Connecting Syncmers to FracMinHash: similarities and advantages

Shaopeng Liu[1] and David Koslicki[1,2,3]

[1]Huck Institutes of the Life Sciences, Pennsylvania State University, USA
[2]School of Electrical Engineering and Computer Science, Pennsylvania State University, USA
[3]Department of Biology, Pennsylvania State University, USA

December 12, 2023

Abstract

Motivation Sketching methods provide scalable solutions for analyzing rapidly growing genomic data. A recent innovation in sketching methods, syncmers, has proven effective and has been employed for read alignment. Syncmers share fundamental features with the FracMinHash technique, a recent modification of the popular MinHash algorithm for set similarity estimation between sets of different sizes. While previous researchers have demonstrated the effectiveness of syncmers in read alignment, their potential for use in genomic analysis (for which FracMinHash was designed) has not been fully realized.

Results We demonstrate that the open syncmer sketch is equivalent to a FracMinHash sketch when applying to k-mer-based similarities, yet it exhibits superior distance distribution and genomic coverage. Moreover, we can expand the concept of k-mer truncation to open syncmers, enabling multi-resolution estimation in metagenomics as well as flexible-sized seeding for sequence comparisons.

Reproducibility All analysis scripts can be found on GitHub.

1 Introduction

K-mer-based methods are widely used in bioinformatic analyses. Especially in metagenomics, sketching methods, which involve selecting a subset of unique k-mers (called "sketch") generated from genomic data, have become indispensable, enabling efficient computational comparisons for large-scale datasets [13, 15, 9]. Two prominent methods in this category are FracMinHash[8] and Syncmers[3], both designed to generate random subsamples (sketches) of all k-kmers from arbitrary genomic sequences.

FracMinHash, an extension of the popular MinHash algorithm tailored to estimate set similarity, particularly when set sizes vary significantly, selects k-mers by applying cutoffs to the hash space to which k-mers are projected [8]. FracMinHash has gained widespread adoption in metagenomics for conducting extensive genomic comparisons such as taxonomic profiling [15], ANI estimation[19, 10]. On the other hand, Syncmers select k-mers characterized by the presence of their smallest substring at some specific location(s) [3]. Notably, Syncmers have found successful applications in read aligners, surpassing the widely-used Minimizer method in performance [18, 2, 16, 17].

It's important to note that these two methods operate in different contexts. In metagenomics, alignment-free genomic comparisons are usually performed using distinct k-mer sets, though there are algorithms based on weighted k-mers, generated from genomic sequences. On the other hand, in read alignment tools with the seed-and-extension algorithm, the duplication of k-mers is a consideration for seed matching status and an important consideration for optimizing speed and time [18]. Unless explicitly stated, we will primarily focus on the former scenario, as the latter has already been extensively discussed.

These two methods have three core attributes in common: 1) Context-free (1-Locality); the selection of a k-mer is solely determined by its own content and is not influenced by adjacent bases. Therefore, if a particular k-mer is chosen from one sequence, it will also be selected from other sequences. 2) random and fractional subsample (sketch); both methods select random sets of k-mers with sizes proportional to the original k-mer set size. 3) k-mers in each sketch are min-wise independent (discussed in section 2.2.3), allowing them to estimate the Jaccard and containment indices.

Conversely, the primary distinction between them lies in the situation of overlapping k-mers. In FracMinHash, hash values of every k-mer are independent and identically distributed. Therefore, the events of being selected into a FracMinHash sketch are independent of overlapping (i.e. k-mers sharing substrings are independent of being selected into a FracMinHash sketch). In Syncmers, however, overlapping k-mers can influence one another in terms of their likelihood of being classified as Syncmers (Fig 1). While syncmers have demonstrated their effectiveness in read alignment, their potential for use in genomic analysis has not been fully realized.

In this paper, we conduct a theoretical analysis of open syncmers in comparison to FracMinHash sketches. In practical applications, open syncmers exhibit comparable capabilities to FracMinHash for estimating Jaccard and containment indices. Furthermore, as illustrated in our previous research on multi-resolution k-mer-based estimations [12], the idea of k-mer truncation can be applied to open syncmers as well to fit various sizes to accommodate both sequence comparison and seed matching, indicating its potential for both read alignment and metagenomics analysis.

2 Method

2.1 Preliminary

2.1.1 Jaccard and containment index

In the metagenomics context, genomic comparisons are usually carried out using unique k-mer (all substrings of length k extracted from genomic sequences) sets derived from the samples, with consideration given to the weight of k-mers (e.g. for quantification). Similarity between pairs of genomic

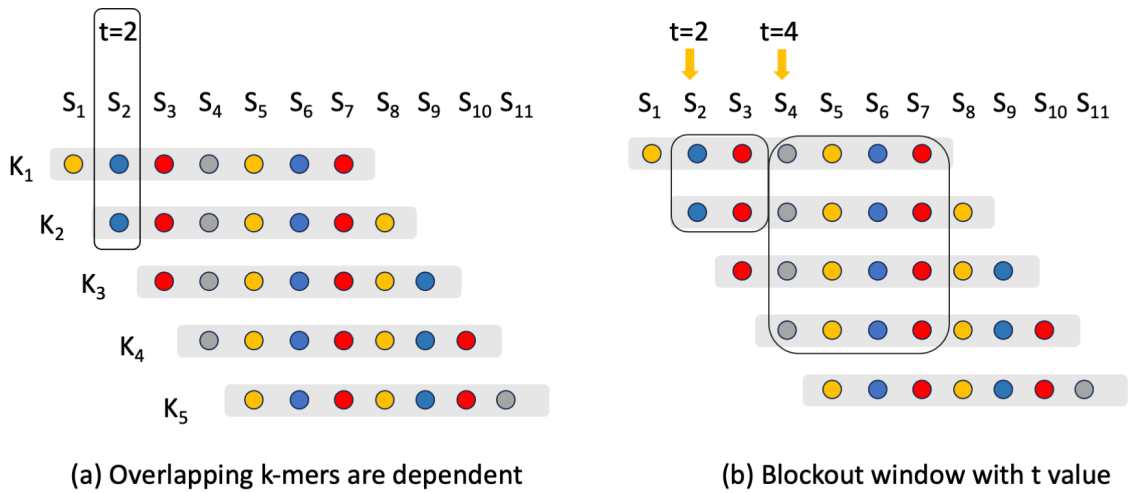

Figure 1: **Overlapping k-mers are dependent on the status of being an open syncmer.** There are 5 consecutive k-mers and each k-mer has 7 substrings (every dot represents a substring of length s). (a) An open syncmer may affect its overlapping k-mers regarding the likelihood of being an open syncmer. When $t = 2$, if K_1 is an open syncmer (i.e. $S_2 < S_3$), then K_2 must not be an open syncmer. (b) When $t > 1$, every open syncmer will have a blockout window where some downstream k-mers must not be an open syncmer. If K_1 is an open syncmer: when $t = 2$, we have $S_2 < S_3$ so K_2 must not be an open syncmer; when $t = 4$, we have $S_4 < S_5, S_6, S_7$ so K_2, K_3, K_4 must not be an open syncmer.

data can be measured by the similarity of their respective k-mer sets: the collection of all distinct k-mers appearing as contiguous substrings in the data.

The Jaccard index (JI) measures the similarity of two non-empty sets by comparing the relative size of the intersection over the union and the containment index (CI) measures the relative size of the intersection over the size of one set [1]. We use A^k to denote the set of all distinct k-mers derived from a set of sequences/strings A. When applied to sets of k-mers, they are defined as follows:

$$J(A^k, B^k) = \frac{|A^k \cap B^k|}{|A^k \cup B^k|} \text{ and } C(A^k, B^k) = \frac{|A^k \cap B^k|}{|A^k|}. \tag{1}$$

Typically, the value of k used is not small (e.g. k=21 for metagenomic profiling [15, 21]) such that it's very unlikely to have duplicate k-mers purely by chance. Nonetheless, real genomes are not random and contain repetitive regions. While duplicate k-mers are frequently encountered in practice, it is generally assumed that their occurrences are limited in number.

2.1.2 FracMinHash

The FracMinHash (FMH) sketching technique (a.k.a. mincode submer by Edgar[3], scaled MinHash by Irber et al.[8], and universe minimizer by Ekim et al.[4]) is a generalization of modulo hash that supports Jaccard and containment estimation between sets of different sizes. Instead of using a fixed sketch size, it samples k-mers of a fixed proportion from a bag of k-mers [8]. Let $[0, H]$ be the hash

space of some perfect hash function h that can randomly project k-mers into the hash space, for a pre-selected scaling factor c (i.e. reciprocal of the proportion), a FMH sketch for sequences A is:

$$\mathrm{FMH}(A^k) = \left\{ x \in A^k \ s.t. \ h(x) < \frac{H}{c} \right\} \tag{2}$$

Therefore, it's expected that $\frac{1}{c}$ of all distinct k-mers will be selected at random and the scaling factor c can be tuned to modify the sketch size. Besides, sampled k-mers are uniformly distributed along the genome (assuming no repeat regions) as every k-mer has an equal probability of being selected along the genome.

2.1.3 Open syncmers

Open syncmers (OS) represent a k-mer sampling approach involving the random selection of k-mers based on the relative order of their substrings. Open syncmers are defined by a *triple* (k, s, t): a k-mer becomes an open syncmer if its smallest (relative to some hash function) substring of length s ($s < k$) shows up at a specific location t. Using $(k = 5, s = 3, t = 1)$ as an example, a 5-mer $AAGCT$ is an open syncmer because it has 3 substrings $h(AAG) < h(AGC) < h(GCT)$ where the 1st substring is the smallest if the h simply reflects the alphabetical order. An open syncmer sketch for sequences A is:

$$\mathrm{OS}(A^k) = \{ x \in A^k \ s.t. \underset{m \in [1, k-s+1]}{argmin} \ h(x[m : m + s - 1]) = t \} \tag{3}$$

Intuitively, a k-mer will have $(k - s + 1)$ substrings of random order. It will be selected as an open syncmer with a probability of $\frac{1}{k-s+1}$. This setting is similar to the FracMinHash technique except the dependence of overlapping k-mers: they are not independent of each other regarding being an open syncmer because they share substrings (Fig 1).

Various types of syncmers exist [3]. We opted for open syncmers because they already offer an excellent genome compression ratio and genome coverage; and they have demonstrated successful applications in minimap2[11] and Strobealign[16]. A generalized approach, parameterized syncmer scheme[2], has been discussed to include an arbitrary number of locations for the smallest substring, with open syncmers representing the case of only one substring location, closed syncmers representing the case with two fixed substring locations, etc.

Syncmers can be further downsampled to accommodate a larger compression factor by employing the FracMinHash technique (referred to as "Mincode submer" in the original manuscript) [3]. Using a downsampling ratio d ($d \geq 1$), a random k-mer will be selected as an open syncmer with a probability of $\frac{1}{d(k-s+1)}$. While keeping in mind that open syncmers can adapt to arbitrary scaling factors with followup downsampling, we assume $d = 1$ (no downsampling) for the following discussion without loss of generality.

Next, we will demonstrate, both theoretically and through practical examples, how an open syncmer sketch compares to a FracMinHash sketch for genomic comparisons.

2.2 Compare the open syncmer sketch to the FracMinHash sketch

In this section, we will compare the open syncmer sketch to the FracMinHash sketch from four perspectives: compression factor, k-mer spread, compatibility with the MinHash algorithm, and multi-resolution estimation.

2.2.1 Compression factor can be identical for both sketches

The compression factor (a.k.a. scaling factor) is calculated as the ratio of all unique k-mers to the number of selected k-mers in a sketch of a long random string. Following the same argument as

Edgar [3], we can simply show that both FracMinHash and open syncmers share the same expected compression factors by linearity of expectation.

Theorem 1. $\mathrm{OS}(A^k)$ *and* $\mathrm{FMH}(A^k)$ *have the same expected sketch size if* $c = k - s + 1$.

Proof. Let $A^k = \{K_1, K_2, ..., K_n\}$ be the set of all distinct k-mers derived from a set of sequences/strings A. Let $I_M(K)$ be the binary indicator for the event that a given k-mer K is selected via a sketching method M. Assuming that hash values of all substrings/k-mers are randomly and uniformly distributed:

$$E(I_{\mathrm{OS}}(K_i)) = \frac{1}{k-s+1} = \frac{1}{c} = E(I_{\mathrm{FMH}}(K_i)) \quad \text{for } i = 1 \dots n. \tag{4}$$

By linearity of expectation, no matter the dependence of overlapping k-mers, the expected sketch sizes for both methods are:

$$E(|\mathrm{OS}(A^k)|) = \sum_{i=1}^{n} I_{\mathrm{OS}}(K_i) = \sum_{i=1}^{n} I_{\mathrm{FMH}}(K_i) = E(|\mathrm{FMH}(A^k)|). \tag{5}$$

In section 3.1, we demonstrated this result in real data.

2.2.2 The open syncmer sketch provides better k-mer spread than the FracMinHash sketch

A subsampling method should avoid selecting k-mers with extensive overlaps or clustering (on genomes), as such k-mers provide less information compared to more widely distributed k-mers [6]. We can use a discrete random variable, denoted as D, to measure the distances between the starting index of two consecutive k-mers along the genome (or a long string) within a given k-mer sketch.

Corollary 1.1. *Both sketches have the same expected k-mer distance if* $c = k - s + 1$.

Proof. When $c = k-s+1$, the expected sketch sizes for both methods are the same based on Theorem 1. Considering a long random string A with n distinct k-mers in total and assuming a low repeat level, $E(|\mathrm{OS}(A^k)|) = E(|\mathrm{FMH}(A^k)|) = \frac{n}{c}$.

The total distances between all adjacent k-mers in the sequence sum up to the entire genome length, except a few bases preceding the first k-mer and trailing bases after the last k-mer. When the genome size is large, the total distances approximate the value of n (assuming a low duplication level). Consequently, the average distances for both sketches are equivalent:

$$\overline{D} \approx \frac{n}{|\mathrm{FMH}(A^k)|} = c = \frac{n}{|\mathrm{OS}(A^k)|} \tag{6}$$

In practice, however, the average distances for both sketches are usually smaller than the expected value due to duplicated k-mers.

We can further derive the distribution of distance between adjacent k-mers along the genome in the FracMinHash sketches using the negative binomial model. Each k-mer has a fixed and independent probability of being selected into a FracMinHash sketch. We can view this as a sequence of identical and independent Bernoulli trials along the genome, where "success" corresponds to being selected into the sketch. Therefore, the nucleotides located between two consecutive k-mers represent the "failures" that precede the subsequent "success". This concept leads to the subsequent theorem, which provides insights into the distribution of distances within the FracMinHash sketch.

Theorem 2.

$$D_{\mathrm{FMH}} - 1 \sim \mathrm{NB}\left(1,\ \frac{1}{c}\right), \text{ where } c \text{ is the scaling factor.} \tag{7}$$

However, obtaining an elegant closed-form expression for the distance distribution among k-mers in the open syncmer sketch is challenging due to the previously mentioned interdependence. This interdependence plays a crucial role in enhancing the k-mer spread of open syncmers. In an open syncmer setting with parameters (k, s, t), although t is not a free parameter when optimizing genome coverage[18], the expected sketch size and the average distance are not affected. As we will see in the next section, tuning the value of t can ensure a "window-free scenario" where there is a lower bound for the distances between adjacent k-mers along the genome (Figure 1). Combined with the fact that the average distance remains unaffected, the open syncmer sketch with optimal window-free feature will exhibit a more favorable distance distribution.

Open syncmers may have lower bound for distances between adjacent k-mers In the context of open syncmers with parameters $(k, s, t > 1)$, every open syncmer creates a blocking window such that the next several consecutive k-mers must not be open syncmers (Fig 1b). This blocking event occurs when the t-th substrings of both the open syncmer (referred to as K_1) and the other k-mer (referred to as K_x) are covered in their overlaps. Using $K[i]$ to denote the i-th substring in a k-mer, the fact that K_1 is an open syncmer implies that $K_1[t] < K_x[t]$, making it impossible for K_x to be an open syncmer. The size of this blocking window is bounded by its distance from the two terminal substring of a k-mer, as exceeding this distance would leave one of the t-th substrings uncovered by the overlapping segment. Consequently, the blocking window is symmetrical and achieves its maximum length when the t value is positioned at the midpoint of the k-mer. Namely, $t = \lceil \frac{k-s+1}{2} \rceil$ gives the optimal situation where every 2 consecutive open syncmers must be at least t bases away.

This observation aligns with the conclusion made by Shaw and Yu in [18, Theorem 8] that the optimal choice of t value is the middle of the k-mer and this gives the best genome coverage. Please note that selecting the optimal t value, while it improves base-level conservation with more scattered k-mers, does not impact sketch size or k-mer-level conservation (Figure 4b).

Syncmers do have the feature of window guarantee, a situation where a k-mer must be selected within a window of some length. But it has less practical relevance for open syncmers as the maximum possible distance can be large [3].

Conjecture 3. *Open syncmer sketches can provide better k-mer spreads at the same compression level.*

Proof. When we increase the value of t for open syncmers until the middle point, the expected sketch size (Theorem 1) and the average distances (Equation 6) remains the same.

However, the "window-free" feature discussed above will eliminate all small distances, resulting in a distribution more centered around the mean (less extreme values).

2.2.3 The open syncmer sketch is suitable for the MinHash algorithm

In the context of the FracMinHash algorithm, some proper hash function h is employed to uniformly and randomly project arbitrary k-mers into hash space $[0, H]$. This feature serves as the cornerstone of the MinHash algorithm [1]. Although not explicitly stated, it's crucial to note that this randomness is unrelated to the value of k. In other words, the hash values of two k-mers are independent, whether they share some substrings (overlapping) or one is a substring of the other. This property is known as the "avalanche property", which characterizes a scenario where a minor alteration in the input causes a substantial alteration in the output, making it statistically indistinguishable from randomness (the murmurhash family are such functions with excellent avalanche properties) [5]. In brief, we have for $n \geq 1$ and $h : \{A, C, G, T\}^n \to [0, H]$:

$$h \overset{iid}{\sim} Unif\{0, H\}. \tag{8}$$

With this approximation, k-mers within the open syncmer sketch are uniformly and randomly distributed in the hash space. This is because each k-mer is selected based on the hash values of its

substrings, which are unrelated to the hash value of the original k-mer. Restating these observations as a theorem, we have:

Theorem 4. *K-mers in the syncmer sketch are uniformly randomly distributed in the hash space.*

Remember that both sketching methods choose k-mers based solely on their own k-mer sequences (1-locality), ensuring that if a specific k-mer is selected from one sequence, it will also be chosen from other sequences. Define $h_{min}(S)$ to be the minimal member of set S with respect to some hash function h, we have the same feature of the open syncmer sketch to the FracMinHash sketch to estimate set similarities because the probability that any k-mer in the open syncmer sketch possesses the minimum hash value is equal among all k-mers.

Corollary 4.1.

$$Pr(h_{min}(\mathrm{OS}(A^k)) = h_{min}(\mathrm{OS}(B^k))) = \frac{|\mathrm{OS}(A^k) \cap \mathrm{OS}(B^k)|}{|\mathrm{OS}(A^k) \cup \mathrm{OS}(B^k)|} \tag{9}$$

In other words, the open syncmer sketch is well suited in the context of k-mer set similarity comparison as it is a random sample of all distinct k-mers, similar to MinHash [1]. This observation aligns with the principle of the FracMinHash algorithm. In a similar context, Shibuya et al. demonstrated that closed syncmers are comparable to random sampling and can provide reliable estimations for Jaccard Indices [20, Figure 1].

Estimate Jaccard and containment indices Open syncmer sketches can be used to estimate Jaccard and containment index, just as MinHash sketches can. We then compare it to the FracMinHash sketch for estimating set similarity. Following the definition of FracMinHash given by Irber et al (2022) [8], we can get the estimate of Jaccard and containment indices via the FracMinHash sketch:

$$\hat{J}_{\mathrm{FMH}}(A^k, B^k) = \frac{|\mathrm{FMH}(A^k) \cap \mathrm{FMH}(B^k)|}{|\mathrm{FMH}(A^k) \cup \mathrm{FMH}(B^k)|}, \;\; \hat{C}_{\mathrm{FMH}}(A^k, B^k) = \frac{|\mathrm{FMH}(A^k) \cap \mathrm{FMH}(B^k)|}{|\mathrm{FMH}(A^k)|} \tag{10}$$

Similarly, the estimation by the open syncmer sketch:

$$\hat{J}_{\mathrm{OS}}(A^k, B^k) = \frac{|\mathrm{OS}(A^k) \cap \mathrm{OS}(B^k)|}{|\mathrm{OS}(A^k) \cup \mathrm{OS}(B^k)|}, \;\; \hat{C}_{\mathrm{OS}}(A^k, B^k) = \frac{|\mathrm{OS}(A^k) \cap \mathrm{OS}(B^k)|}{|\mathrm{OS}(A^k)|} \tag{11}$$

Please keep in mind that these estimations are not unbiased [7, Theorem 1, Supplementary A.3], but we may ignore the error term in practice as it is exponentially small with respect to the set sizes (which are appreciably large).

2.3 Multi-resolution application with the open syncmer sketch

In our prior work CMash [12], we illustrated that the MinHash sketch is capable of offering multi-resolution estimations of Jaccard and containment indices by truncating a longer k-mer into shorter lengths. Now we broaden the k-mer truncation idea to include open syncmers for multi-resolution applications, encompassing both similarity estimation and seed matching. As previously discussed, we use $t = \lceil \frac{k-s+1}{2} \rceil$ for the open syncmers for the optimal genome coverage.

A multi-resolution open syncmer sketch (MOS) is defined by $(k, k_{max}, s, t = \lceil \frac{k-s+1}{2} \rceil)$: a k_{max}-mer X is selected by MOS if its center substring of length k is a $(k, s, t = \lceil \frac{k-s+1}{2} \rceil)$ open syncmer (i.e. $X[\frac{k_{max}-k}{2} + 1, \frac{k_{max}+k}{2}]$ is an open syncmer).

$$\mathrm{MOS}(A^{k_{max}}) = \{x \text{ if } x \in A^{k_{max}} \text{ and } \underset{m \in [\frac{k_{max}-k}{2}+1, \frac{k_{max}+k}{2}-s+1]}{argmin} h(x[m : m+s-1]) = t\} \tag{12}$$

Then, MOS can facilitate k-mer-based applications with any k values between $[k, k_{max}]$ by simply truncating itself, eliminating the necessity to repeat the sketching/seeding step. When $k_{max} = k$, equation 12 is exactly the definition of open syncmers and MOS is just the regular open syncmer sketch. In other cases, we can get an open syncmer sketch with new but smaller k sizes (preferably the difference is even) by simply truncating the MOS until length k.

$$\text{MOS}(A^{k_{max}-2}) = \{x[2, k_{max} - 1] \; \forall x \in \text{MOS}(A^{k_{max}})\} \tag{13}$$

Coupled with a k-mer ternary search tree, we can store a single sketch for use across a range of k values, and conveniently modify the k size through prefix lookups. In addition to enabling multi-resolution estimations of k-mer-based similarities, the multi-resolution open syncmer sketch (MOS) offers a means to balance mapping quality and speed when utilizing a sketching based seeding approach in a seed-and-extend alignment algorithm. As demonstrated by Shaw and Yu, the use of open syncmers in minimap2 increases sensitivity but can lead to more repetitive seed matches [18]. Implementing multi-resolution seed matching has the potential to reduce repetition while preserving the sensitivity of smaller k values.

3 Results

In this section, we assess the performance of the open syncmer sketch in comparison to the FracMinHash sketch. We obtained 20 random genomes in the Brucella genus, as these genomes exhibit a broad spectrum of Jaccard Index (JI) values, from the GTDB database [14] to serve as reference sequences.

3.1 Compression factor and k-mer conservation are consistent across open syncmers and FracMinHash sketches

The expected sketch size can be made equal if we choose parameters in such a way that $c = k - s + 1$. In our comparison, we set $k = 20$ and $s = 11$ for the open syncmer sketch and $c = 10$ for the FracMinHash sketch, resulting in both having an expected compression ratio of 10. We count via brute-force all unique k-mers across all genomes and compared them to the sketches we generated. Similarly, k-mer level conservation (measured by the ratio of k-mers recovered from a mutated genome with 0.05 uniform random mutation rate) is consistent between FracMinHash sketches and open syncmers with different t values.

The results are presented in Figure 2, revealing that the compression factors align with our expectations and remain consistent irrespective of the value of t. Since open syncmers comprise a random selection from all unique k-mers, we can anticipate that k-mer conservation will be similar to what we obtain from FracMinHash. As discussed previously, adjusting the value of t only affects genome coverage, not k-mer conservation.

3.2 Similarity estimation

We proceeded to evaluate the performance of using open syncmers for Jaccard and containment estimation, as outlined in the previous section. Following the construction of FracMinHash and open syncmer sketches for all 20 genomes (with parameters: $k = 20,\ s = 11,\ t = 5,\ c = 10$), we estimated their pairwise similarities using open syncmers and FracMinHash sketches. Ground truth values were determined through brute-force calculations of all k-mers from the reference genomes. In theory, open syncmer sketches should produce identical results (Section 2.2.3) and this is indeed what we observed in Figure 3.

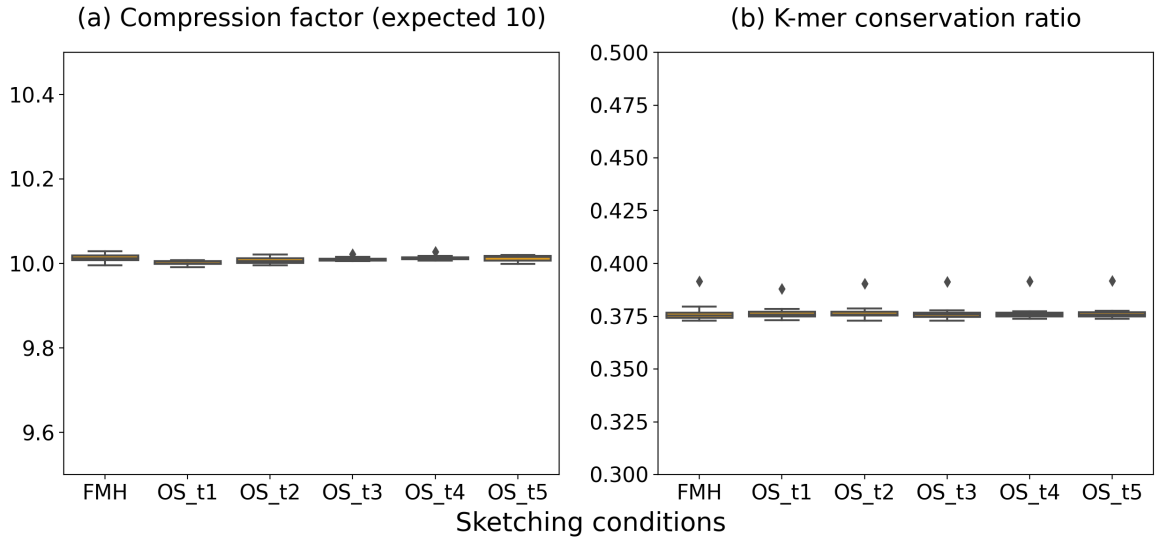

Figure 2: **Open syncmer vs FracMinHash: compression ratio and k-mer conservation are consistent.** Boxplot of compression ratio for the FracMinHash sketch with $c = 10$ and the open syncmer sketch with $k = 20, s = 11$ and t ranging in $t = 1 \ldots 5$. By choosing appropriate parameters, both the FracMinHash sketch and the open syncmer sketch can achieve (a) the same expected sketch size and (b) comparable k-mer level conservation.

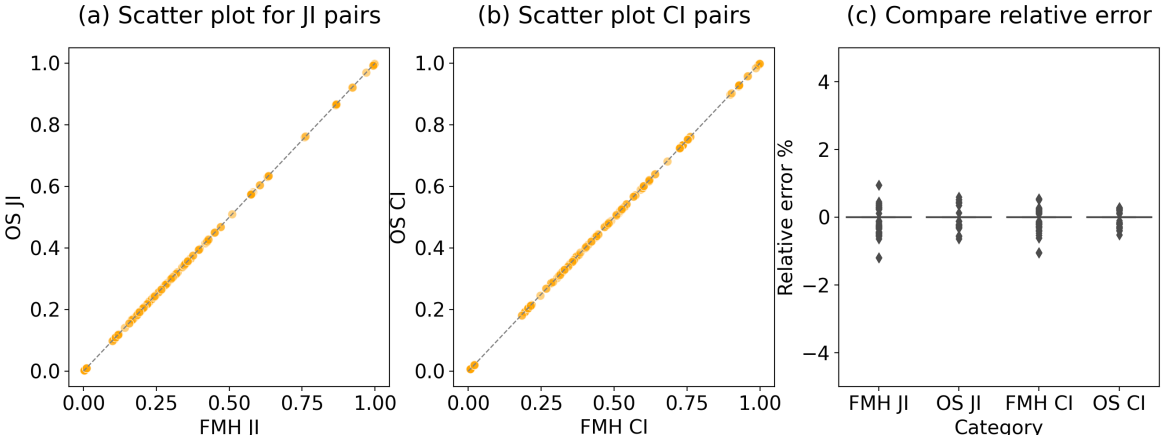

Figure 3: **The open syncmer sketch fits the MinHash algorithm.** (a) The scatter plot of Jaccard Index (JI) estimation by open syncmers and FracMinHash. The data points align closely along the $y = x$ line, indicating a high degree of consistency between the two methods. (b) The scatter plot of containment index estimation by open syncmers and FracMinHash. (c) Relative error (in percentage) for all JI and CI estimations are small and comparable between the two methods.

3.3 Distance distribution

The results above aligns with our expectation that the open syncmer sketch is on par with the FracMinHash sketch when it comes to Jaccard and containment estimation (i.e. k-mer conservation between two sets). While the status of k-mer matching remains constant, it's noteworthy that open syncmers offer a window-free guarantee, a situation where 2 consecutive k-mers in the sketch are at

least t base apart until t reaches the midpoint of the k-mer.

This window-free feature, though not affecting k-mer conservation, will increase overall genome coverage. It has been proven by Shaw and Yu that the maximum genome coverage is achieved when t is the midpoint [18, Theorem 8]. In figure 4 we illustrated that genome coverages in the 20 random genomes increase with increased t value ($k = 12,\ s = 5$); and the distance distribution will shift towards the mean to give a more evenly distributed coverage.

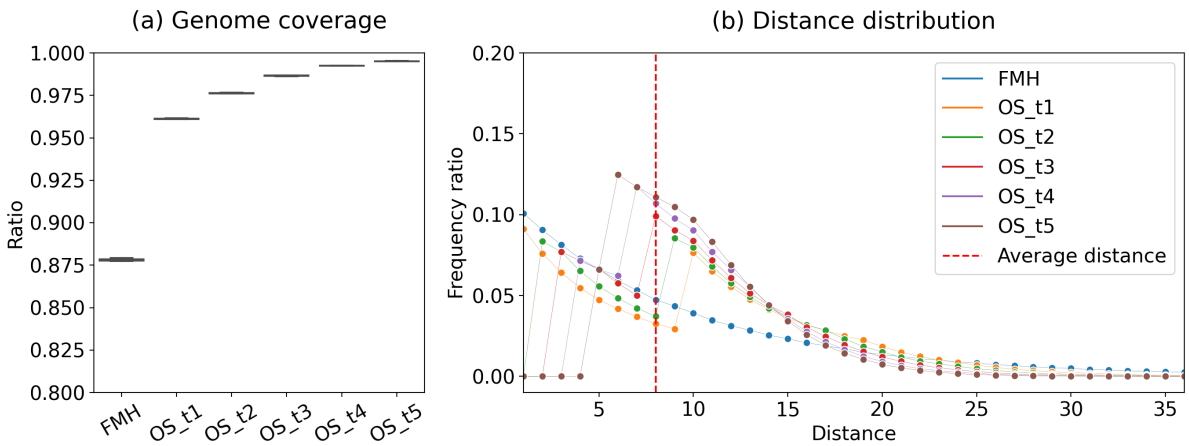

Figure 4: **Genome coverage and distance distribution** (a) Boxplot of genome coverage by the FracMinHash sketch with $c = 10$ and the open syncmer sketch with $k = 20, s = 11$ and t ranging in $t = 1...5$. Open synmcers provide better genome coverage when the value of t increases. (b) Lineplot for the average distances between adjacent k-mers along the genome, under the same condition. Distances are more centered around the mean when the value of t approaches its optimal location (OS_t5). And we can observe the "window-free" event for different t values.

3.4 Multi-resolution estimation and seed matching

With parameters ($k = 20,\ k_{max} = 30,\ s = 11,\ t = 5$), we created multi-resolution open syncmer (MOS) sketches for the 20 random genomes and subsequently assessed k-mer conservation across various mutation rates for a spectrum of k values.

For each reference genome, we utilized a simple random point mutation model to generate ten mutated genomes. Subsequently, we generated Open Syncmer (OS) sketches and multi-resolution Open Syncmer (MOS) sketches with the same k value and employed them independently to compute the average k-mer conservation between the original genome and the mutated genomes.

Our observations indicate that the MOS sketches exhibit similar performance to regular open syncmers, implying their capacity to offer multi-resolution similarity estimation without the need to reiterate the sketching workflow with a different k size (Fig 5). This insight also presents an optimization strategy for addressing duplicate seed matching in read aligners: we can sketch with a larger k value and scale it down to accommodate an appropriate matching status; or we may expand hits with multiple matches to minimize duplication.

4 Conclusion

This article presents a comparison between two sketching methods: open syncmers and FracMinHash. Open syncmers have found successful applications in seed construction for read aligners, while the FracMinHash algorithm offers considerable power in metagenomic analysis with set similarity estimation. Our demonstration reveals that open syncmer sketches are equivalent to the FracMinHash

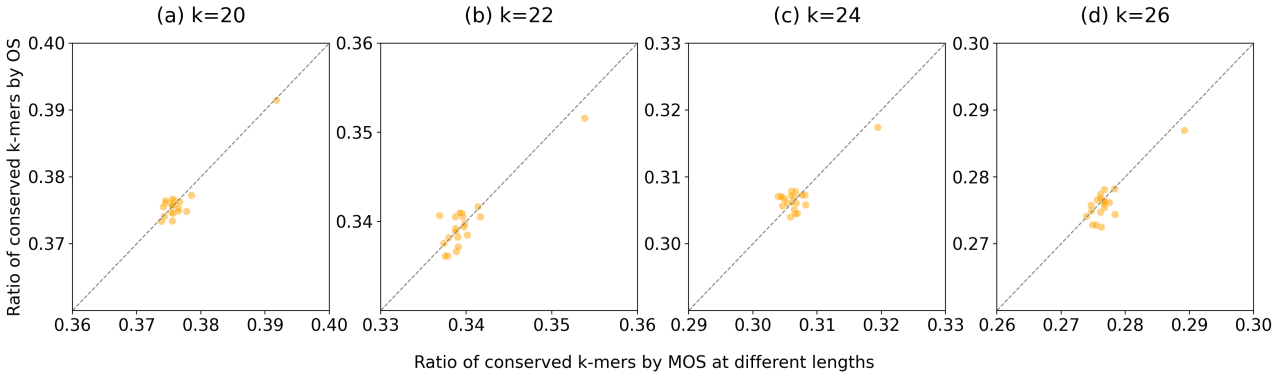

Figure 5: **Multi-resolution open syncmers are comparable to regular open syncmers** We used the parameter $k = 20, k_{max} = 30, s = 11, t = 5$ to build the multi-resolution open syncmer sketches and compared their k-mer conservation to regular open syncmers at (a)$k = 20$, (b)$k = 22$, (c)k=24, and (d)k=26. In addition to the variability introduced by the random mutation process, the multi-resolution estimation aligns closely with regular open syncmers, highlighting the capacity of MOS to facilitate multi-resolution estimation.

sketch within the context of k-mer-based set similarity estimation, despite the fact that overlapping k-mers are not independent of being selected as open syncmers. However, the open syncmer sketch is considered a random sample (though not unique) from the set of k-mers of a genome because all hash values are uniformly and randomly distributed (Section 2.2.3).

This dependence of overlapping k-mers, as observed in open syncmers, offers specific advantages. It allows us to establish a "window-free" guarantee, ensuring that the distance between consecutive k-mers has a minimum value. This leads to a more evenly distributed distance pattern (still having the same average distance), providing higher genome coverage. This attribute is particularly beneficial for seed matching in read alignment.

Furthermore, we have extended our previous work related to multi-resolution k-mer-based estimation to incorporate open syncmers, resulting in a novel sketching strategy for multi-resolution open syncmers (MOS). MOS sketches maintain the same capability to the regular open syncmer sketches for estimating k-mer conservations and can be employed for multi-resolution similarity estimation, as well as for seed matching in alignment tasks.

4.1 Future direction

The analyses described above can be extended to other syncmer settings, such as closed syncmers[3] or PSS[2]. However, the performance of open syncmers is already satisfactory. The combination of being "window-free" with a large t value and having a constant expected size ensures that: most distances between adjacent k-mers will be smaller than k, resulting in a high genome coverage (Figure 4).

One challenge when applying open syncmers in read alignment practice is the issue of duplication. Kristoffer tried to address this concern by utilizing open syncmers in combination with strobermers for seed construction [16]. Our work on MOS offers an alternative approach: since we can easily modify seed sizes without compromising performance (Figure 5), MOS could be integrated into the seed matching step of the read aligner, potentially mitigating the duplication problem.

References

[1] A. Z. Broder. On the resemblance and containment of documents. In *Proceedings. Compression and Complexity of SEQUENCES 1997 (Cat. No. 97TB100171)*, pages 21–29. IEEE, 1997.

[2] A. Dutta, D. Pellow, and R. Shamir. Parameterized syncmer schemes improve long-read mapping. *PLOS Computational Biology*, 18(10):e1010638, 2022.

[3] R. Edgar. Syncmers are more sensitive than minimizers for selecting conserved k-mers in biological sequences. *PeerJ*, 9:e10805, 2021.

[4] B. Ekim, B. Berger, and R. Chikhi. Minimizer-space de bruijn graphs: Whole-genome assembly of long reads in minutes on a personal computer. *Cell systems*, 12(10):958–968, 2021.

[5] C. Estébanez, Y. Saez, G. Recio, and P. Isasi. Performance of the most common non-cryptographic hash functions. *Software: Practice and Experience*, 44(6):681–698, 2014.

[6] M. C. Frith, L. Noé, and G. Kucherov. Minimally overlapping words for sequence similarity search. *Bioinformatics*, 36(22-23):5344–5350, 2020.

[7] M. R. Hera, N. T. Pierce-Ward, and D. Koslicki. Deriving confidence intervals for mutation rates across a wide range of evolutionary distances using fracminhash. *Genome Research*, pages gr–277651, 2023.

[8] L. Irber, P. T. Brooks, T. Reiter, N. T. Pierce-Ward, M. R. Hera, D. Koslicki, and C. T. Brown. Lightweight compositional analysis of metagenomes with fracminhash and minimum metagenome covers. *bioRxiv*, pages 2022–01, 2022.

[9] C. Jain, L. M. Rodriguez-R, A. M. Phillippy, K. T. Konstantinidis, and S. Aluru. High throughput ani analysis of 90k prokaryotic genomes reveals clear species boundaries. *Nature communications*, 9(1):5114, 2018.

[10] D. Koslicki, S. White, C. Ma, and A. Novikov. Yacht: an ani-based statistical test to detect microbial presence/absence in a metagenomic sample. *bioRxiv*, pages 2023–04, 2023.

[11] H. Li. Minimap2: pairwise alignment for nucleotide sequences. *Bioinformatics*, 34(18):3094–3100, 2018.

[12] S. Liu and D. Koslicki. Cmash: fast, multi-resolution estimation of k-mer-based jaccard and containment indices. *Bioinformatics*, 38(Supplement_1):i28–i35, 2022.

[13] B. D. Ondov, G. J. Starrett, A. Sappington, A. Kostic, S. Koren, C. B. Buck, and A. M. Phillippy. Mash screen: high-throughput sequence containment estimation for genome discovery. *Genome biology*, 20:1–13, 2019.

[14] D. H. Parks, M. Chuvochina, C. Rinke, A. J. Mussig, P.-A. Chaumeil, and P. Hugenholtz. Gtdb: an ongoing census of bacterial and archaeal diversity through a phylogenetically consistent, rank normalized and complete genome-based taxonomy. *Nucleic acids research*, 50(D1):D785–D794, 2022.

[15] N. T. Pierce, L. Irber, T. Reiter, P. Brooks, and C. T. Brown. Large-scale sequence comparisons with sourmash. *F1000Research*, 8, 2019.

[16] K. Sahlin. Strobealign: flexible seed size enables ultra-fast and accurate read alignment. *Genome Biology*, 23(1):260, 2022.

[17] K. Sahlin, T. Baudeau, B. Cazaux, and C. Marchet. A survey of mapping algorithms in the long-reads era. *Genome Biology*, 24(1):1–23, 2023.

[18] J. Shaw and Y. W. Yu. Theory of local k-mer selection with applications to long-read alignment. *Bioinformatics*, 38(20):4659–4669, 2022.

[19] J. Shaw and Y. W. Yu. Fast and robust metagenomic sequence comparison through sparse chaining with skani. *bioRxiv*, pages 2023–01, 2023.

[20] Y. Shibuya, D. Belazzougui, and G. Kucherov. Efficient reconciliation of genomic datasets of high similarity. In *22nd International Workshop on Algorithms in Bioinformatics (WABI 2022)*, volume 242 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 14:1–14:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.

[21] D. E. Wood and S. L. Salzberg. Kraken: ultrafast metagenomic sequence classification using exact alignments. *Genome biology*, 15(3):1–12, 2014.