pr_{l}\left[\cup_{j=1}^{\infty} RANK_t(h(X)) \in b_j \right] \left| (\frac{t}{m})^d (1+\epsilon)^d - \frac{\binom{k}{d}}{\binom{n}{d}} \right| \\
& + \sum_{i=2}^{\infty} \Pr_{l}\left[\cup_{j=i}^{\infty} RANK_t(h(X)) \in b_j \right] \left| (\frac{t}{m})^d (1+\epsilon i)^d - (\frac{t}{m})^d (1+\epsilon(i-1))^d \right|
\end{aligned}$$

Applying Lemmas 3.3 and 3.4, and upper bounding the probabilities

$$\Pr_{l}\left[\cup_{j=-\infty}^{0} RANK_t(h(X)) \in b_j \right] \text{ and } \Pr_{l}\left[\cup_{j=1}^{\infty} RANK_t(h(X)) \in b_j \right] \text{ with } 1,$$

we get:

$$\begin{aligned}
|\Delta| \leq & \sum_{i=0}^{\infty} \frac{1}{|i+1|^{d+1}} \left| (\frac{t}{m})^d (1-\epsilon i)^d - (\frac{t}{m})^d (1-\epsilon(i+1))^d \right| \\
& + \left| (\frac{t}{m})^d - \frac{\binom{k}{d}}{\binom{n}{d}} \right| \\
& + \left| (\frac{t}{m})^d (1+\epsilon)^d - \frac{\binom{k}{d}}{\binom{n}{d}} \right| \\
& + \sum_{i=2}^{\infty} \frac{1}{(i-1)^{d+1}} \left| (\frac{t}{m})^d (1+\epsilon i)^d - (\frac{t}{m})^d (1+\epsilon(i-1))^d \right|
\end{aligned}$$

Denote $\alpha = (\frac{n}{n-d} \cdot \frac{k-d+1}{k})(\frac{n-1}{n-d} \cdot \frac{k-d+1}{k-1}) \cdots (\frac{n-d+1}{n-d} \cdot \frac{k-d+1}{k-d+1})$, such that $\alpha \cdot \frac{\binom{k}{d}}{\binom{n}{d}} = (\frac{k-d+1}{n-d})^d = (\frac{t}{m})^d$. We continue as follows:

$$
\begin{aligned}
|\Delta| &\leq \sum_{i=0}^{\infty} \frac{1}{|i+1|^{d+1}} \left| \alpha \cdot \frac{\binom{k}{d}}{\binom{n}{d}} (1-\epsilon i)^d - \alpha \cdot \frac{\binom{k}{d}}{\binom{n}{d}} (1-\epsilon(i+1))^d \right| \\
&\quad + \left| \alpha \cdot \frac{\binom{k}{d}}{\binom{n}{d}} - \frac{\binom{k}{d}}{\binom{n}{d}} \right| \\
&\quad + \left| \alpha \cdot \frac{\binom{k}{d}}{\binom{n}{d}} (1+\epsilon)^d - \frac{\binom{k}{d}}{\binom{n}{d}} \right| \\
&\quad + \sum_{i=2}^{\infty} \frac{1}{(i-1)^{d+1}} \left| \alpha \cdot \frac{\binom{k}{d}}{\binom{n}{d}} (1+\epsilon i)^d - \alpha \cdot \frac{\binom{k}{d}}{\binom{n}{d}} (1+\epsilon(i-1))^d \right| \\
&\leq \alpha \cdot \frac{\binom{k}{d}}{\binom{n}{d}} \sum_{i=0}^{\infty} \frac{1}{|i+1|^{d+1}} \left| (1-\epsilon i)^d - (1-\epsilon(i+1))^d \right| \\
&\quad + \frac{\binom{k}{d}}{\binom{n}{d}} |\alpha - 1| \\
&\quad + \frac{\binom{k}{d}}{\binom{n}{d}} \left| \alpha(1+\epsilon)^d - 1 \right| \\
&\quad + \alpha \cdot \frac{\binom{k}{d}}{\binom{n}{d}} \sum_{i=2}^{\infty} \frac{1}{(i-1)^{d+1}} \left| (1+\epsilon i)^d - (1+\epsilon(i-1))^d \right| \\
&\leq \alpha \cdot \frac{\binom{k}{d}}{\binom{n}{d}} \sum_{i=0}^{\infty} \frac{1}{|i+1|^{d+1}} \left| d^{\frac{d+1}{2}} (i^{d-1})\epsilon \right| \\
&\quad + \frac{\binom{k}{d}}{\binom{n}{d}} |\alpha - 1| \\
&\quad + \frac{\binom{k}{d}}{\binom{n}{d}} \left| \alpha(1+\epsilon)^d - 1 \right| \\
&\quad + \alpha \cdot \frac{\binom{k}{d}}{\binom{n}{d}} \sum_{i=2}^{\infty} \frac{1}{(i-1)^{d+1}} \left| d^{\frac{d+1}{2}} (i^{d-1})\epsilon \right|
\end{aligned}
$$

By definition $\epsilon d < 1$, and $(1-\epsilon d) < \alpha < (1+\epsilon d)$ by Lemma A.2, therefore $\alpha < 2$.

$$
\begin{aligned}
&\leq 2 \cdot \frac{\binom{k}{d}}{\binom{n}{d}} \sum_{i=0}^{\infty} \frac{1}{|i+1|^{d+1}} (d^{\frac{d+1}{2}} (i^{d-1})\epsilon) \\
&\quad + \frac{\binom{k}{d}}{\binom{n}{d}} (d\epsilon) \\
&\quad + \frac{\binom{k}{d}}{\binom{n}{d}} (d^{\frac{d+1}{2}}\epsilon + d^{\frac{d+3}{2}}\epsilon^2 + \epsilon d) \\
&\quad + 2 \cdot \frac{\binom{k}{d}}{\binom{n}{d}} \sum_{i=2}^{\infty} \frac{1}{(i-1)^{d+1}} (d^{\frac{d+1}{2}} (i^{d-1})\epsilon) \\
&\leq 4 \cdot \frac{\binom{k}{d}}{\binom{n}{d}} (d^{\frac{d+1}{2}}\epsilon) \\
&\quad + \frac{\binom{k}{d}}{\binom{n}{d}} (2d\epsilon)
\end{aligned}
$$

$$+\frac{\binom{k}{d}}{\binom{n}{d}}(2d^{\frac{d+1}{2}}\epsilon)$$
$$+4\cdot\frac{\binom{k}{d}}{\binom{n}{d}}(2^d d^{\frac{d+1}{2}}\epsilon)$$
$$\leq 6\cdot\frac{\binom{k}{d}}{\binom{n}{d}}(2^d d^{\frac{d+1}{2}}\epsilon) \qquad \square$$

We conclude with the following theorem:

Theorem 3.1. *For constant d, $\epsilon \in (0,1)$, $k = O(\frac{1}{\epsilon^2})$, $l \geq 2d+2$, any $(l+d)$-wise independent family of hash functions is approximately d-k-min-wise (ϵ-d-k-min-wise).*

Proof. To conclude the proof apply Lemma 3.5 to Lemma 3.2, and set up ϵ in Lemma 3.5 to be $\frac{\epsilon}{6\cdot 2^d d^{\frac{d+1}{2}}}$. $\square$

4. General framework for *min-wise-based* algorithms

In this section, we propose a general framework that utilizes the construction of *2-k-min-wise* (d-k-min-wise with $d=2$) functions for improving many *min-wise-based* algorithms, such as in [7–18,26–28]. As mentioned before, *min-wise* enables us to sample elements from the universe, i.e., sample such that each element has an equal probability for being sampled while ignoring repetitions. Common practice was to use some k-independent, approximately *min-wise* hash functions where each samples one element. Usually k depends on the error bound $\epsilon \in (0,1)$ and the failure probability $\tau \in (0,1)$, among other constraints. The drawback is that these functions require above a constant degree of independence, which impacts both time and space [19,21]. In SODA '11 [1], a new sampling method was proposed in which, instead of using and maintaining k different functions, they used only one *k-min-wise* independent function; this exponentially improved the time needed for sampling in various *min-wise-based* applications. Here, we take it a step forward by relaxing the need for k independent samples. Our technique uses a much lesser degree of independence, specifically only a constant degree per function, and amplifies the precision by probabilistic techniques.

We propose a procedure that does not change the algorithm itself, but only the way it samples; thus it is simple to adapt. We found that *2-k-min-wise* independent functions are sufficient to preserve the precision. In detail, one can use $O(\log\frac{1}{\tau})$ approximately *2-k-min-wise* independent functions, where each samples $\frac{k}{\log\frac{1}{\tau}}$ elements (where k is the same as in *k-min-wise*). For cases where the original *min-wise-based* estimator values are numeric, one can use the original *min-wise-based* estimator on each sampled element, and then average the values using Chebyshev's inequality. Chebyshev's inequality can be applied since the elements are *2-wise* independent. Next, the precision is amplified by taking the median out of the $\log\frac{1}{\tau}$ groups, and by using the Chernoff bound the precision becomes as desired. This procedure exponentially improves the space and time complexity (as the space for each function is constant). For cases where the original *min-wise-based* estimator values are not numeric, and therefore averaging and taking the median is not applicable, the *k-min-wise* technique is still valuable for the exponential time improvement. As illustrative examples, the rest of this section describes algorithms which utilize the above framework for estimating similarity and rarity.

4.1. 2-k-min-wise estimator for similarity

One of the possible uses of our framework is similarity estimation of two data streams. As mentioned in the introduction, the problem was studied by Datar et al. [7] and later in [1]. The use of our construction exponentially improves the space and the degree of independence of currently known results. We will now present two algorithms for solving the problem, in the unbounded and in the windowed data stream models. The technique we use is general and can be utilized to improve many *min-wise-based* algorithms, since most of them handle the *min-wise* functions similarly.

Based on the *k-min-wise* estimator from [30] (which is also formally defined in [1]), we construct a *2-k-min-wise* estimator. Let $h_1(A), h_2(A), \ldots, h_k(A)$ and $h_1(B), h_2(B), \ldots, h_k(B)$ be k pair-wise independent min hash values for the sets A and B, respectively, and $h_{1\ldots k}(A)$ be the set containing $h_1(A), h_2(A), \ldots, h_k(A)$. In addition, let $S(A,B)$ be the similarity of the two sets. We estimate the similarity by running the following procedure $\log\tau^{-1}$ times, and choosing the median value of

$$\hat{S}(A,B) = \frac{|\, h_{1\ldots k}(A) \cap h_{1\ldots k}(B) \cap h_{1\ldots k}(A\cup B)\,|}{k},\ 0<\epsilon<1, 0<\tau<1, k\geq 2\epsilon^{-2}$$

Theorem 4.1. *For an error bound ϵ with success probability at least $1-\tau$:*

$$\hat{S}(A,B)_{\text{median}} \in \frac{|A\cap B|}{|A\cup B|} \pm \epsilon$$

Proof. Let x_i be an indicator variable which equals 1 if the i-th element in $h_{1\ldots k}(A \cup B)$ exists in $h_{1\ldots k}(A) \cap h_{1\ldots k}(B)$, and 0 otherwise. In addition, we define $X = \sum_{i=1}^{k} x_i$. As the indicator variables are pair-wise independent, we can use Chebyshev's inequality to bound the failure probability of $\hat{S}(A, B)$.

$$\Pr\left(|X - k \cdot S(A,B)| \geq k\epsilon\right) \leq \frac{k \cdot S(A,B) \cdot (1 - S(A,B))}{k^2\epsilon^2} \leq \frac{1}{4k\epsilon^2} \leq \frac{1}{4}$$

We next define z_j to be an indicator variable such that $z_j = 1$ if the j-th iteration of $\hat{S}(A, B)$ failed, and 0 otherwise: $Z = \sum_{j=1}^{\log \frac{1}{\tau}} z_i$. Since the $\log \tau^{-1}$ values of $\hat{S}(A, B)$ are independent, we can apply the Chernoff bound to increase the success probability. The following is the overall failure probability of the estimator:

$$\Pr\left(Z > \frac{\log \frac{1}{\tau}}{2}\right) < \left(\frac{e}{2^2}\right)^{\frac{\log \frac{1}{\tau}}{4}} < \left(\frac{2^2}{e}\right)^{\frac{-\log \frac{1}{\tau}}{4}} < \left(\frac{2^2}{e}\right)^{\frac{\log \tau}{4}} < \tau \qquad \square$$

4.2. Unbounded data stream algorithm

We now present a similarity estimation algorithm in the unbounded data stream model. In this model, we consider a stream allowing only one pass over the data, and the storage available is poly-logarithmic in the size of the stream. Our algorithm uses the *2-k-min-wise* estimator, with $k = 2\epsilon^{-2}$, and runs it $\log \frac{1}{\tau}$ times, as described above.

For each iteration of the estimator, we randomly choose a function from an approximately *2-k-min-wise* independent family of hash functions, using a family of constant degree polynomials over $GF(n)$ (for prime n). To maintain the lowest k hash values of the elements observed in the stream, we use a binary tree of size k for each stream. On element arrival, we calculate its hash value and add it to the tree if the value is smaller than the maximal value currently in the tree, and is not already in the tree; for a large enough stream, the expected time of these operations is constant.

At query time, we iterate over the $\log \frac{1}{\tau}$ pairs of trees. Each of the trees has k minimal values of the relevant stream. For each pair, we take the set of the k lowest hash values among the $2k$ values, $h_{1\ldots k}(A \cup B)$, and intersect it with the two sets of k values, $h_{1\ldots k}(A)$ and $h_{1\ldots k}(B)$. The result of the iteration is the size of the intersections divided by k. The query returns the median among these results.

The space consumption is composed of $O(k \log \frac{1}{\tau})$ words for the trees and $O(\log \frac{1}{\tau})$ for maintaining the hash functions. The expected running time is $O(\log \frac{1}{\tau})$ per element and $O(k \log \frac{1}{\tau})$ per query. When comparing this result to [1], notice that Table 1.1 is for constant failure probability.

4.3. Windowed data stream algorithm

The similarity estimation procedure in this case is similar to the one given in [1]. The difference is that we run it $\log \frac{1}{\tau}$ times in parallel and use the *2-k-min-wise* estimator (given above). Notice that the value of k here is less than the value of k in [1] by a factor of $\log \frac{1}{\tau}$.

The expected running time per element, per function, is $O(1)$ in amortized analysis, since the time for hashing is constant. As we run the procedure $\log \frac{1}{\tau}$ times in parallel, we get a total of $O(\log \frac{1}{\tau})$. The space consumption is composed of $O(k \log N)$ words per functions, hence $O(k \log N \log \frac{1}{\tau})$ in total.

4.4. An improved rarity estimator

A similar technique can be applied for rarity estimation. Consider a stream A. #α-rare is defined as the number of elements that appears exactly α times, and #distinct is the number of distinct elements in the stream. The α-rarity of the stream is defined as $R_\alpha = \frac{\#\alpha - rare}{\#\text{distinct}}$.

Datar and Muthukrishnan proposed a rarity estimator [7] that uses *min-wise* functions. They maintain the minimal hash value and the frequency of its corresponding element for each of the functions. We denote the frequency of the minimal hash value for the j-th function with freq(j); the rarity is then estimated as follows:

$$\hat{R_\alpha}(A) = \frac{|\{j | 1 \leq j \leq k', freq(j) = \alpha\}|}{k'}, \ \hat{R_\alpha}(A) \in R_\alpha(A) \pm \epsilon$$

For an error bound ϵ, success probability at least $1 - \tau$ and k'-*min-wise* functions; s.t. $0 < \epsilon < 1$, $0 < \tau < 1$, and $k' \geq 2\epsilon^{-2} \log \tau^{-1}$.

Using our construction, one can utilize *2-k-min-wise* functions to perform fewer iterations and improve the overall complexity by choosing $\log \frac{1}{\tau}$ independent *2-k-min-wise* hash functions, such that $k \geq 2\epsilon^{-2}$. For each of the functions, we maintain the k minimal values and apply the same estimator ($\hat{R_\alpha}(A)$). The desired result is the median among these values. The precision is preserved because the k values of each function are pair-wise independent, and the $\log \frac{1}{\tau}$ functions are

independent. Hence, we can use both Chebyshev's inequality and the Chernoff bound, as described before for similarity, in Theorem 4.1.

5. Conclusion and future work

In this paper, we introduced a general framework that exponentially improves the time, space, and degree of independence required by *min-wise-based* algorithms. These exponential improvements are obtained by defining and constructing *d-k-min-wise* hash functions, in which, surprisingly for the practical case where $d = 2$, only 8-wise independence is required. Our approach gets around the $\Omega(\log \frac{1}{\epsilon})$ independence lower bound [21], as it relaxes the needs of approximately *min-wise* functions. Moreover, it does not change the algorithm itself, but only the way it samples and is thus simple to adapt.

There are few interesting directions for future work. First, to try to nail down the *8-wise* independence required for *2-k-min-wise* functions, or, alternatively, to find a matching lower bound for the approximately *d-k-min-wise* functions. In addition, it would be interesting to try to close the space gap for non-numeric estimators, where one cannot utilize our framework as the probabilistic techniques used are not applicable.

Appendix A

Lemma 3.4. *For $i > 0$, constant d, $\epsilon \in (0,1)$, $k > d - 1 + 2 \cdot 8^{\frac{2}{l}} \frac{(6l)^{1+\frac{1}{l}}}{\epsilon^2}$ and $l = 2d + 2$*

$$\Pr_l[RANK_t(h(X)) \in b_{-i}] \leq \Pr_l\left[\cup_{j=-\infty}^{-i} RANK_t(h(X)) \in b_j\right] \leq \frac{1}{i^{d+1}}$$

Proof. For block b_{-i}, $X = \{x_1, \ldots, x_m\}$, we define Z_j to be indicator variable such that

$$Z_j = \begin{cases} 1 & h(x_j) < (1-\epsilon i)\frac{t|U|}{m} \\ 0 & otherwise \end{cases}$$

In addition, we define $Z = \sum_j Z_j$, and E_{-i} to be the expected value of Z. Notice that since Z is sum of indicator variables, $E_{-i} = (1-\epsilon i)\frac{t}{m}m = (1-\epsilon i)t$.

We use the above definitions to show that

$$\begin{aligned}
\Pr_l[RANK_t(h(X)) \in b_{-i}] &\leq \Pr_l\left[\cup_{j=-\infty}^{-i} RANK_t(h(X)) \in b_j\right] \\
&\leq \Pr_l[\text{number of hash values smaller than the upper boundary of block } b_{-i} \geq t] \\
&\leq \Pr_l[Z \geq t] = \Pr_l[Z - E_{-i} \geq t - E_{-i}] \\
&\leq \Pr_l[|Z - E_{-i}| \geq t\epsilon i)]
\end{aligned}$$

Using Markov's inequality, as l is even:

$$\Pr_l[|Z - E_{-i}| \geq t\epsilon i] \leq \frac{E(|Z - E_{-i}|^l)}{[t\epsilon i]^l}$$

We use the following from Lemma A.1:

$$E(|Z - E_i|^l) \leq 8(6l)^{\frac{l+1}{2}}(E_i)^{\frac{l}{2}}$$

Thus,

$$\Pr_l[RANK_t(h(X)) \in b_i] \leq \frac{8(6l)^{\frac{l+1}{2}}(t(1-\epsilon i))^{\frac{l}{2}}}{[t\epsilon i]^l} = \frac{8(6l)^{\frac{l+1}{2}}(1-\epsilon i)^{\frac{l}{2}}}{t^{\frac{l}{2}}[\epsilon i]^l}$$

Note that $t > 2 \cdot 8^{\frac{2}{l}} \frac{(6l)^{1+\frac{1}{l}}}{\epsilon^2}$ for $t = k - d + 1$, and proceed as follows:

$$\Pr_l[RANK_t(h(X)) \in b_i] \leq \frac{8(6l)^{\frac{l+1}{2}}(1-\epsilon i)^{\frac{l}{2}}}{[2 \cdot 8^{\frac{2}{l}} \frac{(6l)^{1+\frac{1}{l}}}{\epsilon^2}]^{\frac{l}{2}}[\epsilon i]^l} = \frac{8(6l)^{\frac{l+1}{2}}(1-\epsilon i)^{\frac{l}{2}}}{2^{\frac{l}{2}} \cdot 8 \cdot \frac{(6l)^{\frac{l+1}{2}}}{\epsilon^l}[\epsilon i]^l} = \frac{(1-\epsilon i)^{\frac{l}{2}}}{2^{\frac{l}{2}} i^l} \leq \frac{1}{2^{\frac{l}{2}} i^l} < \frac{1}{i^l}$$

As $l = 2d + 2$ is defined, we get:

$$\Pr_l[RANK_t(h(X)) \in b_i] \leq \frac{1}{i^{d+1}} \qquad \square$$

Lemma A.1. *Let Z_j be a set of indicator variables, let $Z = \sum_j Z_j$, let E_i be the expected value of Z, and let $l = 6$.*

$$E(|Z - E_i|^l) \leq 8(6l)^{\frac{l+1}{2}}(E_i)^{\frac{l}{2}}$$

Proof. The proof is based on Indyk's lemma 2.2 [19] for the case where l is constant, in particular for the case where $l = 6$. Here, we show how one can reduce an exponent factor. In the original proof, the probability $\Pr|Z - E_i| \geq \epsilon E_i$ is estimated by an upper bound, hence one can remove the previous redundant addition in each addend of the sum:

$$E(|Z - E_i|^l) \leq 2\sum_{j=1}^{\infty}((j^l - (j-1)^l) \cdot 2e^{-\frac{j^2}{2E_i^2}E_i})$$

as $l = 6$, the equality is bounded by

$$E(|Z - E_i|^l) \leq 2\sum_{j=1}^{\infty}(6j^{l-1} \cdot 2e^{-\frac{j^2}{2E_i^2}E_i})$$

by continuing the proof as in [19], we get $E(|Z - E_i|^l) \leq 8(6l)^{\frac{l+1}{2}}(E_i)^{\frac{l}{2}}$. □

Lemma A.2. *Let $\epsilon \in (0,1)$, $n > k > d + \frac{d}{\epsilon^2}$, $d < \frac{1}{\epsilon}$ and $\alpha = (\frac{n}{n-d} \cdot \frac{k-d+1}{k})(\frac{n-1}{n-d} \cdot \frac{k-d+1}{k-1}) \cdots (\frac{n-d+1}{n-d} \cdot \frac{k-d+1}{k-d+1})$:*

$$1 - \epsilon d < \alpha < 1 + \epsilon d$$

Proof. For $\alpha < 1 + \epsilon d$:

$$\begin{aligned}
\alpha &= (\frac{n}{n-d} \cdot \frac{k-d+1}{k})(\frac{n-1}{n-d} \cdot \frac{k-d+1}{k-1}) \cdots (\frac{n-d+1}{n-d} \cdot \frac{k-d+1}{k-d+1}) \\
&< \frac{n}{n-d} \cdot \frac{n-1}{n-d} \cdots \frac{n-d+1}{n-d} \\
&< (\frac{n}{n-d})^d \\
&= (1 + \frac{d}{n-d})^d \\
&< (1 + \frac{d}{\frac{d}{\epsilon^2}})^d \\
&= (1+\epsilon^2)^d \\
&= \binom{d}{0}1 + \binom{d}{1}\epsilon^2 + \binom{d}{2}\epsilon^4 + \cdots + \binom{d}{d}\epsilon^{2d} \\
&\leq 1 + \epsilon d
\end{aligned}$$

For $\alpha > 1 - \epsilon d$:

$$\begin{aligned}
\alpha &= (\frac{n}{n-d} \cdot \frac{k-d+1}{k})(\frac{n-1}{n-d} \cdot \frac{k-d+1}{k-1}) \cdots (\frac{n-d+1}{n-d} \cdot \frac{k-d+1}{k-d+1}) \\
&> \frac{k-d+1}{k} \cdot \frac{k-d+1}{k-1} \cdots \frac{k-d+1}{k-d+1} \\
&> (\frac{k-d+1}{k})^d \\
&> (1 - \frac{d}{k})^d \\
&> (1 - \frac{d}{\frac{d}{\epsilon^2}})^d \\
&= (1-\epsilon^2)^d \\
&> \binom{d}{0}1 - \binom{d}{1}\epsilon^2 - \binom{d}{2}\epsilon^4 - \cdots - \binom{d}{d}\epsilon^{2d} \\
&\geq 1 - \epsilon d \qquad \square
\end{aligned}$$

References

[1] G. Feigenblat, E. Porat, A. Shiftan, Exponential time improvement for min-wise based algorithms, in: Proceedings of the Twenty-second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA '11, San Francisco, California, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2011, pp. 57–66, http://dl.acm.org/citation.cfm?id=2133036.2133041.

[2] M.N. Wegman, J.L. Carter, New hash functions and their use in authentication and set equality, J. Comput. Syst. Sci. 22 (3) (1981) 265–279.

[3] K. Mulmuley, Randomized geometric algorithms and pseudo-random generators, in: SFCS '92: Proceedings of the 33rd Annual Symposium on Foundations of Computer Science, IEEE Computer Society, Washington, DC, USA, 1992, pp. 90–100.

[4] A.Z. Broder, M. Charikar, A.M. Frieze, M. Mitzenmacher, Min-wise independent permutations (extended abstract), in: STOC '98: Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing, ACM, New York, NY, USA, 1998, pp. 327–336.

[5] A.Z. Broder, On the resemblance and containment of documents, in: Compression and Complexity of Sequences, SEQUENCES97, IEEE Computer Society, 1997, pp. 21–29.

[6] A.Z. Broder, S.C. Glassman, M.S. Manasse, G. Zweig, Syntactic clustering of the web, in: Selected Papers from the Sixth International Conference on World Wide Web, Elsevier Science Publishers Ltd., Essex, UK, 1997, pp. 1157–1166.

[7] M. Datar, S. Muthukrishnan, Estimating rarity and similarity over data stream windows, in: Proceedings of 10th Annual European Symposium on Algorithms, in: Lect. Notes Comput. Sci., vol. 2461, 2002, pp. 323–334.

[8] E. Cohen, Size-estimation framework with applications to transitive closure and reachability, J. Comput. Syst. Sci. 55 (3) (1997) 441–453.

[9] A.Z. Broder, Identifying and filtering near-duplicate documents, in: COM '00: Proceedings of the 11th Annual Symposium on Combinatorial Pattern Matching, Springer-Verlag, London, UK, 2000, pp. 1–10.

[10] G.S. Manku, A. Jain, A. Das Sarma, Detecting near-duplicates for web crawling, in: WWW '07: Proceedings of the 16th International Conference on World Wide Web, ACM, New York, NY, USA, 2007, pp. 141–150.

[11] H. Yang, J. Callan, Near-duplicate detection by instance-level constrained clustering, in: SIGIR '06: Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM, New York, NY, USA, 2006, pp. 421–428.

[12] M. Henzinger, Finding near-duplicate web pages: a large-scale evaluation of algorithms, in: SIGIR '06: Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM, New York, NY, USA, 2006, pp. 284–291.

[13] E. Cohen, H. Kaplan, Tighter estimation using bottom k sketches, Proc. VLDB Endow. 1 (1) (2008) 213–224.

[14] E. Cohen, H. Kaplan, Summarizing data using bottom-k sketches, in: PODC, 2007, pp. 225–234.

[15] T.H. Haveliwala, A. Gionis, D. Klein, P. Indyk, Evaluating strategies for similarity search on the web, in: WWW '02: Proceedings of the 11th International Conference on World Wide Web, ACM, New York, NY, USA, 2002, pp. 432–442.

[16] A.S. Das, M. Datar, A. Garg, S. Rajaram, Google news personalization: scalable online collaborative filtering, in: WWW '07: Proceedings of the 16th International Conference on World Wide Web, ACM, New York, NY, USA, 2007, pp. 271–280.

[17] Y. Bachrach, R. Herbrich, E. Porat, Sketching algorithms for approximating rank correlations in collaborative filtering systems, in: J. Karlgren, J. Tarhio, H. Hyyrö (Eds.), SPIRE, in: Lect. Notes Comput. Sci., vol. 5721, Springer, 2009, pp. 344–352.

[18] Y. Bachrach, E. Porat, J.S. Rosenschein, Sketching techniques for collaborative filtering, in: The Twenty-First International Joint Conference on Artificial Intelligence (IJCAI 2009), Pasadena, CA, 2009, pp. 2016–2021.

[19] P. Indyk, A small approximately min-wise independent family of hash functions, J. Algorithms (1999) 454–456.

[20] M. Saks, A. Srinivasan, S. Zhou, D. Zuckerman, Low discrepancy sets yield approximate min-wise independent permutation families, in: Proc. International Workshop on Randomization and Approximation Techniques in Computer Science, Springer, 1999, pp. 29–32.

[21] M. Pătraşcu, 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.

[22] Y. Bachrach, E. Porat, Sketching for big data recommender systems using fast pseudo-random fingerprints, in: Proceedings of the 40th International Conference on Automata, Languages, and Programming – Volume Part II, ICALP'13, Springer-Verlag, Berlin, Heidelberg, 2013, pp. 459–471.

[23] Y. Bachrach, E. Porat, Fingerprints for highly similar streams, Inf. Comput. 244 (C) (2015) 113–121.

[24] G. Feigenblat, E. Porat, A. Shiftan, Exponential space improvement for minwise based algorithms, in: D. D'Souza, T. Kavitha, J. Radhakrishnan (Eds.), IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2012), in: LIPIcs. Leibniz Int. Proc. Inform., vol. 18, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 2012, pp. 70–85.

[25] M. Thorup, Bottom-k and priority sampling, set similarity and subset sums with minimal independence, in: Proceedings of the Forty-Fifth Annual ACM Symposium on Theory of Computing, STOC '13, ACM, New York, NY, USA, 2013, pp. 371–380.

[26] G. Cormode, S. Muthukrishnan, What's new: finding significant differences in network data streams, IEEE/ACM Trans. Netw. 13 (6) (2005) 1219–1232.

[27] S. Ganguly, M. Garofalakis, R. Rastogi, Processing set expressions over continuous update streams, in: SIGMOD '03: Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data, ACM, New York, NY, USA, 2003, pp. 265–276.

[28] P.B. Gibbons, S. Tirthapura, Estimating simple functions on the union of data streams, in: SPAA '01: Proceedings of the Thirteenth Annual ACM Symposium on Parallel Algorithms and Architectures, ACM, New York, NY, USA, 2001, pp. 281–291.

[29] M. Pătraşcu, M. Thorup, The power of simple tabulation hashing, in: Proceedings of the Forty-Third Annual ACM Symposium on Theory of Computing, STOC '11, San Jose, California, USA, ACM, New York, NY, USA, ISBN 978-1-4503-0691-1, 2011, pp. 1–10.

[30] E. Cohen, M. Datar, Sh. Fujiwara, A. Gionis, P. Indyk, R. Motwani, D.J. Ullman, Ch. Yang, Finding interesting associations without support pruning, IEEE Trans. Knowl. Data Eng. 13 (1) (2001) 64–78.