Exponential Space Improvement for *min-wise* Based Algorithms

Guy Feigenblat[1,2], Ely Porat[1], and Ariel Shiftan[1]

1 **Department of Computer Science**
Bar-Ilan University, Ramat Gan 52900, Israel
`{feigeng, porately, shiftaa}@cs.biu.ac.il`

2 **IBM Haifa Research Lab, Haifa University Campus**
Mount Carmel, Haifa 31905, Israel

Abstract

In this paper we introduce a general framework that exponentially improves the space, the degree of independence, and the time needed by *min-wise* based algorithms. The authors, in SODA11, [15] introduced an exponential time improvement for *min-wise* based algorithms by defining and constructing an almost *k-min-wise* independent family of hash functions. 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 gets around the $\Omega(\log \frac{1}{\epsilon})$ independence lower bound in [23]. This is done 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. Moreover, as the degree of independence is a small constant, our function can be implemented efficiently.

Informally, under this definition, 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. This property measures the randomness of the family, as choosing a truly random function, obviously, satisfies the definition for $d = k = |X|$. We define and give an efficient time and space construction of approximately *d-k-min-wise* independent family of hash functions for the case where $d = 2$, as this is sufficient for the additional exponential improvement. 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 studied and no construction was given before. As an example 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.

1998 ACM Subject Classification F.1.2 Modes of Computation - Online computation

Keywords and phrases Streaming, Min-Wise, Hash Functions, Similarity, On line algorithms, Sub-linear algorithms

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2012.70

1 Introduction

Hash functions are fundamental building blocks of many algorithms. They map values from one domain to another, usually smaller. 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, one that would be 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 nowadays 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 *2-wise-independent* family of hash functions assures the independence of each pair of elements. It is known that only $2\log(u')$ bits are enough in order to choose and maintain such a function out of the family.

This work is focused on the area of min-hashing. One derivative of min-hashing is *min-wise* independent permutations, which were first introduced in [22, 6]. 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 for any $x \in X$, where h is chosen uniformly at random in $\mathcal{H}$, we have:

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

Min hashing is a widely used tool for solving problems in computer science such as estimating similarity [6, 4, 7], rarity [14], transitive closure [8], web page duplicate detection [5, 21, 26, 19], sketching techniques [11, 10], and other data mining problems [18, 13, 1, 3].

One of the key properties of min hashing is that it enables to sample the universe of the elements being hashed. This is because each element, over the random choice of hash function 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. It is 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, as described in [6, 4, 7, 8].

Indyk, in [20], was first to give a construction of a small approximately *min-wise* independent family of hash functions, another construction was proposed in [25]. 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 in [23] that $\Omega(\log \frac{1}{\epsilon})$ independence is needed for maintaining an approximately *min-wise* function, hence Indyk's construction is optimal.

Recently, in STOC11 [24], a new novel technique that bypasses the above lower bound was proposed by the same authors. They showed that simple tabulation yields approximately *min-wise* hash function, hence it requires constant time and space for each function.

In a previous paper [15] the authors defined and gave a construction for an **approximately k-min-wise *(*ϵ-k-min-wise *)* independent** family of 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 $Y \subset X$, $|Y| = k$ 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. It was also shown in [15] 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. Formerly, most *min-wise* based applications used k different approximately *min-wise* independent hash functions, i.e. they maintained k different samples. The authors, in [15], proposed to use only one approximately *k-min-wise* independent hash function in order to maintain the k samples, by using the k minimal hash values. As by the definition the k minimal elements are fully independent, the estimators' precision can be preserved. Furthermore, the use of this function was found to reduce exponentially the running time of previous known results for *min-wise* based algorithms. The authors offered a general framework, and gave examples of how to apply it for estimating similarity, rarity and entropy of random graphs. In this paper we take it a step forward and reduce exponentially the space and the degree of independence needed by *min-wise* base algorithms. Recently, Porat and Bachrach in [2] 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. In other words, one can store just a single bit rather than the full element IDs.

Both approximately *min-wise* and *k-min-wise* hash functions, use low degree of independence (hence potentially low memory and runtime), and therefore are 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, it is only allowed to perform one pass over the data. 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. Similarly 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 reduces exponentially the space, the degree of independence, and the time needed by *min-wise* based algorithms, in addition to the exponential time improvement achieved in SODA11[15]. We do this by defining and constructing a small approximately *d-k-min-wise* independent family of hash functions. As will be discussed here, many *min-wise* based estimators can be adjusted to use our construction, and reduce exponentially the space and the degree of independence consumed.

First, we extend the notion of *min-wise* independent family of hash functions by defining a *d-k-min-wise* independent family of hash functions. Then, we show a construction of an approximately such family for the practical case where $d = 2$. The construction for general d is a technical generalization of that case, and since $d = 2$ is enough for the improvements, we omit the generalization to the full version. Finally, as a usage example, we show how to apply it to estimating similarity and rarity over data streams.

Under our definition for *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, for $d = 2$, is a 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 [23] 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 exponential space and degree of independence improvement of *min-wise* based algorithms, such as in [8, 14, 5, 18, 10, 11, 1, 3, 21, 12, 16, 13, 17, 26, 19]. Formerly, *min-wise* based algorithms used either k independent approximately *min-wise* hash functions (which can be implemented using either [20] or [24]), or one approximately *k-min-wise* independent function, as was proposed in [15]. This was done in order to sample k independent elements from the universe, notice that even if we use tabulation [24] 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 improved exponentially 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 much less degree of independence, specifically only constant degree, and amplify the precision using probabilistic techniques. In comparison to the technique used by Porat and Bachrach in [2], we use more space (as we maintain the elements' IDs), but our running time is better in more than $O(\log \frac{1}{\epsilon})$ factor.

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, pair-wise is sufficient for applying Chebyshev, and this is why $d = 2$ should be used. By taking the median out of the independent samples, using 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. We found this to improve exponentially the space and the degree of independence (as it is constant for each function), while maintaining the exponential time improvement in [15].

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, hence they can be aggregated (as described above). The estimators' values in most of the applications we considered were indeed numerics, 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 comparison of results.

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 constant failure probability.

	previous [14, 9] result	previous result [15]	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	$\frac{1}{\epsilon^2}$	$\frac{1}{\epsilon^2}$	$\frac{1}{\epsilon^2}$

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

2 Definitions and Notations

d-k-min-wise independent family of hash functions, are generalization of *min-wise* and *k-min-wise* independent family of hash functions. Informally, under definition 2.1 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. In other words, 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 $d = k = |X|$.

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 2.1.** A family of functions $\mathcal{H} \subseteq [u] \to [u]$ (where $[u] = \{0 \ldots u-1\}$) 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 2.2.** A family of functions $\mathcal{H} \subseteq [u] \to [u]$ (where $[u] = \{0 \ldots u-1\}$) 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

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

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

We will also use the following notation. 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 $m = n - d = |X|$, where k, d and n are drawn from the definitions above.

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 it. In the subsequent sections we delve into the full technical details.

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, and in contrast we show that the probability of a high 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

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

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 s.t. b_0 boundaries 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 $\Pr\left[h(y_1), h(y_2), \ldots, h(y_d) < RANK_{k-d+1}(h(X))\right]$ in the l-wise case is

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

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 $\Pr_l\left[RANK_{k-d+1}(h(X)) \in b_i\right]$ decrease polynomially (roughly $\frac{1}{|i|^{d+1}}$), for each block b_i. Then, by adding to l another d degrees of independence, we can assume the elements of Y are randomly distributed, which is utilized to bound $\Pr_l\left[h(y_1),\ldots,h(y_d) \leq RANK_{k-d+1}(h(X))\right]$. Eventually, we use both in order to bound the difference, and to 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.**

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

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:

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

◀

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

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

$$\frac{k}{n}\frac{k-1}{n-1}\cdots\frac{k-d+1}{n-d+1}\sum_{i\in\phi}\Pr_l\left[RANK_{k-d+1}(h(X)) \in b_i\right]$$

▶ **Lemma 3.2.** *Let d, k, ϵ, n, and $h:[u]\to[u]$ be as in definition 2.2, and denote* $\Delta =$

$$\sum_{i\in\phi}\Pr_l\left[RANK_{k-d+1}(h(X)) \in b_i\right]\times$$

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

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

$$\Pr_l\left[h(y_1), h(y_2), \ldots, h(y_d) < RANK_{k-d+1}(h(X))\right] =$$

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

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

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

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

◀

3.3 Blocks partitioning

We divide the universe $[0, |U|]$ into non-overlapping blocks. We construct the blocks around $\frac{t|U|}{m}$ as follows: for $i \in \mathbb{Z}$, $b_i = \left[(1+\epsilon(i-1))\frac{t|U|}{m}, (1+\epsilon i)\frac{t|U|}{m}\right)$, e.g.
$\ldots, b_{-1} = \left[(1-2\epsilon)\frac{t|U|}{m}, (1-\epsilon)\frac{t|U|}{m}\right)$, $b_0 = \left[(1-\epsilon)\frac{t|U|}{m}, \frac{t|U|}{m}\right)$, $b_1 = \left[\frac{t|U|}{m}, (1+\epsilon)\frac{t|U|}{m}\right), \ldots$

Notice that, by the blocks partitioning and according to definition 2.2, 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 'negative blocks' otherwise ($i \le 0$). For the rest of the paper, we ignore blocks which are outside $[0, |U|]$.

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

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 is decreasing polynomially with the growth of $|i|$. For convenience, we separate the bound for the negative and positive blocks, and specifically we use 1 as an upper bound for the probabilities of b_0, b_1. In addition, we bound $\Pr_l\left[RANK_t(h(X)) \in b_i\right]$, 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 $i > 1$, $d = 2$, $\epsilon \in (0,1)$, $k > d + 2 \cdot 8^{\frac{2}{l}} \frac{(6l)^{1+\frac{1}{l}}}{\epsilon^2} - 1$ and $l = 2d + 2$:*

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

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

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

In addition we define $Z = \sum_j Z_j$, and E_i to be the expected value of Z. Notice that since Z is 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

$$\Pr_l\left[RANK_t(h(X)) \in b_i\right] \leq$$

$$\Pr_l[\text{number of hash values smaller than the lower boundary of block } b_i < t] =$$

$$\Pr_l[Z < t] \leq \Pr_l[E_i - Z \geq E_i - t] \leq \Pr_l[|E_i - Z| \geq E_i - t] =$$

$$\Pr_l[|Z - E_i| \geq t(1+\epsilon(i-1)) - t]$$

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,

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

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:

$$< \frac{8(6l)^{\frac{l+1}{2}}(1+\epsilon(i-1))^{\frac{l}{2}}}{[2 \cdot 8^{\frac{2}{l}} \frac{(6l)^{1+\frac{1}{l}}}{\epsilon^2}]^{\frac{l}{2}}[\epsilon(i-1)]^l} = \frac{8(6l)^{\frac{l+1}{2}}(1+\epsilon(i-1))^{\frac{l}{2}}}{2^{\frac{l}{2}} \cdot 8 \cdot \frac{(6l)^{\frac{l+1}{2}}}{\epsilon^l}[\epsilon(i-1)]^l} = \frac{(1+\epsilon(i-1))^{\frac{l}{2}}}{2^{\frac{l}{2}}(i-1)^l}$$

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

As $l = 2d + 2$ is defined,

$$\leq \left(\frac{1}{i-1}\right)^{d+1}$$

◀

The proof for the matching lemma for negative blocks is similar and thus, due to lack of space, is omitted to the full version of the paper.

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

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

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 $d = 2$, $\epsilon \in (0, 1)$, $k = \Omega(d + \frac{(l)^{1+\frac{1}{l}}}{\epsilon^2})$, $l \geq 2d + 2$, and using $(l + d)$ independence:*

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

Proof.

$$\sum_{i=-\infty}^{\infty} \Pr_{l+d}\left[RANK_t(h(X)) \in b_i\right] \times$$

$$\left[\Pr_{l+d}\left[h(y_1), h(y_2), \ldots, h(y_d) \leq RANK_t(h(X)) \mid RANK_t(h(X)) \in b_i\right] - \frac{\binom{k}{d}}{\binom{n}{d}}\right]$$

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:

$$\leq \sum_{i=-\infty}^{\infty} \Pr_{l}\left[RANK_t(h(X)) \in b_i\right]\left[(\frac{t}{m})^d(1 + \epsilon i)^d - \frac{\binom{k}{d}}{\binom{n}{d}}\right]$$

$$\leq \left[\sum_{i=-\infty}^{0}\left(\Pr_{l}\left[\cup_{j=-\infty}^{i} RANK_t(h(X)) \in b_j\right] - \Pr_{l}\left[\cup_{j=-\infty}^{i-1} RANK_t(h(X)) \in b_j\right]\right)\right.$$
$$\left. + \sum_{i=1}^{\infty}\left(\Pr_{l}\left[\cup_{j=i}^{\infty} RANK_t(h(X)) \in b_j\right] - \Pr_{l}\left[\cup_{j=i+1}^{\infty} RANK_t(h(X)) \in b_j\right]\right)\right] \times$$

$$\left[(\frac{t}{m})^d(1 + \epsilon i)^d - \frac{\binom{k}{d}}{\binom{n}{d}}\right]$$

By changing the order we get a telescoping sum as follows:

$$= \sum_{i=-\infty}^{-1} \Pr_{l}\left[\cup_{j=-\infty}^{i} RANK_t(h(X)) \in b_j\right]\left[(\frac{t}{m})^d(1 + \epsilon i)^d - (\frac{t}{m})^d(1 + \epsilon(i + 1))^d\right] +$$

$$\Pr_{l}\left[\cup_{j=-\infty}^{0} RANK_t(h(X)) \in b_j\right]\left[(\frac{t}{m})^d - \frac{\binom{k}{d}}{\binom{n}{d}}\right] +$$

$$\Pr_{l}\left[\cup_{j=1}^{\infty} RANK_t(h(X)) \in b_j\right]\left[(\frac{t}{m})^d(1 + \epsilon)^d - \frac{\binom{k}{d}}{\binom{n}{d}}\right] +$$

$$\sum_{i=2}^{\infty} \Pr_{l}\left[\cup_{j=i}^{\infty} RANK_t(h(X)) \in b_j\right]\left[(\frac{t}{m})^d(1 + \epsilon i)^d - (\frac{t}{m})^d(1 + \epsilon(i - 1))^d\right] \leq$$

Applying lemmas 3.3 and 3.4, bounding the probabilities of blocks b_0 and b_1 with 1:

$$\sum_{i=-\infty}^{-1} \frac{1}{|i|^{d+1}} |(\frac{t}{m})^d(1+\epsilon i)^d - (\frac{t}{m})^d(1+\epsilon(i+1))^d| +$$

$$|(\frac{t}{m})^d - \frac{\binom{k}{d}}{\binom{n}{d}}| + |(\frac{t}{m})^d(1+\epsilon)^d - \frac{\binom{k}{d}}{\binom{n}{d}}| +$$

$$\sum_{i=2}^{\infty} \frac{1}{(i-1)^{d+1}} |(\frac{t}{m})^d(1+\epsilon i)^d - (\frac{t}{m})^d(1+\epsilon(i-1))^d| =$$

We now substitute d (recall that $d = 2$)

$$\sum_{i=1}^{\infty} \frac{1}{|i|^3} |(\frac{k-1}{n-2})^2(\epsilon^2(2i-1) - 2\epsilon))| +$$

$$|(\frac{k-1}{n-2})^2 - \frac{k}{n}\frac{k-1}{n-1}| + |(\frac{k-1}{n-2})^2(1+\epsilon)^2 - \frac{k}{n}\frac{k-1}{n-1}| +$$

$$\sum_{i=2}^{\infty} \frac{1}{(i-1)^3} |(\frac{k-1}{n-2})^2(\epsilon^2(2i-1) + 2\epsilon))| \leq$$

$$2\frac{k}{n}\frac{k-1}{n-1} \sum_{i=1}^{\infty} \frac{1}{|i|^3} |\epsilon^2(2i-1) - 2\epsilon| +$$

$$|(\frac{k-1}{n-2})^2 - \frac{k}{n}\frac{k-1}{n-1}| + |(\frac{k-1}{n-2})^2(1+\epsilon)^2 - \frac{k}{n}\frac{k-1}{n-1}| +$$

$$2\frac{k}{n}\frac{k-1}{n-1} \sum_{i=2}^{\infty} \frac{1}{(i-1)^3} |\epsilon^2(2i-1) + 2\epsilon| \leq$$

$$8\frac{k}{n}\frac{k-1}{n-1}\epsilon$$

◀

The lemma for the lower bound, i.e. $-\Delta \leq \epsilon\frac{\binom{k}{d}}{\binom{n}{d}}$, is similar and due the lack of space is omitted to the full version of the paper.

▶ **Lemma 3.6.** *For $d = 2$, $\epsilon \in (0,1)$, $k = \Omega(d + \frac{(l)^{1+\frac{1}{l}}}{\epsilon^2})$, $l \geq 2d+2$, and using $(l+d)$ independence:* $-\Delta \leq \epsilon\frac{\binom{k}{d}}{\binom{n}{d}}$.

We conclude with the following theorem:

▶ **Theorem 3.7.** *For $d = 2$, $\epsilon \in (0,1)$, $k = \Omega(d + \frac{(l)^{1+\frac{1}{l}}}{\epsilon^2})$, $l \geq 2d+2$, any 8-wise independent family of hash functions is approximately 2-k-min-wise (ϵ-2-k-min-wise).*

Proof. Applying lemma 3.5 and lemma 3.6 to lemma 3.2 concludes the proof. ◀

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* functions for improving many *min-wise* based algorithms, such as in [8, 14, 5, 18, 10, 11, 1, 3, 21, 12, 16, 13, 17, 26, 19]. As was mentioned before, *min-wise* enables us to sample elements from the universe. In other words, 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 requires super constant degree of independence, which impacts both time and space [20, 23]. Recently, in SODA11 [15], 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 was found to improve exponentially 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 much less degree of independence, specifically only constant degree per function, and by probabilistic techniques amplifies the precision.

We propose a procedure that does not change the algorithm itself, but only the way it samples, hence it is simple to adapt. We found that *2-k-min-wise* independent functions are sufficient in order to preserve the precision. In details, 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 elements, and then average the values using Chebyshev's inequality. Notice that 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 Chernoff bound the precision becomes as desired. This procedure improves exponentially 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 utilizes 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 was mentioned in the introduction, the problem was studied by [14] and recently in [15]. The use of our construction improves exponentially the space and the degree of independence of current 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 [9] (which is also formally defined in [15]) we construct *2-k-min-wise* estimator. Let $h_1(A), h_2(A), ..., h_k(A)$ and $h_1(B), h_2(B), ..., h_k(B)$ be k pair-wise independent min hash values for the sets A and B, respectively, and $h_{1...k}(A)$ be the set containing $h_1(A), h_2(A), ..., h_k(A)$. 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

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

▶ **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 s.t. $z_j = 1$ if the j-th iteration of $\hat{S}(A,B)$ failed, and 0 otherwise. Respectively $Z = \sum_{j=1}^{\log\frac{1}{\tau}} z_i$. Since the $\log\tau^{-1}$ values of $\hat{S}(A,B)$ are independent, we can apply Chernoff bound to increase the success probability. The following is the overall failure probability of the estimator:

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

◀

4.2 Unbounded Data Stream Algorithm

We now present a similarity estimation algorithm in the unbounded data stream model. Recall that 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). In order 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. Notice that for 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 [15], 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 [15]. 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 [15] by a factor of $\log \frac{1}{\tau}$.

The expected running time per element, per function, is $O(1)$ in amortized, 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

Similar technique can be applied for rarity estimation. Consider a stream A. $\#\alpha$-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 in [14], using *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:

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

For $0 < \epsilon < 1$,$0 < \tau < 1$, and $k' \geq 2\epsilon^{-2} \log \tau^{-1}$ for an error bound ϵ, success probability at least $1 - \tau$ and k'-*min-wise* functions.

Using our construction, one can utilize *2-k-min-wise* functions to perform less iterations and improve the overall complexity. This is done by choosing $\log \frac{1}{\tau}$ independent *2-k-min-wise* hash functions, s.t. $k \geq 2\epsilon^{-2}$. 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 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 the degree of independence required by *min-wise* based algorithms. This exponential improvements are obtained by defining and constructing *d-k-min-wise* wise hash functions, in which surprisingly for the practical case, where $d = 2$, only 8-wise independent is required. Our approach gets around the $\Omega(\log \frac{1}{\epsilon})$ independence lower bound in [23], as it relaxes the need of approximately *min-wise* functions. Moreover, it does not change the algorithm itself, but only the way it samples, hence it is simple to adapt.

There are few interesting directions for future work. First, to try to nail down the 8 independence required, 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 can not utilize our framework as the probabilistic techniques used are not applicable.

References

1 Yoram Bachrach, Ralf Herbrich, and Ely Porat. Sketching algorithms for approximating rank correlations in collaborative filtering systems. In Jussi Karlgren, Jorma Tarhio, and Heikki Hyyrö, editors, *SPIRE*, volume 5721 of *Lecture Notes in Computer Science*, pages 344–352. Springer, 2009.

2 Yoram Bachrach and Ely Porat. Fast pseudo-random fingerprints. volume abs/1009.5791, 2010.

3 Yoram Bachrach, Ely Porat, and Jeffrey S. Rosenschein. Sketching techniques for collaborative filtering. In *The Twenty-First International Joint Conference on Artificial Intelligence (IJCAI 2009)*, pages 2016–2021, Pasadena, California, July 2009.

4 Andrei Z. Broder. On the resemblance and containment of documents. In *In Compression and Complexity of Sequences (SEQUENCES97*, pages 21–29. IEEE Computer Society, 1997.

5 Andrei Z. Broder. Identifying and filtering near-duplicate documents. In *COM '00: Proceedings of the 11th Annual Symposium on Combinatorial Pattern Matching*, pages 1–10, London, UK, 2000. Springer-Verlag.

6 Andrei Z. Broder, Moses Charikar, Alan M. Frieze, and Michael Mitzenmacher. Min-wise independent permutations (extended abstract). In *STOC '98: Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 327–336, New York, NY, USA, 1998. ACM.

7 Andrei Z. Broder, Steven C. Glassman, Mark S. Manasse, and Geoffrey Zweig. Syntactic clustering of the web. In *Selected papers from the sixth international conference on World Wide Web*, pages 1157–1166, Essex, UK, 1997. Elsevier Science Publishers Ltd.

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

9 Edith Cohen, Mayur Datar, Shinji Fujiwara, Aristides Gionis, Piotr Indyk, Rajeev Motwani, Jeffrey D. Ullman, and Cheng Yang. Finding interesting associations without support pruning, 1999.

10 Edith Cohen and Haim Kaplan. Summarizing data using bottom-k sketches. In *PODC*, pages 225–234, 2007.

11 Edith Cohen and Haim Kaplan. Tighter estimation using bottom k sketches. *PVLDB*, 1(1):213–224, 2008.

12 Graham Cormode and S. Muthukrishnan. What's new: finding significant differences in network data streams. *IEEE/ACM Trans. Netw.*, 13(6):1219–1232, 2005.

13 Abhinandan S. Das, Mayur Datar, Ashutosh Garg, and Shyam Rajaram. Google news personalization: scalable online collaborative filtering. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 271–280, New York, NY, USA, 2007. ACM.

14 Mayur Datar and S Muthukrishnan. Estimating rarity and similarity over data stream windows. In *In Proceedings of 10th Annual European Symposium on Algorithms, volume 2461 of Lecture Notes in Computer Science*, pages 323–334, 2002.

15 Guy Feigenblat, Ely Porat, and Ariel Shiftan. Exponential time improvement for min-wise based algorithms. In *To appear SODA '11: Proceedings of the 22nd annual ACM-SIAM symposium on Discrete algorithms.* Society for Industrial and Applied Mathematics, 2011.

16 Sumit Ganguly, Minos Garofalakis, and Rajeev Rastogi. Processing set expressions over continuous update streams. In *SIGMOD '03: Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 265–276, New York, NY, USA, 2003. ACM.

17 Phillip B. Gibbons and Srikanta 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*, pages 281–291, New York, NY, USA, 2001. ACM.

18 Taher H. Haveliwala, Aristides Gionis, Dan Klein, and Piotr Indyk. Evaluating strategies for similarity search on the web. In *WWW '02: Proceedings of the 11th international conference on World Wide Web*, pages 432–442, New York, NY, USA, 2002. ACM.

19 Monika 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*, pages 284–291, New York, NY, USA, 2006. ACM.

20 Piotr Indyk. A small approximately min-wise independent family of hash functions. In *Journal of Algorithms*, pages 454–456, 1999.

21 Gurmeet Singh Manku, Arvind Jain, and Anish Das Sarma. Detecting near-duplicates for web crawling. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 141–150, New York, NY, USA, 2007. ACM.

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

23 Mihai Pătraşcu and Mikkel Thorup. On the k-independence required by linear probing and minwise independence. In *Proc. 37th International Colloquium on Automata, Languages and Programming (ICALP)*, pages 715–726, 2010.

24 Mihai Pătraşcu and Mikkel Thorup. The power of simple tabulation hashing. In *Proc. 43rd ACM Symposium on Theory of Computing (STOC)*, 2011. To appear. See also arXiv:1011.5200.

25 Michael Saks, Aravind Srinivasan, Shiyu Zhou, and David Zuckerman. Low discrepancy sets yield approximate min-wise independent permutation families. In *In Proc. International Workshop on Randomization and Approximation Techniques in Computer Science*, pages 29–32. Springer, 1999.

26 Hui Yang and Jamie 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*, pages 421–428, New York, NY, USA, 2006. ACM.

A Appendix

▶ **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 in [20] 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. Notice that 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

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

by continuing the proof as in [20] we get, $E(|Z - E_i|^l) \leq 8(6l)^{\frac{l+1}{2}}(E_i)^{\frac{l}{2}}$

◀