IMPROVING MIN HASH VIA THE CONTAINMENT INDEX WITH APPLICATIONS TO METAGENOMIC ANALYSIS

DAVID KOSLICKI[1*], HOOMAN ZABETI[1]

[1] *Mathematics Department, Oregon State University, Corvallis, OR.*

Abstract. Min hash is a probabilistic method for estimating the similarity of two sets in terms of their Jaccard index, defined as the ration of the size of their intersection to their union. We demonstrate that this method performs best when the sets under consideration are of similar size and the performance degrades considerably when the sets are of very different size. We introduce a new and efficient approach, called the *containment min hash* approach, that is more suitable for estimating the Jaccard index of sets of very different size. We accomplish this by leveraging another probabilistic method (in particular, Bloom filters) for fast membership queries. We derive bounds on the probability of estimate errors for the containment min hash approach and show it significantly improves upon the classical min hash approach. We also show significant improvements in terms of time and space complexity. As an application, we use this method to detect the presence/absence of organisms in a metagenomic data set, showing that it can detect the presence of very small, low abundance microorganisms.

Keywords: *Min hash, Jaccard index, metagenomics, taxonomic classification, containment.*

1. Introduction

Min hash [3] is a fast, probabilistic method of estimating the Jaccard index, allowing for quick estimation of set similarity. Since the introduction of this technique by Broder (1997), this method has been used in a broad range of applications: from clustering in search engines [9], to association rule learning in data mining [8], to recent applications in computational biology [5, 18]. As a probabilistic method, min hash uses random sampling to estimate the Jaccard index of two sets. Bounds can be obtained on the probability of deviation from the true Jaccard index value in terms of the number of random samples used along with the magnitude of the true Jaccard value. Using the Chernoff bounds (Theorem 2.1), this probability of deviating from the true value grows exponentially as the size of the true Jaccard value decreases to zero. Hence, min hash returns tight estimates only when the true Jaccard index is large. This requirement for a large Jaccard index limits this technique to situations where the sets under consideration are of similar relative size and possess significant overlap with each other. In this manuscript, we introduce a modification of the min hash technique, which we call *containment min hash*, that does not possess this limitation and hence is appropriate for use when the sets under consideration have very different size.

After introducing the containment min hash technique, we derive rigorous probabilistic error bounds and compare these to those of the traditional min hash technique. This allows us to precisely state when the containment approach is superior to the classical approach. To demonstrate the practical utility of the containment min hash technique, we consider an application in the area of metagenomics (the study of all sampled DNA from a community of microorganisms), where the goal is to detect the presence or absence of a given genome in a metagenomic sample. This is an area

Date: October 5, 2017.
[*] Corresponding Author: `david.koslicki@math.oregonstate.edu`.

of study where the sets of interest differ in relative size by orders of magnitude. This application highlights the improvements of this approach: in both theory and practice, in many situations of interest, containment min hash is significantly more accurate, has smaller computational complexity, and utilizes less memory than the traditional min hash approach.

We begin by giving a high-level summary of the main idea behind the containment min hash technique which is summarized in Figure 1. Consider the case of estimating the Jaccard index between two sets A and B of very different size. Briefly, the traditional min hash randomly samples from the union $A \cup B$ and uses the number of sampled points that fall in $A \cap B$ to estimate the Jaccard index. With more sampled elements falling in $A \cap B$, the more accurate the Jaccard estimate will be. Part A) of Figure 1 demonstrates the case of sampling 100 random points from $A \cup B$ leading to 3 points lying in $A \cap B$. In the containment min hash approach, we randomly sample elements only from the smaller set (in this case, A) and use another probabilistic technique (in this manuscript, a bloom filter) to quickly test if this element is in B (and hence in $A \cap B$). This is used to estimate the containment index, which is then used to estimate the Jaccard index itself. Part B) of Figure 1 demonstrates this approach while sampling only 50 points from A and finds 22 points lying in $A \cap B$. This containment approach also experiences decreasing error as more points in $A \cap B$ are sampled, and so sampling from a smaller space (the set A instead of $A \cup B$) leads to significant performance improvements.

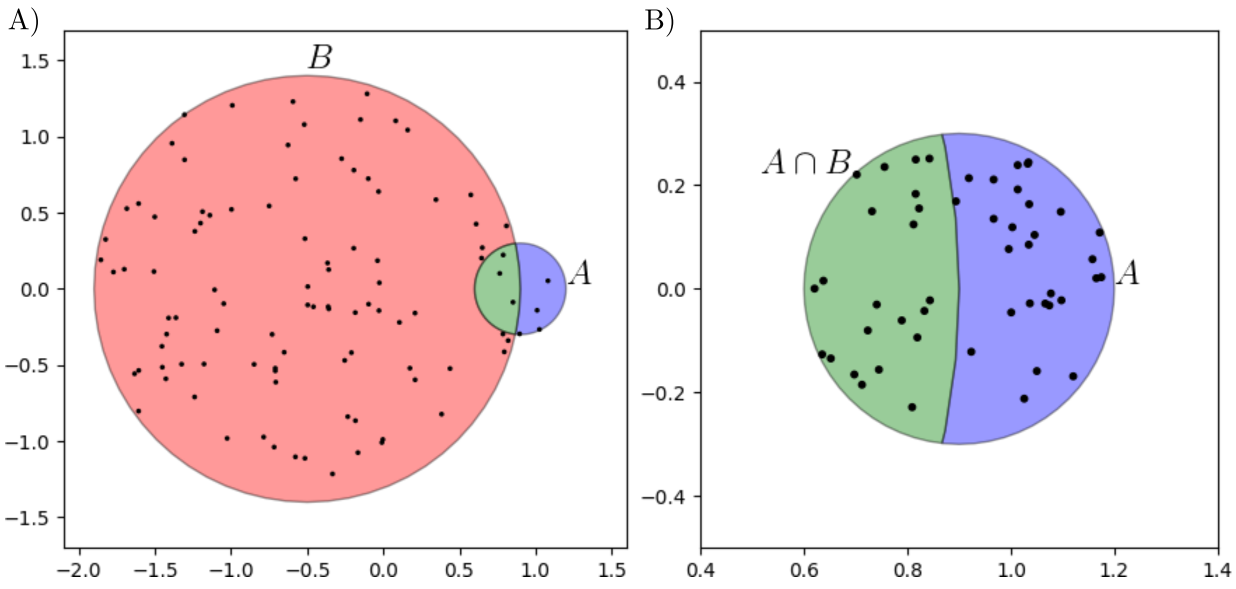


Figure 1. Conceptual comparison of classical min hash to the proposed containment approach when estimating the Jaccard index of very different sized sets. A) Sampling 100 points from $A \cup B$ (as is done in the classical min hash approach) leads to finding only 3 elements in $A \cap B$. B) Sampling just 50 points of A and testing if a point $x \in A \cap B$, finds 22 elements in $A \cap B$. This latter approach will be seen to lead to a better estimate of the Jaccard index.

2. Methods

Before describing min hash and containment min hash, we recall a few definitions and results of interest.

2.1. **Preliminaries.**

2.1.1. *Jaccard and Containment index.* The Jaccard index, also known as the Jaccard similarity coefficient, measures the similarity of two sets by comparing the relative size of the intersection to the union [14]. That is, for two non-empty finite sets A and B,

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

Hence, $0 \leq J(A,B) \leq 1$ with larger values indicating more overlap.

To compare the relative size of the intersection to the size of A, we similarly define the containment index of A in B (both non-empty) as:

$$C(A,B) = \frac{|A \cap B|}{|A|}.$$

So $0 \leq C(A,B) \leq 1$ and larger values indicate more of A lying in B.

2.1.2. *Chernoff Bounds.* We will use the classic Chernoff bounds in its multiplicative form to estimate the probability of relative error in the probabilistic methods we consider.

Theorem 2.1 ([17, Thm 4.4-4.5]). *Suppose $X_1, X_2, \ldots, X_n$ are independent, identically distributed Bernoulli random variables and let $X = \sum_{i=1}^n X_i$ and $\mu = \mathbb{E}(X)$, then the following statements hold for $0 < \delta < 1$:*

$$P(X \leq (1-\delta)\mu) \leq e^{-\frac{\delta^2 \mu}{2}} \tag{2.1}$$

$$P(X \geq (1+\delta)\mu) \leq e^{-\frac{\delta^2 \mu}{3}} \tag{2.2}$$

And hence,

$$P\left(\left|\frac{X-\mu}{\mu}\right| \geq \delta\right) \leq 2e^{-\frac{\delta^2 \mu}{3}}. \tag{2.3}$$

2.1.3. *Bloom Filters.* As mentioned in the Introduction, we will need a fast test of set membership to implement the containment min hash approach. While there exist many different such data structures (such as Bloom Filters [2], skip lists [20], cuckoo filters [10], quotient filters [1], etc.), we utilize the Bloom filter due to its ubiquity [6,7,12,16,18,19,21,22] in the application area considered here (computational biology) and the ease of its mathematical analysis.

We give a brief description of the construction of a bloom filter following the exposition of [17]. Given a set B with cardinality $n = |B|$, a bloom filter $\tilde{B} = (\tilde{B}_i)_{i=1}^m$ is a bit array of length m where m is a chosen natural number. Fix a set of hash functions $h_1, \ldots, h_k$ each with domain containing B and with range $\{1, \ldots, m\}$. Initially, each entry in the bloom filter is set to zero: $\tilde{B}_i = 0$ for each $i = 1, \ldots, m$. For each $x \in B$ and $j = 1, \ldots, k$, we set $\tilde{B}_{h_i(x)} = 1$. Given an element y in the domain of our hash functions, if

$$\prod_{j=1}^{k} \tilde{B}_{h_j(y)} = 0,$$

then by construction, we know that $y \notin B$. Because of this, we write $y \in \tilde{B}$ if $\prod_{j=1}^k \tilde{B}_{h_j(y)} = 1$, and $y \notin \tilde{B}$ otherwise. A straightforward calculation (with idealized hash functions) shows that the probability of $y \in \tilde{B}$ and yet $y \notin B$ is given by:

$$p = \left(1 - \left(1 - \frac{1}{m}\right)^{kn}\right)^k \approx \left(1 - e^{\frac{-kn}{m}}\right)^k. \tag{2.4}$$

The optimal false positive rate p is given by $p = 2^{-\frac{m}{n}\ln 2}$ when the number of hash functions is given by $k = \frac{m}{n}\ln 2$. Conversely, given a target false positive rate p, the optimal length m of the bloom filter is given by $m = -\frac{n \ln p}{\ln^2 2}$.

2.2. **Containment Min Hash.** Before detailing the containment min hash approach, we recall the classic min hash for comparison purposes.

2.2.1. *Classic Min Hash.* By the *classic min hash*, we mean the construction of Broder (1997). We detail this construction now. Given two non-empty sets A and B, we wish to estimate $J(A, B)$. Fix $k \in \mathbb{N}$ and select a family of (min-wise independent [4]) hash functions $\{h_1, \dots, h_k\}$ of $k \in \mathbb{N}$ each with domain containing $A \cup B$. For a set S in the domain of the hash functions, define $h_i^{\min}(S) = \arg\min_{s \in S} h_i(s)$ as an element of S that causes h_i to achieve its minimum value on S. Given appropriate hash functions (or an order on S), this minimum is unique. Define the random variables

$$X_i = \begin{cases} 1 & h_i^{\min}(A) = h_i^{\min}(B) \\ 0 & \text{otherwise.} \end{cases}$$

The probability of a collision (that is $h_i^{\min}(A) = h_i^{\min}(B)$ and hence $X_i = 1$) is equal to the Jaccard index of A and B [4] and hence the expectation is given by

$$E(X_i) = \frac{|A \cap B|}{|A \cup B|} = J(A, B).$$

Thus, for $X^k = \sum_{i=1}^{k} X_i$, the expectation is given by $E(X^k) = kJ(A, B)$ and so $J_X^{est} = \frac{X^k}{k}$ is used as the estimate of $J(A, B)$. Note that in practice, a single hash function h is commonly used and the elements hashing to the smallest k values are used in place of the h_i.

Applying the two-sided Chernoff bounds from equation (2.3) to the classic min hash approach, we have

$$P\left(\left| \frac{\frac{X^k}{k} - J(A,B)}{J(A,B)} \right| \geq \delta \right) \leq 2e^{-\delta^2 k J(A,B)/3}. \tag{2.5}$$

Thus, two quantities control the accuracy of this method for δ fixed: k and $J(A, B)$. Setting t as the threshold of probability of deviation from the Chernoff bounds (i.e. $t = 2e^{-\delta^2 k J(A,B)/3}$), we calculate the number of hash functions $k = k_X$ required to achieve this threshold:

$$k_X = \frac{-3 \ln(t/2) |A \cup B|}{\delta^2 |A \cap B|}. \tag{2.6}$$

2.2.2. *Containment Min Hash.* The containment min hash approach we propose differs from the classic min hash in that the family of k hash functions $\{h_1, \dots, h_k\}$ have domain containing A and we randomly sample from A instead of $A \cup B$. This results in estimating the containment index $C = C(A, B)$ instead of the Jaccard index $J = J(A, B)$, but we will late show how to recover an estimate of $J(A, B)$. The containment approach proceeds as follows: Let $\tilde{B}$ be a bloom filter with given false positive rate p and define the random variables

$$Y_i = \begin{cases} 1 & h_i^{\min}(A) \in \tilde{B} \\ 0 & \text{otherwise.} \end{cases}$$

These random variables essentially sample (uniformly) randomly from A and tests for membership in B via the bloom filter $\tilde{B}$. Following the same proof of Broder (1997), it is straightforward to show that $E(Y_i) = \dfrac{|A \cap B|}{|A|} + p$. Thus for $Y^k = \sum_{i=1}^{k} Y_i$, $E(Y^k) = k\dfrac{|A \cap B|}{|A|} + kp = kC + kp$. Hence, we use $C_{est} = \frac{Y^k}{k} - p$ as the estimate of $C(A, B)$.

Applying the two-sided Chernoff bounds from equation (2.3) now gives

$$P\left(\left|\frac{\left(\frac{Y^k}{k}-p\right)-C}{C}\right|\geq\delta\right)\leq 2e^{-\left(\frac{C}{C+p}\right)^2\delta^2k(C+p)/3} \tag{2.7}$$

It is important to note that $\frac{Y^k}{k}-p$ estimates $C(A,B)$ and not $J(A,B)$, as desired. To directly compare these quantities, we must derive the Jaccard index from the containment estimate and calculate the probability of deviation from the true value of the Jaccard index. To that end, for C_{est} the estimation of the containment, let $J_Y^{est} := \frac{|A|C_{est}}{|A|+|B|-|A|C_{est}}$. Note that in practice, a fast cardinality estimation technique (such as HyperLogLog [11]) can be used to approximate $|A|$ and $|B|$. The bound in the following proposition can be directly compared to that of equation (2.5).

Proposition 2.2. *For* $0<\delta<1$, *let* $\delta'=\frac{\delta|A\cup B|}{|A\cup B|+(1+\delta)|A\cap B|}$, *then*

$$P\left(\left|\frac{J_Y^{est}-J}{J}\right|\geq\delta\right)\leq 2e^{-(\frac{C}{C+p})^2\delta' k(C+p)/3}.$$

Proof. We first derive a useful characterization of the probability under consideration:

$$\begin{aligned}
P\left(\left|\frac{J_Y^{est}-J}{J}\right|\geq\delta\right) &= P\left(\frac{|A|C_{est}}{|A|+|B|-|A|C_{est}}\geq(1+\delta)J\right)+P\left(\frac{|A|C_{est}}{|A|+|B|-|A|C_{est}}\leq(1-\delta)J\right)\\
&= P\left(C_{est}\geq\frac{(1+\delta)\frac{|A|C}{|A\cup B|}(|A|+|B|)}{|A|\frac{|A\cup B|+(1+\delta)|A\cap B|}{|A\cup B|}}\right)+P\left(C_{est}\leq\frac{(1-\delta)\frac{|A|C}{|A\cup B|}(|A|+|B|)}{|A|\frac{|A\cup B|+(1-\delta)|A\cap B|}{|A\cup B|}}\right)\\
&= P\left(C_{est}\geq\frac{(1+\delta)(|A\cup B|+|A\cap B|)}{|A\cup B|+(1+\delta)|A\cap B|}C\right)+P\left(C_{est}\leq\frac{(1-\delta)(|A\cup B|+|A\cap B|)}{|A\cup B|+(1-\delta)|A\cap B|}C\right)\\
&= P\left(C_{est}\geq\left(1+\frac{\delta|A\cup B|}{|A\cup B|+(1+\delta)|A\cap B|}\right)C\right)\\
&\quad+P\left(C_{est}\leq\left(1-\frac{\delta|A\cup B|}{|A\cup B|+(1-\delta)|A\cap B|}\right)C\right).
\end{aligned} \tag{2.8}$$

Then for δ' as defined in the statement of the proposition and for $\delta''=\frac{\delta|A\cup B|}{|A\cup B|+(1-\delta)|A\cap B|}$, note that $0<\delta',\delta''<1$. Then using the two-sided Chernoff bounds from equations (2.1) and (2.2) applied to equation (2.8), we have that

$$\begin{aligned}
P\left(\left|\frac{J_Y^{est}-J}{J}\right|\geq\delta\right) &\leq e^{-(\frac{C}{C+p})^2\delta'^2k(C+p)/3}+e^{-(\frac{C}{C+p})^2\delta''^2k(C+p)/2}\\
&\leq 2e^{-\left(\frac{C}{C+p}\right)^2\delta'^2k(C+p)/3}.
\end{aligned}$$

□

Figure 2 gives a comparison of the bounds of deviation (as a function of δ for a fixed number of hash functions) for the classical min hash Jaccard estimate (equation (2.5)) and the containment min hash estimate of the Jaccard index (Proposition 2.2). This figure shows that the containment min hash approach has a significantly smaller probability of the estimate deviating from the true value.

Now, let $k=k_Y$ be the number of hash functions which is required to achieve a desire threshold upper bound of $t=2e^{-\left(\frac{C}{C+p}\right)^2\delta'^2k(C+p)/3}$. We then have that

$$k_Y=\frac{-3(C+p)\ln(t/2)\left(|A\cup B|+(1+\delta)|A\cap B|\right)^2}{C^2\delta^2|A\cup B|^2}. \tag{2.9}$$

3. Results

We begin by detailing the theoretical results obtained before turning to the application of interest.

3.1. **Theoretical Results.**

3.1.1. *Number of Hash Functions Required.* For both the classical min hash approach and the containment min hash approach, we use equations (2.6) and (2.9) to compare the number of hash functions k_X and k_Y required for a specified threshold of probability of deviation t. Calculating, we obtain:

$$\frac{k_Y}{k_X} = \frac{(C+p)}{C^2}\left(\frac{|A\cap B|}{|A\cup B|} + 2(1+\delta)\frac{|A\cap B|^2}{|A\cup B|^2} + (1+\delta)^2\frac{|A\cap B|^3}{|A\cup B|^3}\right). \tag{3.1}$$

Note that $\frac{|A\cap B|}{|A\cup B|} < 1$, so we have that:

$$\begin{aligned}
\frac{k_Y}{k_X} &\leq \frac{(C+p)}{C^2}\left(\frac{|A\cap B|}{|A\cup B|} + 2(1+\delta)\frac{|A\cap B|}{|A\cup B|} + (1+\delta)^2\frac{|A\cap B|}{|A\cup B|}\right) && (3.2)\\
&\leq \frac{(C+p)}{C}\frac{|A|}{|B|}\left(1 + 2(1+\delta) + (1+\delta)^2\right). && (3.3)\\
&\propto \frac{|A|}{|B|} && (3.4)
\end{aligned}$$

where the last proportionality holds when C is bounded away from zero. Hence, the containment approach uses a fraction (proportional to $\frac{|A|}{|B|}$) of the hashes that the classical approach uses to achieve the same bound of error. When $|B|$ is significantly larger than $|A|$, this reduction in the number of hash functions required can be significant. In Figure 3, we compare the relative error δ in estimating the Jaccard index to the number of hash functions required for the classical and proposed approach to have a probability of $\leq 1\%$ of more than δ relative deviation in their estimate. Interestingly, the number of hashes required by the containment approach is nearly constant as a function of $\frac{|B|}{|A|}$ whereas for the classical approach, the number of hashes required is increasing. Figure 4 depicts this observation.

3.1.2. *Time/Space Complexity.* Given sets A and B of size m and n respectively, both the classic min hash and containment min hash require $\mathcal{O}(m+n)$ time to form their respective data structures. When calculating their estimates of the Jaccard index $J(A, B)$, both approaches use computational time linear in the number of hash functions required (in the case of the containment min hash approach, this is due to Bloom filter queries being constant time [2]). Because of equation (3.1) and the discussion that followed, this implies that the ratio of time complexity of the classical min hash to the containment min hash is $\mathcal{O}\left(\frac{k_X}{k_Y}\right)$. Given equations (3.1) and (3.4), when B is very large in comparison to A, this implies that the containment min hash approach is significantly faster than the classical min hash approach.

In terms of space complexity, if all one desires is an estimate of the Jaccard index from a single pair of sets, the containment approach will use more space as a bloom filter must be formed from one of the sets. However, in the application of interest, we have one large ($n \gg m$) "reference" set B and many smaller sets $\{A_i\}_{i=1}^{M}$ each with $|A_i| \leq m$. Here, we wish to estimate $J(B, A_i)$ for each i. In this case, the additional space required to store a bloom filter is dwarfed by the space savings that come from using significantly fewer hash functions. Indeed, for S_X and S_Y the space required for the classic and containment min hash respectively, we have that $S_X \propto k_X(M+1)$ and

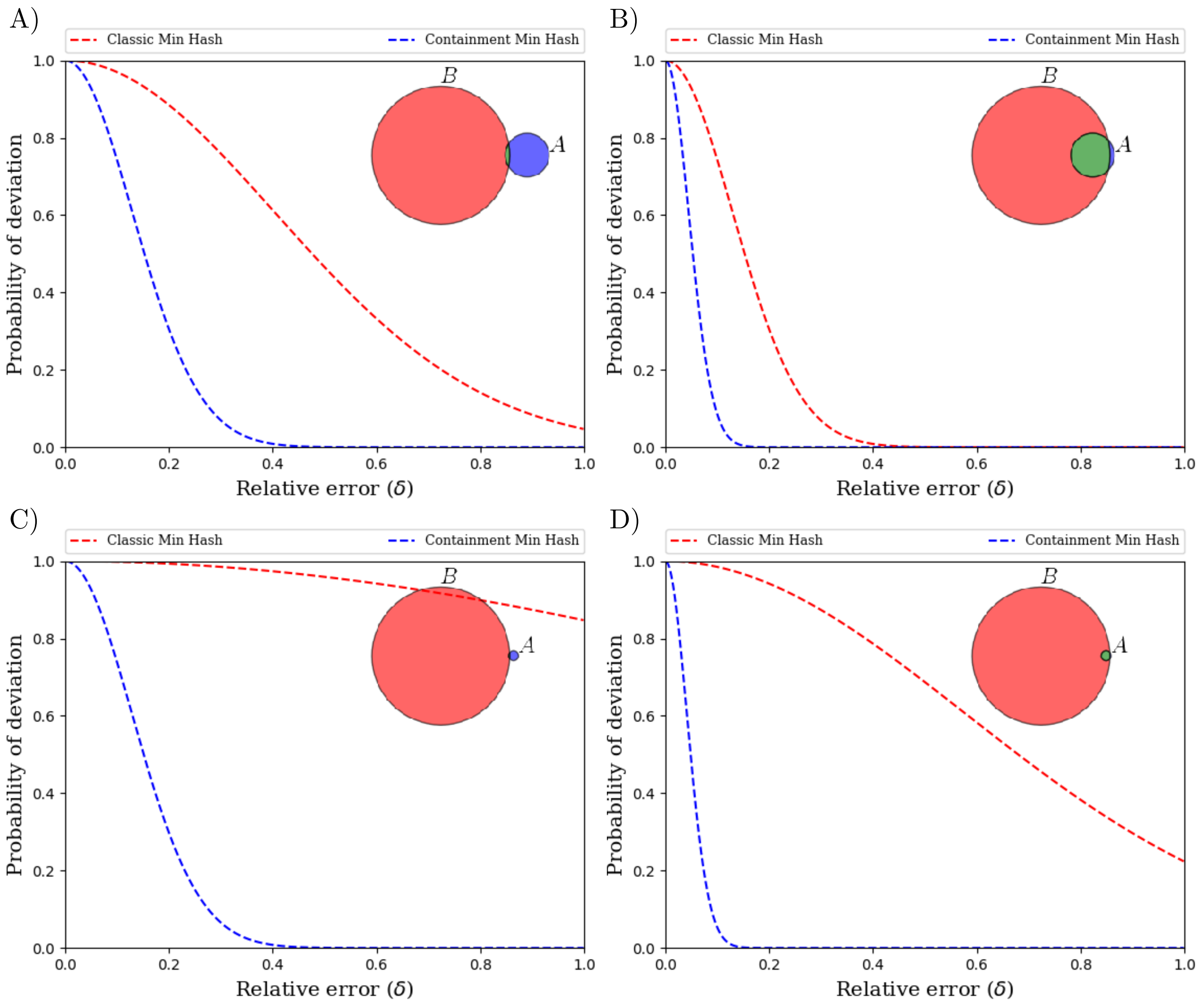


FIGURE 2. Comparison of the probability of deviation (red line: equation (2.5), blue line: Proposition (2.2)) versus the relative error δ with a fixed number of hash functions $(1,000)$. Relative sizes and Jaccard indexes of the sets in A)-D) are overlain as colored discs. Throughout, the containment min hash estimate of the Jaccard index has a much lower probability of error than the classical min hash approach. Note that in part D), there is approximately an 80% chance to have a deviation greater than 0.4 in the classic approach, whereas in the containment approach, the chance of having a deviation of greater than just 0.2 is almost zero.

$S_Y \propto k_Y + 1.44\log_2(1/p)n$ where the proportionality is in terms of the number of bits required to store a single hash value. Holding this constant fixed (as well as p and t), we then have:

$$\frac{S_Y}{S_X} \propto \frac{k_Y}{k_X} + \frac{n}{k_X(M+1)} \tag{3.5}$$

$$\leq \frac{k_Y}{k_X} + \frac{nk_Y}{k_X(M+1)} \tag{3.6}$$

$$\propto \frac{m}{n} + \frac{m}{M+1} \tag{3.7}$$

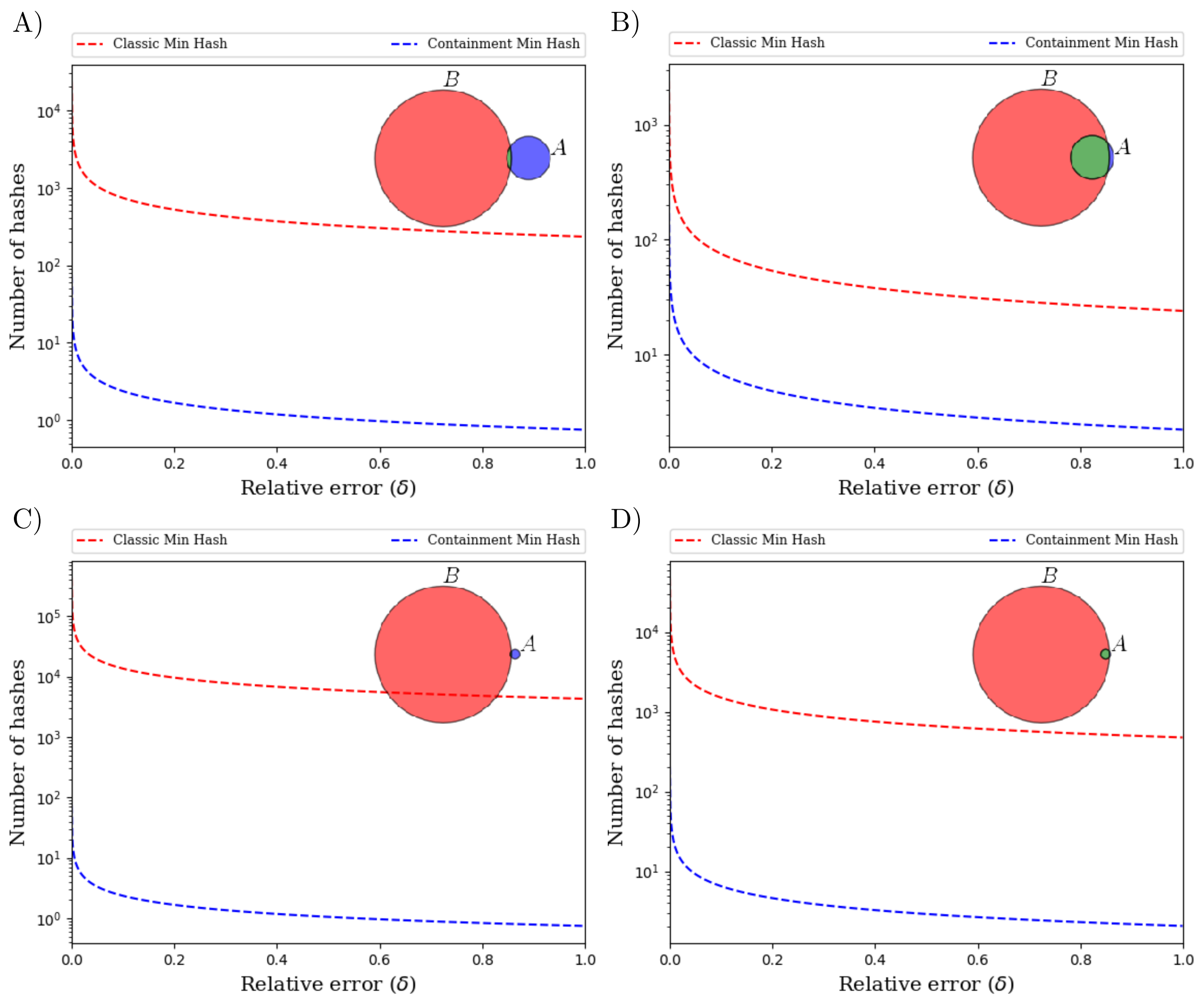


Figure 3. Comparison of the number of required hashes and relative error δ with probability of deviation $\leq 1\% := t$. The relative sizes and Jaccard indexes of the sets in A)-D) are overlain pictorially as colored discs. In all cases, the containment min hash estimate of the Jaccard index uses significantly fewer hash functions. For example, in part D), the classic min hash method (red line) needs $\approx 35,339$ hash functions to have less than 1% chance to have greater than 1% relative deviation in its estimate, whereas with the same thresholds, the containment min hash estimate of the Jaccard index (blue line) needs only ≈ 152 hash functions.

demonstrating that for large enough M, the containment approach will use less space than the classical min hash approach. A more detailed space analysis can be obtained by combining equations (2.6) and (2.9). For example when $|B| = 10^8$ and one wishes to estimate the Jaccard index of $M = 30000$ smaller sets A_i with containment $C(A_i, B) \geq 80\%$ that are small enough that $J(A_i, B) \leq 10^{-4}$ and with $t = 95\%$ confidence that the estimate is not off by more than $\delta = 50\%$, using a false positive rate of $p = 0.01$ for the bloom filter results in a 33X space savings when using the containment approach instead of the classical min hash approach.

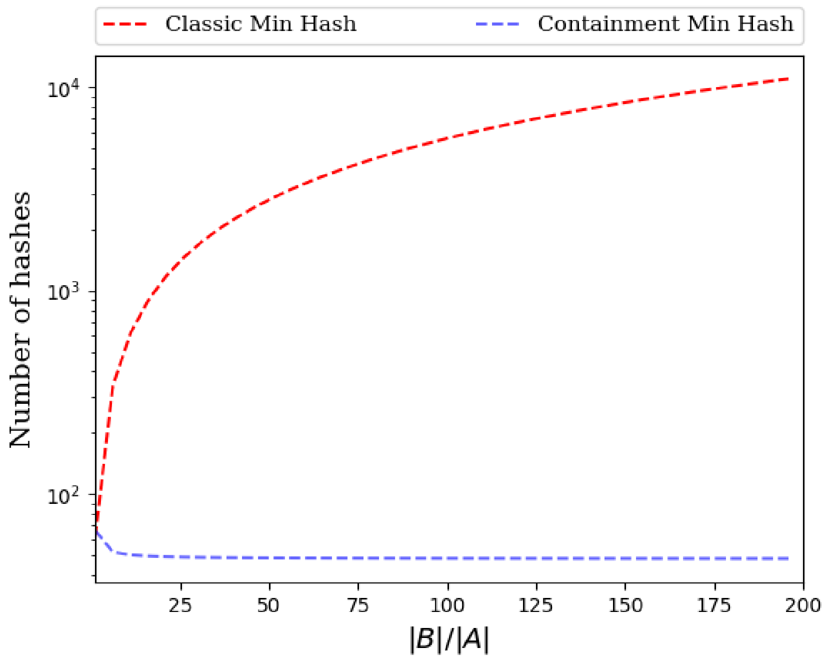


FIGURE 4. Comparison of the number of hash functions required by each method as a function of the relative sizes of the sets under consideration. The relative error $\delta = 0.1$ and the probability of deviation $t = 0.01$ are fixed. Increasing $\frac{|B|}{|A|}$, the number of required hash functions required for the classic min hash approach is increasing (red line). However, for the containment approach, the number of hash functions is nearly constant for $1 \leq \frac{|B|}{|A|} \leq 200$ (blue line) which is in agreement with equation 3.4.

3.2. **Simulated and Real Data.** In this section, we compare the classic min hash approach to the proposed containment method on real and simulated data. All code and software required to reproduce the results contained here are available at:

`https://github.com/dkoslicki/MinHashMetagenomics`.

Included in this repository is an automated script that can reproduce this paper in its entirety:

`https://github.com/dkoslicki/MinHashMetagenomics/blob/master/src/MakePaper.sh`

3.2.1. *Simulated data.* Here we illustrate the improved accuracy of the containment min hash approach over classical min hash in estimating the Jaccard index. To that end, we generated two random strings w_A and w_B on the alphabet $\{A, C, T, G\}$. We set $|w_B| = 10,000$ and $|w_A| = 15$ to simulate the situation of interest where one wishes to estimate the Jaccard index of two sets of very different size. We picked a k-mer (substring of length k) size of 11 and considered the sets A and B to be the set of all k-mers in w_A and w_B respectively. We then generated strings w_{C_i} of increasing length, formed a set C_i of all its k-mers, and considered $J(A \cup C_i, B \cup C_i)$. The number of hash functions for each method was fixed to be $k_X = k_Y = 100$. Figure 5 depicts the comparison of the containment min hash approach with the classical min hash Jaccard estimate on this data and effectively illustrates the results in section 2.2 showing improved performance of the containment min hash approach. The mean and variance of the classic min hash approach on this data was -0.001461 ± 0.001709 while using the containment approach was 0.000818 ± 0.000007 , demonstrating a substantial decrease in variance. This improved variance was observed over a range of k-mer sizes, number of hashes, and lengths of input strings.

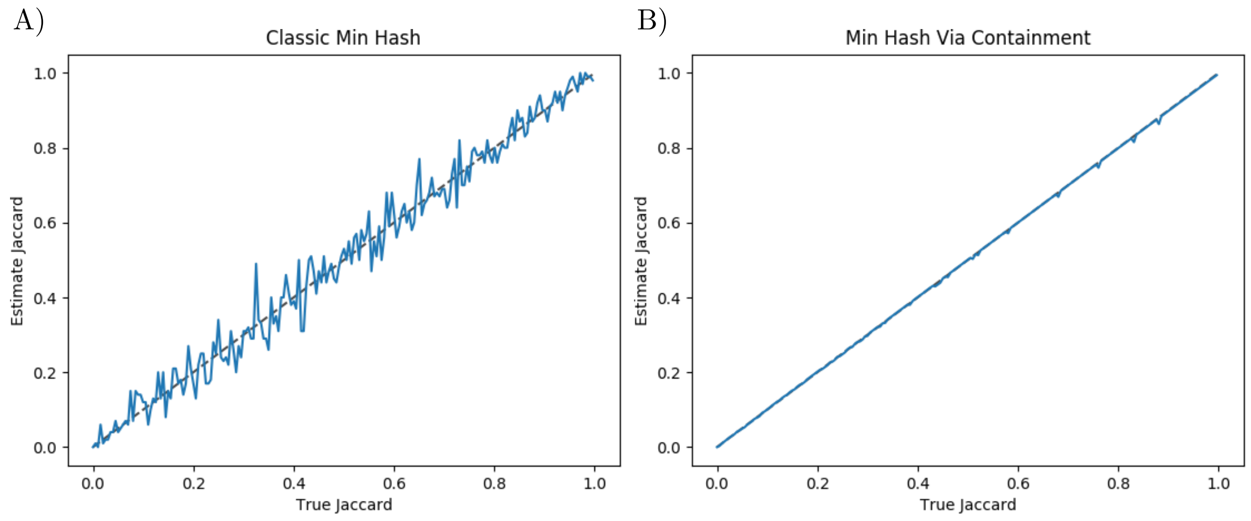


FIGURE 5. Comparison of the containment min hash approach to the classical min hash estimate of the Jaccard index on synthetic data. Each method utilized the 100 smallest hashes of the murmer3 hash function on the 11-mers of two randomly generated strings with sizes 10,000 and 15 respectively after appending a common substring of increasing size. A) Classical min hash estimate of the Jaccard index. B) Containment min hash estimate of the Jaccard index.

3.2.2. *Simulated biological data.* To demonstrate the exponential improvement of the containment min hash approach over the classical min hash for increasing sample sizes, we contrast here the mean relative performance of each approach on simulated biological data. We utilized GemSIM [15] to simulate two sets of metagenomic data from randomly selected bacterial genomes. We them aim to estimate the Jaccard index between the metagenomic data and each of the bacterial genomes (thereby simulating the case when attempting to detect if a bacterial genome appears in a given metagenomic sample).

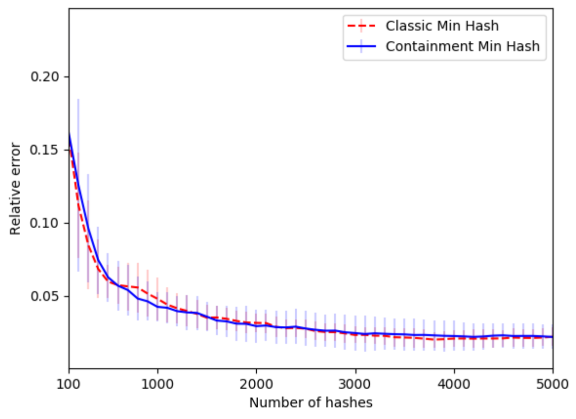
For the first set of simulated data, we used GemSIM to simulate 10K reads (of length 100) from 20 randomly selected bacterial genomes G_i (considered here as the set of all k-mers in the genome). We fixed the k-mer size to $k = 11$. We then formed a set M_1 of all 11-mers in all reads in the simulated metagenome. We then repeated this 16 times. A false positive rate of 0.001 was used for the bloom filter in the containment approach.

The second set of simulated data was produced similarly, except for the fact that we used 1M reads of the same length as before and formed a set M_2 of all 11-mers in all reads of the metagenome. Again, we then repeated this 16 times using the same false positive rate for the bloom filter.

Figure 6 depicts the relative error of the classic min hash approach and the containment approach on these two sets of simulated data when estimating $J(M_1, G_i)$ in part A), and $J(M_2, G_i)$ in part B), as a function of the number of hashes used. Observe that the containment approach has significantly less error when, as is commonly seen in practice, the number of 11-mers in the sample is appreciable in comparison to the number of 11-mers in a given reference genome G_i. This improvement of the containment approach over the classic approach continues to grow as the metagenome size grows in relation to the reference genome sizes.

3.3. **Real biological data.** Real metagenomes contain many magnitudes more k-mers than those found in any reference organisms, indicating the advantage of the containment min hash approach to determining the presence/absence of reference organisms in a given metagenome. To evaluate this, we analyzed a subset of DNA generated by the study in [13] consisting of those reads contained

A)



B)

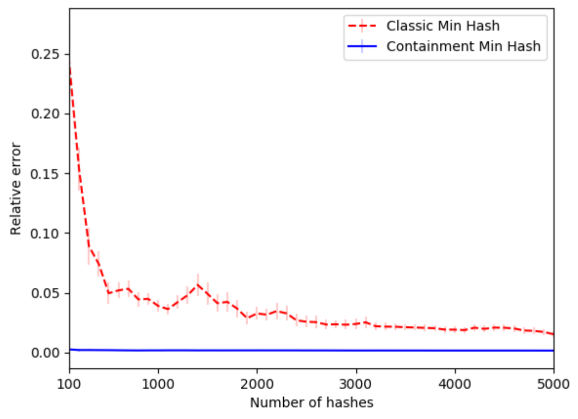


FIGURE 6. Comparison of the relative error of the containment min hash approach to the classical min hash estimate of the Jaccard index on simulated biological data. a) On 16 replicates of samples consisting of 20 genomes G_i with only 10K reads, showing the similarity of the methods in estimating $J(M_1, G_i)$ when the sets to be compared are roughly the same size. b) On 16 replicates of samples consisting of 20 genomes G_i with 1M reads, demonstrating the improvement of the containment approach in estimating $J(M_2, G_i)$ when the sets are of significantly different size.

in the sample 4539585.3.fastq. This sample consisted of 25.4M reads with average length of 65bp. We formed a bloom filter consisting of all 21-mers of this sample and formed 500 hashes from each of 4,798 viral genomes obtained from NCBI [23]. Utilizing the proposed containment min hash approach, we found the largest containment index between the reference viral metagenomes and the sample to be 0.0257 for the virus *Sauropus leaf curl disease associated DNA beta* which corresponds to a Jaccard index of 2.398e-08. Note that with this small of a Jaccard index, the results in Section 3.1.1 show that the classic approach would need millions of hash functions to accurately estimate this quantity.

To evaluate if this extremely low-abundance organism is actually present in the sample, we utilized the SNAP alignment tool [24] to align the sample to the *Sauropus leaf curl disease associated DNA beta* genome. The script *MakeCoveragePlot.sh* provides the exact commands and parameters used to perform the alignment. We found that 288 reads aligned with a MAPQ score above 20 (i.e. high-quality alignments). The coverage of the viral genome is depicted in Figure 7 using a square-root scale and a window size of 10. These high-quality mapped reads to such a small genome lends evidence to support the claim that this particular virus is actually present in the sample metagenome.

4. Conclusion

In this manuscript, we introduced "containment min hash": an improvement to the min hash approach of estimating the Jaccard index. We derived exact formulas for the probability of error for this method and also showed that it is faster, more accurate and uses less space in many situations of practical interest. This improved approach was used to analyze simulated and real metagenomic data and was found to give results superior to those of the classic min hash approach. As the theoretical results we obtained are agnostic to the application of interest, we believe this method will be useful in any application where estimates of similarity of very differently sized sets is desired. Hence, this advancement can be useful outside of the field of metagenomics and computational

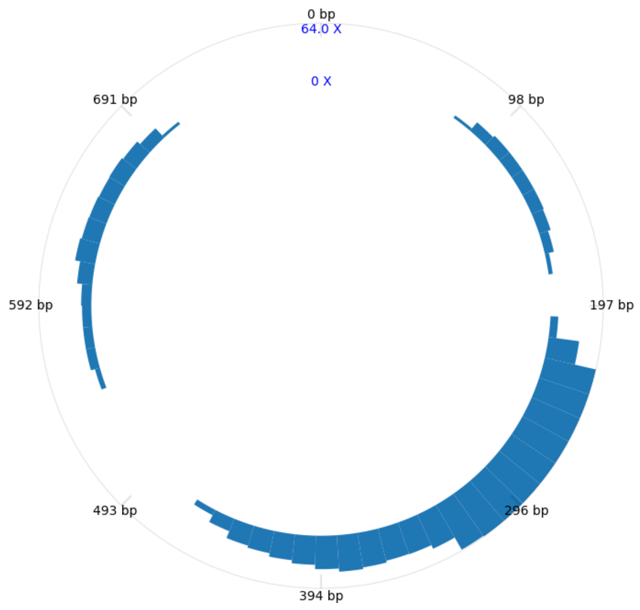


Figure 7. Plot of the real metagenomic sample alignment coverage to the virus *Sauropus leaf curl disease associated DNA beta* detected by the proposed containment min hash approach. A total of 288 reads aligned with a MAPQ score above 20 (i.e. high-quality alignments) using the SNAP aligner [24]. A square root scale and a window size of 10 was used for the plot, resulting in an average per-window coverage of 24.217X.

biology, with possible applications in data mining, web clustering or even near duplicate image detection.

References

[1] M. A. Bender, M. Farach-Colton, R. Johnson, R. Kraner, B. C. Kuszmaul, D. Medjedovic, P. Montes, P. Shetty, R. P. Spillane, and E. Zadok. Don't thrash: how to cache your hash on flash. *Proceedings of the VLDB Endowment*, 5(11):1627–1637, 2012.

[2] B. H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426, 1970.

[3] A. Z. Broder. On the resemblance and containment of documents. In *Compression and Complexity of Sequences 1997. Proceedings*, pages 21–29. IEEE, 1997.

[4] A. Z. Broder, M. Charikar, A. M. Frieze, and M. Mitzenmacher. Min-wise independent permutations. *Journal of Computer and System Sciences*, 60(3):630–659, 2000.

[5] T. C. Brown and L. Irber. sourmash: a library for minhash sketching of dna. *The Journal of Open Source Software*, 1(5), 2016.

[6] R. Chikhi and G. Rizk. Space-efficient and exact de bruijn graph representation based on a bloom filter. In *International Workshop on Algorithms in Bioinformatics*, pages 236–248. Springer, 2012.

[7] J. Chu, S. Sadeghi, A. Raymond, S. D. Jackman, K. M. Nip, R. Mar, H. Mohamadi, Y. S. Butterfield, A. G. Robertson, and I. Birol. Biobloom tools: fast, accurate and memory-efficient host species sequence screening using bloom filters. *Bioinformatics*, 30(23):3402–3404, 2014.

[8] E. Cohen, M. Datar, S. Fujiwara, A. Gionis, P. Indyk, R. Motwani, J. D. Ullman, and C. Yang. Finding interesting associations without support pruning. *IEEE Transactions on Knowledge and Data Engineering*, 13(1):64–78, 2001.

[9] A. S. Das, M. Datar, A. Garg, and S. Rajaram. Google news personalization: scalable online collaborative filtering. In *Proceedings of the 16th international conference on World Wide Web*, pages 271–280. ACM, 2007.

[10] B. Fan, D. G. Andersen, M. Kaminsky, and M. D. Mitzenmacher. Cuckoo filter: Practically better than bloom. In *Proceedings of the 10th ACM International on Conference on emerging Networking Experiments and Technologies*, pages 75–88. ACM, 2014.

[11] P. Flajolet, É. Fusy, O. Gandouet, and F. Meunier. Hyperloglog: the analysis of a near-optimal cardinality estimation algorithm. In *AofA: Analysis of Algorithms*, pages 137–156. Discrete Mathematics and Theoretical Computer Science, 2007.

[12] Y. Heo, X.-L. Wu, D. Chen, J. Ma, and W.-M. Hwu. Bless: bloom filter-based error correction solution for high-throughput sequencing reads. *Bioinformatics*, 30(10):1354–1362, 2014.

[13] A. C. Howe, J. K. Jansson, S. A. Malfatti, S. G. Tringe, J. M. Tiedje, and C. T. Brown. Tackling soil diversity with the assembly of large, complex metagenomes. *Proceedings of the National Academy of Sciences*, 111(13):4904–4909, 2014.

[14] P. Jarccard. Nouvelles recherches sur la distribution floral. *Bull. Soc. Vaud. Sc. Nat.*, 44(163):223–270, 1908.

[15] K. E. McElroy, F. Luciani, and T. Thomas. Gemsim: general, error-model based simulator of next-generation sequencing data. *BMC genomics*, 13(1):74, 2012.

[16] P. Melsted and J. K. Pritchard. Efficient counting of k-mers in dna sequences using a bloom filter. *BMC bioinformatics*, 12(1):333, 2011.

[17] M. Mitzenmacher and E. Upfal. *Probability and computing: Randomized algorithms and probabilistic analysis*. Cambridge university press, 2005.

[18] B. D. Ondov, T. J. Treangen, P. Melsted, A. B. Mallonee, N. H. Bergman, S. Koren, and A. M. Phillippy. Mash: fast genome and metagenome distance estimation using minhash. *Genome biology*, 17(1):132, 2016.

[19] J. Pell, A. Hintze, R. Canino-Koning, A. Howe, J. M. Tiedje, and C. T. Brown. Scaling metagenome sequence assembly with probabilistic de bruijn graphs. *Proceedings of the National Academy of Sciences*, 109(33):13272–13277, 2012.

[20] W. Pugh. Skip lists: a probabilistic alternative to balanced trees. *Communications of the ACM*, 33(6):668–676, 1990.

[21] B. Solomon and C. Kingsford. Fast search of thousands of short-read sequencing experiments. *Nature biotechnology*, 34(3):300, 2016.

[22] H. Stranneheim, M. Käller, T. Allander, B. Andersson, L. Arvestad, and J. Lundeberg. Classification of dna sequences using bloom filters. *Bioinformatics*, 26(13):1595–1600, 2010.

[23] D. L. Wheeler, T. Barrett, D. A. Benson, S. H. Bryant, K. Canese, V. Chetvernin, D. M. Church, M. DiCuccio, R. Edgar, S. Federhen, et al. Database resources of the national center for biotechnology information. *Nucleic acids research*, 36(suppl_1):D13–D21, 2007.

[24] M. Zaharia, W. J. Bolosky, K. Curtis, A. Fox, D. Patterson, S. Shenker, I. Stoica, R. M. Karp, and T. Sittler. Faster and more accurate sequence alignment with snap. *arXiv preprint arXiv:1111.5572*, 2011.