Entropy predicts fuzzy-seed sensitivity

Benjamin Dominik Maier[1][0000−0001−8442−0536] and Kristoffer Sahlin[1][0000−0001−7378−2320]

Department of Mathematics, Science for Life Laboratory, Stockholm University, 106 91, Stockholm, Sweden
ksahlin@math.su.se

Abstract. In sequence similarity search applications such as read mapping, it is desired that seeds match between a read and reference in regions with mutations or read errors (sensitivity) but do not produce redundant matches due to repeats (specificity). K-mers are likely the most well-known and used seed construct in bioinformatics, and many studies on, *e.g.*, spaced k-mers aim to improve sensitivity and specificity over k-mers. Recently, we developed a fuzzy seeding construct, strobemers, which were empirically demonstrated to have high sensitivity and specificity, but the study lacked a deeper understanding of why.

In this study, we demonstrate that the entropy of a *seed cover* (a stretch of neighboring seeds) is a good predictor for seed sensitivity. We propose a model to estimate the entropy of a seed cover, and find that seed covers with high entropy typically have high match sensitivity. We also present two new strobemer seed constructs, mixedstrobes, and altstrobes. We use both simulated and biological data to demonstrate that mixedstrobes and altstrobes improves sequence matching sensitivity to other strobemers. We implement strobemers into minimap2 and observe slightly faster alignment time and higher accuracy than using k-mers at various error rates.

We believe the most important aspect of this work is our discovered seed stochasticity-sensitivity relationship. The relationship provides a clear explanation of why some fuzzy seeds perform better than others and a framework for designing even more sensitive seeds. In addition, we show that the two new seed constructs, mixedstrobes, and altstrobes, are practically useful. Finally, in cases where our entropy model does not predict the observed sensitivity well, we explain why and how to improve the model in future work.

Keywords: Sequence mapping · seeds · k-mers · strobemers · Information theory.

1 Introduction

Short k-length substrings of a sequence, often referred to as k-mers, are widely used for sequence comparison in bioinformatic applications. A k-mer that is shared by two sequences implies an identical region of size k, and with appropriate length on k, we may detect similar but non-identical regions through shared k-mers. Some of the reasons that k-mers are often used for sequence similarity detection is because they are fast to construct, and due to their fixed length easy to represent, store, and query, *e.g.*, with hash tables, or more succinct data structures such as bloom filters [5], the FM-index [15], and many more [32]. As k-mers indicates shared sequences, they are often used as markers, or *seeds*, indicating regions for more extensive similarity comparison, *e.g.*, through pairwise alignment.

With the broad use of k-mers as seeds, several limitations have also been identified. For example, k-mers are sensitive to mutations. If k is too small, we may obtain many redundant hits (low *specificity*), *e.g.*, due to repeats). On the other hand, a too large k may destroy all matches (low *sensitivity*) in mutation-dense regions or error-prone reads. Detailed modeling of k-mers' sensitivity to substitutions at different rates was performed in [3]. Some studies have proposed altering the underlying biological sequence to reduce the mutation rates with, e.g., homopolymer compression [2] or modifying the mutation distribution using more advanced sequence transformations [4]. However, most work has been aimed at increasing seed sensitivity and specificity of k-mers by proposing alternative *seed constructs*.

1.1 Other seed constructs

Some approaches aim to alleviate specificity issues in downstream analysis by dynamically extending the k-mers to provide a less redundant set of matching seeds such as Maximal Exact Matches (MEMs), Maximal Unique Matches (MUMs) [11], MCAS [25], and Context-aware seeds [50]. These seeding constructs have been referred to as dynamic seeds [40] as they are neither fixed in length nor in the number of CPU cycles for their construction. There are also seeding constructs known as *subsampling methods* that aim to use only a subsample of k-mers as seeds due to their redundant nature using, *e.g.*, minimizers [36] or later subsampling techniques [10,19,13,51,12,20]. For an extensive study of subsampling techniques, see [43].

To overcome the issue of requiring only exact matches, spaced seeds (or spaced k-mers) [31], covering template families [21], and SimHash-based constructs [8] such as permutation-based seeds [28] or BLEND [18] have been proposed that, particularly, tolerate substitutions. Seeds that do not require an identical sequence to match are often called *fuzzy seeds*. In applications where substitutions are frequent, spaced k-mers have had practical success and are used in several state-of-the-art applications such as in the general sequence similarity search software BLAST [1], and for metagenomic classification [7] and long-read mapping [44].

Recently, a new class of substitution and indel tolerant seed constructs, *strobemers* [37], was proposed. Strobemers expand on the ideas of neighbouring minimizer pairs [9,41] and k-min-mers [14]. Strobemers are constructed by linking together a set of smaller k-mers and can be constructed with several different methods to link the k-mers (minstrobes, randstrobes, hybridstrobes), yielding different properties. It was shown that Strobemers could offer higher sensitivity and specificity over k-mers, and they have been used for short-read mapping [38], long-read overlap detection [18], and transcriptomic long-read normalization [33].

1.2 Previous work on seed sensitivity

In spaced seed literature, seed sensitivity has been extensively studied. Typically, when using seeds, an alignment is triggered if a certain number of seeds match in a region, *e.g.*, through requiring either multiple *hits* [6] (seed matches) or a single hit [26] in a region. One drawback of requiring multiple hits is that a threshold does not distinguish highly overlapping hits from disjoint ones. For this reason, seed coverage (union of matching positions in a region) has been proposed [34].

The main conclusion in spaced seed literature is that many highly overlapping seeds are redundant, uninformative, and can lead to unnecessary computations for sequence matching applications. Typically, the aim is to select a set of seed patterns consisting of fixed and wildcard positions that overlap or correlate as little as possible. Related work on minimizing the overlap of hits has been studied in the form of clump statistics [45] or *overlap complexity* [24]. In addition, there are other theoretical studies of seed sensitivity quantifying the correlation between seeds [27] or using generating functions from analytical combinatorics [16], which have also been used in practice to select suitably spaced seeds when mapping short reads [17].

1.3 Our aim

The aforementioned spaced-seed studies are all based on seeds with a fixed sampling pattern (k-mers and spaced k-mers). That is, the sampling decision has no stochasticity (randomness) after the seed pattern has been chosen. Flexible gapped seeds such as strobemers employ a random sampling decision based on the underlying sequence. These seeds are a new class of flexible fuzzy seeds that are more tolerant to indels and open up a venue to study how stochasticity influence seed overlap, correlation, and sensitivity. Also, while there are spaced seed studies focusing on optimal seed selection of a single seed pattern [22], most aforementioned spaced seed studies are centered around selecting a set of spaced seeds with complementary properties. We focus on seed efficacy restricted to one seed per position as is used for k-mers, for query efficacy purposes.

Inspired by spaced k-mer literature that has focused on understanding the mechanics of high sensitivity seeds, we aimed to find why fuzzy seeds such as strobemers achieve beneficial properties such as high sensitivity and specificity. We focused on seed constructs that, similarly to k-mers, (1) produce one seed per position, (2) always sample k fixed positions in a window of w positions (i.e., no dynamic size construction such as MEMs or MCAS), and (3) require only a single look-up in the index for match detection. Here, we call these seeds (k, w)-seeds. For example, k-mers, spaced k-mers, and strobemers are all valid (k, w)-seeds. Constraint 1 allows for fair sensitivity and specificity benchmarking. Constraint 2 ensures that seed construction is constant across string composition for easy benchmarking (i.e., not sensitive to repeats as dynamic seeds). Constraint 3 ensures that, after construction, the seeds are equally fast for sequence similarity comparison (e.g., through a hash table look-up).

1.4 Paper outline and our contributions

In section 2.1, we state the notation and preliminaries. In section 2.2, we formalize the notion of (k, w)-seeds and a seed cover. In section 2.3 and 2.4 we present two new seed constructs that we call *mixedstrobes* and *altstrobes*. In section 2.5, we describe how we measure seed sensitivity and specificity. In section 2.6, we discuss the main finding in this paper, namely that the entropy (randomness) of a seed cover can be used as a predictor of seed sensitivity. Specifically, in most cases, higher entropy seed covers have a higher probability of generating at least one match in a region that has undergone mutations. We provide a simplified estimate to model the entropy of a seed cover. In section 3.1, we empirically verify that, in most cases, a higher predicted entropy of a seed cover results in higher sensitivity for various seed constructs. We also discuss the limitations of our predictor and future improvements. In sections 3.2-3.3 we evaluate how our two proposed new seeding constructs, *mixedstrobes* and *altstrobes*, fares to randstrobes and k-mers in an actual sequence matching scenario with the four different metrics used in [37]. Of immediate practical importance, we find that mixedstrobes and altstrobes can improve over the currently best-known strobemer construct (randstrobes). In section 3.4, we implement strobemer seeds in minimap2. Even though minimap2 performs subsampling of seeds which distorts the entropy and, thus, our sensitivity predictions, we observe faster alignment time (up to 30%) and slightly higher sensitivity (0.2%) than using k-mers of the same size as seeds. Section 4 and 5 discuss the results and describes future direction on how to use our entropy predictor for seed design. In summary, our work presents two new state-of-the-art seeding constructs, mixedstrobes and altstrobes, and our insight and model can be used to design even better seeds than those proposed in this study.

2 Methods

2.1 Notation

We define a *subsequence* of a string as a set of ordered nucleotides that can be derived from the string by removing some or no elements while keeping the order of the remaining elements. If all letters in the subsequence are consecutive, we refer to it as a substring. We use 0-indexed string notation. We typically use S and T to denote strings, and we use the notation $S[i : j]$, $i < j$ to refer to a substring starting at position i and ending and including the character at position j in S. We let the $|\cdot|$ operator denote the length of strings. Finally, we use h to denote a hash function mapping strings to integers.

2.2 Fuzzy seeds and seed covers

At a high level, with a fuzzy seed, we mean a subsequence extracted from a string S that is guaranteed to match another fuzzy seed extracted from another string T if $S = T$ and may match T if T is similar but not identical to S. Seeds considered in this study are produced as follows. Let k and w be two positive

integers with $k \leq w$ where k denotes the number of distinct positions sampled in a substring of length w in a string S. Let $f(i, k, w, S, *)$ denote some function that starts at position i in S and extracts a subsequence of characters at k distinct positions in the substring $S[i : i+w]$ using only the information in $S[i : i+w]$. We use the final argument $*$ to denote any seeding specific parameters, e.g., $(n, \ell, w_{min}, w_{max})$ for strobemers.

Informally, f returns a subsequence of length k of a string of length w, i.e., a fuzzy seed which can be used for sequence matching. For $k = w$, there is only one way to sample a subsequence from $S[i : i + w]$, namely k consecutive characters, *i.e.*, a k-mer, which is not a fuzzy seed. When $k < w$, fuzzy seeds such as spaced k-mers [31] and strobemers [37] are possible constructs. We will refer to seeds constructed with parameters k and w as a (k, w)-seed. We will refer to w consecutive (k, w)-seeds produced by f in a region of length $2w$ as a *seed cover*. Note that the w consecutive seeds in this cover are all sampled within a string of length $2w$. For example, the w seeds extracted from positions $[i : i + w]$ in $S[i : i + 2w]$ is a seed cover of $S[i : i + 2w]$

Constraints on f We impose the following three basic constraints on f to be viable for sequence matching.

C1 f produce the same (k, w)-seeds for two strings S and T if $S = T$.
C2 f produce valid (k, w)-seeds $\forall S, S \in \Sigma^*$.
C3 at most one seed is produced per position in a sequence.

C1 and C2 are necessary for sequence matching. An example of a construct that violates C2 is "sample the position if the letter is A or C" because there may not be enough A's and C's in the window. C3 limits querying to at most one lookup per position making the constructs efficient. We have intentionally described f in a general fashion in order to encompass more general construction techniques. For example, a k-mer would deterministically sample the first k nucleotides (nt), regardless of the size of w. A randstrobe [37] with parameters (2,15,25,50) is a valid $(30, 64)$-seed because the maximum span of sampled positions in the randstrobe is $w = 50 + 15 - 1 = 64$ and f would sample 15nt at $S[i : i + 15]$ as the first strobe, then sample the next strobe starting somewhere in $S[i + 25 : i + 50]$.

E-hits of seeds The definition of E-hits was given in [38] and is a measure of how repetitive the seeds in a query sequence are, on average, in a reference dataset. More specifically, the E-hits computes the expected number of hits that seeds constructed from a query sequence obtained uniformly at random from the reference will have. E-hits can be calculated for any seeding mechanism and reference dataset. For a given reference dataset, let N be the total number of seeds sampled, M the total number of distinct seeds sampled, and z_i be the total number of times the distinct seed i $(1 \leq i \leq M)$ is sampled. Then E-hits is computed as follows

$$\text{E-hits} = \frac{1}{N} \sum_{i=1}^{M} z_i^2 \tag{1}$$

2.3 Mixedstrobes

We here describe a new type of seeding construct that we refer to as mixedstrobes. Mixedstrobes samples either a k-mer or a strobemer at a specified fraction. Any strobemer may be sampled (minstrobes, hybridstrobes and randstrobes), but we will only consider randstrobes here. Analogous to strobemers, we parameterize mixedstrobes as $(n, \ell, w_{min}, w_{max}, q)$, where n is the number of strobes n, ℓ is the strobe length, w_{min} and w_{max} the minimum and maximum downstream offset to last window, and q the strobemer fraction. Whether a strobemer or a k-mer is seeded depends on the hash value of the first strobe $h(S[i : i + \ell])$ and the user-defined strobe fraction q. The strobe fraction q is represented as numerator N and a denominator D (e.g., $q = 0.6$ is represented as $N = 60$ and $D = 100$) so that

$$f(i, k, w, S, *) = \begin{cases} \text{Sample strobemer,} & \text{if } h(S[i : i + \ell]) \ \% \ D < N \\ \text{Sample k-mer,} & \text{otherwise} \end{cases}$$

The full pseudocode to construct mixedstrobes is given in the Suppl. section S1.

2.4 Altstrobes

We refer to our second novel seed construct as altstrobes. Altstrobes are modified randstrobes where the strobe length alternates between shorter and longer strobes. Instead of having two strobes of length $k/2$ as implemented in strobemers of order 2, altstrobes consist of one short strobe k_s one longer strobe k_l, with $|k_s| + |k_l| = k$. We parameterize altstrobes as $(n, |k_s|, |k_l|, w_{min}, w_{max})$. We refer to sampled altstrobe with $n = 2$ as $(|k_s|, |k_l|)$ or $(|k_l|, |k_s|)$, depending on if the short strobe was used first or second, respectively. Whether the first strobe is of length $|k_s|$ or $|k_l|$ is decided based on the hash value of the substring of length $|k_s|$ (*i.e.*, the potential first strobe). Specifically,

$$f(i, k, w, S, *) = \begin{cases} \text{Sample altstrobe } (k_s, k_l), & \text{if } h(k_s) \ \% \ 2 = 0 \\ \text{Sample altstrobe } (k_l, k_s), & \text{otherwise} \end{cases}$$

Similarly to other strobemers, the selection of the downstream strobe is based on a window of size $w_{max} - w_{min}$ nucleotides, and the selection is performed with the randstrobe linking function as described in [37]. The sampled strobe length is decided by the hash value of the shorter strobe. Otherwise, mutations within the positions $[k_s, k_l]$ downstream from the start position may lead to seeds being sampled differently between two sequences.

For fair benchmarking to other strobemer seeds, we implement two evaluation-specific constraints on altstrobes. Firstly, n has to be even to guarantee seeds with the same number of sampled positions. Secondly, to guarantee (k, w)-seeds, we adjust the sampling window offset depending on if it is the long or short strobe we sample first. Specifically, we let k_l in altstrobe (k_s, k_l) be sampled from $[w_{min} - (k_l - k_s)/2, w_{max} - (k_l - k_s)/2]$ and k_s in altstrobe (k_l, k_s) be sampled from $[w_{min} + (k_l - k_s)/2, w_{max} + (k_l - k_s)/2]$. These constraints are only implemented for controlled benchmarking. The full pseudocode to construct altstrobes is given in the Suppl. section S1.

2.5 Objectives for sequence similarity detection

Our objective is similar to what was sought but not thoroughly formulated in [37]. We state the objectives in precise terms here. Let two strings S and T, each of length $2w$ have an edit distance $m \geq 0$ to each other. Let $N_m(k, w)$ be the number of seed matches from the first w consecutive (k, w)-seeds constructed from S and T (see Fig. 1A). We desire a function f that extract fuzzy (k, w)-seeds such that:

O1 $P(N_m(k, w) > 0)$ is as large as possible $\forall m \geq 0$.
O2 The E-hits metric [38] for f is as small as possible.

O1 relates to seed sensitivity, and O2 relates to seed precision. The formulation of $N_m(k, w)$, namely to only consider the first w seeds in a region of $2w$ for short strings, may seem unfair to k-mers. This is because K-mers can produce additional hits between S and T from the last $w - k$ seeds at the ends of S and T (see Fig. 1A). However, this advantage is present only at the ends of sequences. We aim to model a scenario where sequences are substantially longer than the extra $w - k$ seeds in the ends, *e.g.*, as for long reads. Therefore, O1 reflects all regions but the w-long end region of sequences.

2.6 Stochasticity influence seed sensitivity

In [37], it was shown that, e.g., randstrobes and hybridstrobes had higher sensitivity (better at finding matches) than k-mers, spaced k-mers and minstrobes (Fig. 1 and Tab. 1 in [37]). These two constructs have, unlike the rest, a stochastic component in how they select the next strobe, creating a seemingly more random seed cover (see Fig. 1A). In turn, randstrobes had a higher sensitivity than hybridstrobes. By definition of the constructs, the randomness in sampled positions for the second strobe is higher for randstrobes than for hybridstrobes. Therefore, it stands to reason that something in the stochasticity of the sampled positions of a seed cover may be positively correlated with seed sensitivity. To formulate it in a verifiable question, does the probability of having at least one match in a region, $P(N_m(k, w) > 0)$, increase with higher seed cover entropy? For the impatient reader, the answer to this question will be, no, not in all cases, but it is in general a good predictor.

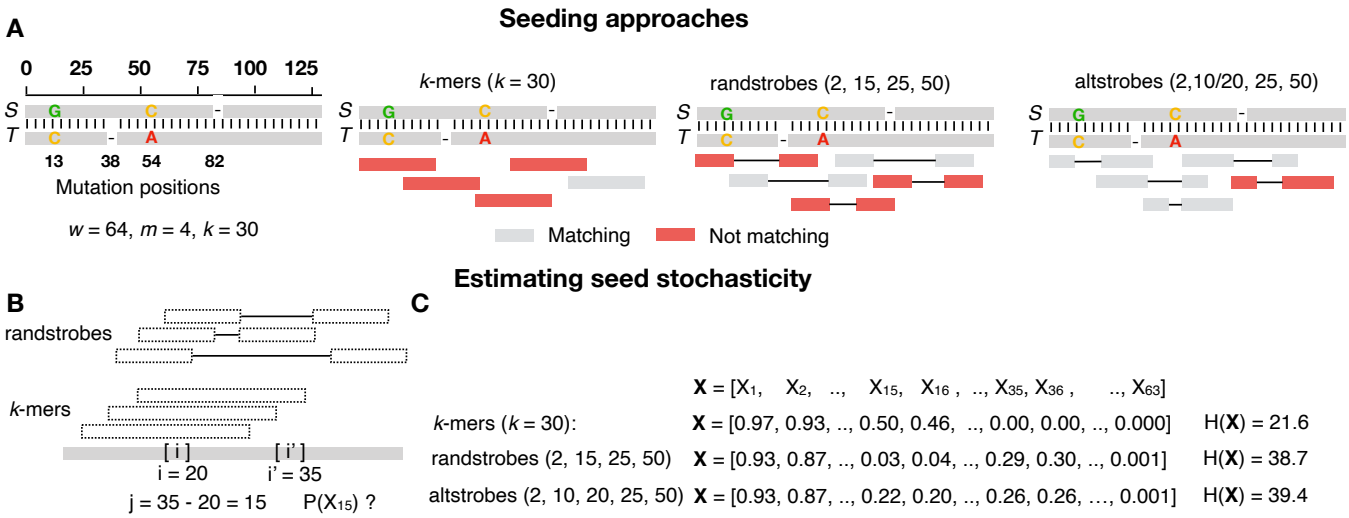


Fig. 1: Sampling of k-mers, randstrobes and altstrobes. Panel A shows a scenario where k-mers, randstrobes and altstrobes are sampled over two similar sequences S and T different by four mutations. Only a subset of five seeds in the region is shown for clarity. In the example, the seed constructs all sample 30 fixed positions in a window of $w = 64$ nucleotides which is the maximal span of a randstrobe with parameters (2,15,25,50). The mutations are distributed such that they destroy all shared k-mers in the region, and most of the randstrobes. Altstrobes have the possibility to sample k-mers of two different lengths at each site, which allow them a higher probability to match between mutations. Panel B illustrates the scenario we use to predict the entropy of a seed cover. If we draw a seed at random out of all the seeds that cover position i, we want to find the probability that this seed also covers position i'. Panel C illustrates example computations of this probability for all reachable downstream positions i' for the scenario and seed constructs in panel A. A high but even probability across positions in $\mathbf{X}$ signals that the seed is high in stochasticity (where k/w is theoretical optimal probability across $\mathbf{X}$).

Modelling entropy of a seed cover At least, the entropy of a seed cover depends on (i) the probability of sampling a position within the window of size w in a seed, (ii) the distribution over coupled positions within a seed, and (iii) the correlation of positions between neighboring seeds. A model for i-iii would, perhaps fully, measure all aspects of the entropy of a seed cover, which is our quantity of interest. Let us denote this quantity $H(Z)$ where Z is some stochastic quantity from a model of i-iii. Unfortunately, we have not found a way to measure iii for general seed constructs. One of the reasons this is tricky is that some stochastic seed constructs pass information (are correlated) between seeds (e.g., hybridstobes) as they use minimizers [36], which are correlated between neighboring windows. However, we constructed a simplified estimate for i and ii.

We estimate the entropy of a seed cover as follows. If we randomly select a (k, w)-seed that covers a given position i, what is the probability that this seed covers another position i', nearby? Assume a (k, w)-seed produced by f samples a position i in a sequence. Let $i' > i$ with $j = i' - i$ and X_j be the binary variable describing the event that the position i' is sampled given that i was sampled ($X_j = 1$). Then $P(X_j)$ is the probability that position i' is sampled given that i is sampled. See Figure 1B) for an example scenario.

Our estimate of seed cover entropy in a region is not restricted to seed constructs with stochasticity in the sampling decision. We will also use it to predict sensitivity for k-mers (see example computation in Fig. 1C). As stated, our measure can not estimate entropy for seeds sharing information between seeds for stochastic decisions, e.g., minstrobes and hybridstrobes. Nevertheless, we will see that the estimate will be a useful predictor for randstrobes and some other fuzzy seed constructs.

Computing $P(X_j)$ We introduce two variables to help us with this calculation. Let $c \in [0, k-1]$ be the 0-indexed coordinate of the k sampled positions within the (k, w)-seed that covers position i. Furthermore, let k_ℓ, $\ell = 1, \ldots, L$ be the contiguous ordered strings that a (k, w)-seed is made up from. For example, a k-mer has $L = 1$, and a strobemer of order n we have $L = n$. Let Y_ℓ be the binary variable denoting the event that c is placed on string ℓ. We have

$$P(X_j) = \sum_{c=0}^{k-1}\sum_{\ell=1}^{L} P(X_j|Y_\ell, c)P(Y_\ell, c) = \sum_{c=0}^{k-1}\sum_{\ell=1}^{L} P(X_j|Y_\ell, c)P(Y_\ell|c)P(c) = \frac{1}{k}\sum_{c=0}^{k-1}\sum_{\ell=1}^{L} P(X_j|Y_\ell, c)P(Y_\ell|c). \quad (2)$$

Here $P(c) = 1/k$ by the assumption that the position that cover i is equally likely to be any of the k positions within a seed. Also, $P(Y_\ell|c)$ becomes an indicator variable equal to 1 if c is positioned on segment k_ℓ and 0 otherwise. The probability, $P(X_j|Y_\ell, c)$ differs depending on the seed construct, but can be structured up into cases and is relatively straight forward to compute. We provide example computations for k-mers, randstrobes, altstrobes, and mixedstrobes in Suppl. section S2.

Computing estimated entropy of a seed cover using $P(X_j)$ $P(X_j)$ is concerned with a specific position j. We are interested in all positions j in a neighborhood of position i. It suffices to consider $X_1, \ldots X_w$ as $P(X_j) = 0$ for $j > w$ for a (k, w)-seed. We let $\mathbf{X} = (X_1, \ldots, X_w)$ be a binary random vector and use notation $P(X_j) = p_j$ (see Fig. 1B). We have that each X_j is a Bernoulli random variable and is associated with an individual probability p_j of success. Furthermore, all realizations of $\mathbf{X}$ have exactly k 1's. The entropy [42] of $\mathbf{X}$, $H(\mathbf{X})$, is computationally infeasible to estimate, but we use the approximation given in equation 5 in [29].

$$H(\mathbf{X}) \approx \sum_{j=1}^{w} H(X_j) - C = \sum_{j=1}^{w} p_j \log p_j - C \quad (3)$$

Where C is a positive constant from an upper bound approximation [29]. Since C is the same constant estimate for any (k, w)-seed, we omit it when comparing the approximations of the entropy for different seeds. However, the accuracy of the approximation of C may be different among the seed constructions (i.e., depending on the p_j's). An upper bound of $H(\mathbf{X})$ occurs when all X_j are independent with $p_j = k/w$. In this case, equation 5 is exact with C=0. We can now estimate $H(\mathbf{X})$ for different seeds by calculating p_j from expression 3 (see Fig. 1C for an example).

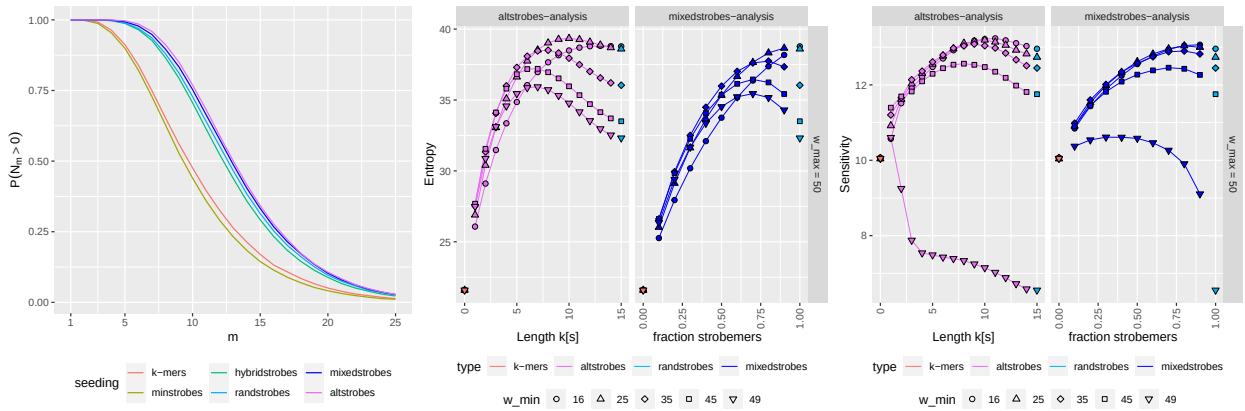
3 Results

3.1 Empirically testing the link between entropy and sensitivity

As a first experiment to build intuition, we computed the $P(N_m(k, w) > 0)$ for $m \in [1, 25]$ for k-mers, minstrobes, hybridstrobes, randstrobes, mixedstrobes and altstrobes. We chose $w = 64$ because it corresponds to the total span of the order 2 strobemer seeds parametrized with (2,15,25,50) in [37]. The result is illustrated in Figure 2A. If we sum over m as a proxy for sensitivity across mutation rates, we get the estimated summed sensitivity to 9.66, 10.1, 12.44, 12.74, 13.01, and 13.18 for minstrobes, k-mers, hybridstrobes, randstrobes, mixedstrobes (q=0.9), and altstrobes (k_s=10), respectively. We will, in later experiments, see that k-mers, randstrobes, mixedstrobes (q=0.9), and altstrobes (k_s=10) also are sorted in increasing entropy. As previously stated, our model cannot estimate entropy for minstrobes and hybridstrobes because of shared information. However, intuitively it makes sense for hybridstrobes to have higher entropy than k-mers but lower entropy than randstrobes, which is where it places in terms of sensitivity. Also, minstrobes perform worse than other strobemers because it has no sampling stochasticity (similar to k-mers). This result builds intuition and indicates that higher entropy implies higher sensitivity.

Next, we study how seed cover entropy is related to sensitivity at a more fine-grained level. Namely, we computed $H(\mathbf{X})$ (Eq. 5, ignoring the constant C) for k-mers and randstrobes as well as altstrobes and mixedstrobes for different parameter settings varying (k_s, k_l) and q, and evaluated them to empirical estimations of $\sum_{m=1}^{60} P(N_m(k, w) > 0)$ as a proxy for sensitivity (simulation details in Suppl. section S3).

Figure 2B shows $H(\mathbf{X})$ for the constructs when using the same parameters as in [37], namely k=30 for k-mers, and (2, 15, 25, 50) for randstrobes which gives valid $(30, 64)$-seeds. We consequently set (2, 15, 25, 50, q) with $q = 0, 0.1, \ldots, 1.0$ for mixedstrobes, and (2, k_s, k_l, 25, 50) with $k_s = 1, 2, \ldots, 14$, for altstrobes. Overall, we see that our model for approximating seed cover entropy (Fig. 2B) predicts roughly the relative sensitivity (Fig. 2C) both between parameters within a construct and between constructs.



(a) p_{match} (b) Entropy (c) Sensititvity

Fig. 2: Simulations showing how stochasticity in seed construct influence probability of w consecutive seeds producing at least one match in a region of length $2w = 128$ between sequences. Panel A shows $P(N_m(30, 64) > 0)$ for seed constructs k-mers (k=30), minstrobes, hybridstrobes, and randstrobes with (2,15,25,50), mixedstrobes (2,15,25,50,0.9), and altstrobes (2,10/20,25,50). Each $P(N_m(30, 64) > 0)$ estimate is derived from 100,000 instances of pairs of strings S and T. In general, a large gap is observed between non-stochastic constructs (k-mers, minstrobes) to constructs with stochasticity (hybridstrobes, randstrobes, mixedstrobes, altstrobes) for most mutation frequencies. The total sum of probabilities across m is higher for more stochastic constructs. Panel B shows $H(\mathbf{X})$ for k-mers, randstrobes, altstrobes (for different (k_s, k_l) and mixedstrobes (for different q). Panel C shows $P(N_m(30, 64) > 0)$ summed over $m \in [1, 60]$ for various window sizes $(w_{min}, 50)$, $(w_{min} \in 16, 25, 35, 45, 49)$.

Specifically, with our entropy model, we capture four trends. First, given a fixed w_{min}, we can typically predict which parameter settings of q and k_s are good for seed sensitivity for mixedstrobes and altstrobes, respectively. Second, our model immediately suggests that in most cases, narrower (w_{min}, w_{max}) leads to lower entropy (Fig. 2B), hence, lower sensitivity (Fig. 2C). For example, for mixedstrobes the optimal fraction of strobes decreases as w_{min} increases. This agrees with intuition, as the higher w_{min} and therefore less random the strobemer selection, the more we benefit from alternating short (k-mer) and long (randstrobe) seeds. A similar argument can be made for altstrobes, where more narrow (w_{min}, w_{max}) windows prompts larger difference in distance between k_s and k_l (Fig 2B). Third, we can compare k-mers, randstrobes, mixedstrobes, and altstrobes. We observe that for most w_{min}, typically altstrobes reach the highest peak entropy, second comes mixedstrobes, third randstrobes, and finally k-mers. This trend is also present in the sensitivity curves (Fig. 2B). Fourth, the relative increase in entropy typically correlates with the relative increase in sensitivity (besides the corner case of fixing the window size $(w_{min} = 49, w_{max} = 50)$, which we discuss in the limitations section). For example, compare the relative distances between entropy and sensitivity peaks for k-mers, randstrobes, altstrobe, and mixedstrobes. In Suppl. Fig. S1, We can see that such general trends also holds experiments for strobemers with $w_{max} = 100$ with varying w_{min}. Of interest, we observe that the theoretical maximum entropy, *i.e.*, uniformly sampling with probability k/w over each position, is 65 and 95 for the $(w_{max} = 50)$ and $(w_{max} = 100)$ experiments, respectively. This suggests that there are likely many better constructs to be found.

Model limitations and its mispredictions We learned that, under our formulation, entropy is not the only feature that predicts seed sensitivity. We believe two additional phenomenon to model is (P1) the probability that a contiguous segment (e.g., strobe or k-mer) is destroyed by mutations and (P2) the probability that a contiguous segment is selected differently between two sequences due to mutations in the sample window. An example of why P1 is needed is the following. For mixedstrobes, assume we would sample a k-mer or a strobemer based on the first k (approach 1) or $k/2$ (approach 2) nucleotides from the start position of the seed. It is straightforward to see that for any $m > 1$, there will be more shared $k/2$-mers than k-mers between the sequences. Hence, the probability of sampling the same seed, and therefore generating a match, is higher for approach 2 (which we implement). Similar argument can be made for altstrobes which is why we decide which length to sample based on the hash of k_s. Our model is agnostic to this phenomenon,

and it illustrates the importance of modelling P1. For P2 it is not clear to the authors the effect of the window size and strobe size combination. We use P1 and P2 to explain the three mispredictions we observe.

- Our model does not predict well the entropy for very narrow (w_{min}, w_{max}), see (w_{min}=49,w_{max}=50) (Fig 2B) and(w_{min}=99,w_{max}=100) (Suppl. Fig. S1). This misprediction is clearly explained by P1. For very narrow windows the strobemer is effectively a contiguous segment (a spaced k-mer) which is destroyed by an indel anywhere in the region, hence worse performing than k-mers.
- The peak altstrobe entropy fluctuates more with w_{min} than the resulting sensitivity. Specifically, the peak entropy goes from a k_s of 6 to 15 (becoming randstrobes) for w_{min} 45 to 16. However, the peak sensitivity only varies from 8 to 11 for the same range of values of w_{min} (Fig 2C and Suppl. Fig. S1). This misprediction can also be explained by P1. For altstrobes, a very short k_s implies a very long k_l (bad according to P1). On the other hand, a long k_s (or randstrobes) is also bad for matches between closely occurring mutations (Fig. 1A).
- While our model prediction for mixedstrobes is in general quite consistent with the sensitivity, the peak entropy becomes $q = 1.0$ (pure randstrobes) for w_{min}=16 while the peak sensitivity stays at 0.9 (Fig 2C and Suppl. Fig. S1). While we have not found a satisfying explanation for peak sensitivity never being pure randstrobes as our model predicts, it could be that increased chance of P2 happening out-weights the benefit of high entropy.

While modelling of P1 and P2 is probably needed for a more accurate prediction, we also observe that the entropy of a seed cover (specifically, randomness between seeds) is a key ingredient in any model predicting sensitivity. Merely modelling P1 and P2 does not capture the interplay between neighboring seeds. Other technical limitations to our model also exits. Firstly, Eq. 5 is an approximation of $H(\mathbf{X})$. Secondly, $H(\mathbf{X})$ assume a perfectly random hash function of linking. This is not true in practice, e.g., as is seen Suppl. Fig. S2, and technical limitations of uniform hashing for randstrobes has been identified [39].

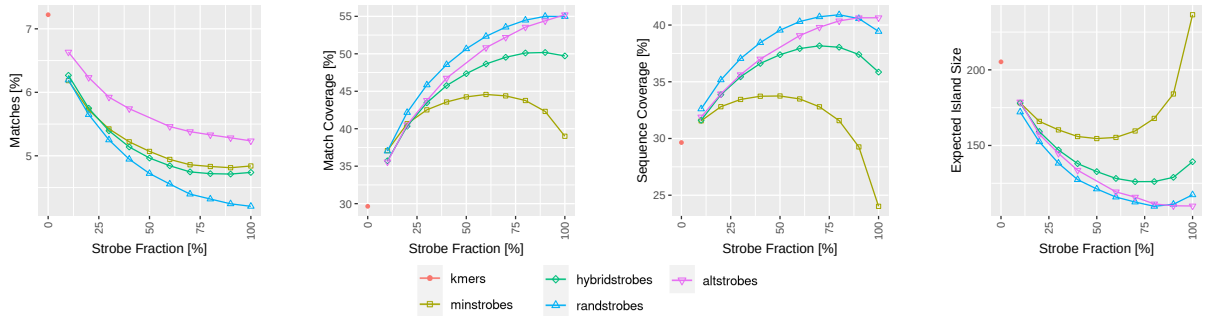


Fig. 3: Comparison between randstrobes, mixed-randstrobes and k-mers when mapping genomic Oxford Nanopore Technology (ONT) reads from *E.coli* to its reference. The *E.coli* reads were split up in long disjoint segments of 2,000nt. Next, the segments were seeded with strobemer fractions from 0% (k-mers) to 100% (randstrobes), downstream windows set to [25,50] and all strobes combined adding up to equal length subsequences of size 30 for better comparison. Then for each segment, the collinear solution of raw hits was computed to subsequently quantify number of matches, match coverage, sequence coverage and expected island size.

3.2 Sequence matching results

We took the best performing altstrobes ($k_s = 10$) and mixedstrobes ($q = 0.8$) according to sensitivity, and evaluated them against k-mers, and the other strobemers. We used the sequence match analysis performed in [37], where several different aspects of sequence matching performance were evaluated, using both the simulations and genomic Oxford nanopore reads from [37]. For details about the data and simulation setup, see Suppl. Section S4. For comparability with the strobemer study [37], we use the same parameters (*e.g.*, $k = 30$, and strobemers with $(2, 15, 25, 50)$). Altstrobes is seeded with $k_s = 10$ and $k_l = 2k_s = 20$ ($(2, 10/20, 25, 50)$), which were shown to provide best matching metrics (see Suppl. Fig. S3) and to ensure high uniqueness/entropy. We also compute altstrobes with various k-mer fractions (i.e., mixed-altstrobes) for formatting consistency with the other results.

Both the simulated (Suppl. Fig. S4, panels with two strobes) and biological (Fig. 3, Suppl. Fig. S5 and S6) experiments confirmed our simulated sensitivity analysis. Mixedstrobes with a strobemer fraction of

roughly 70-80% perform best. The fraction of matches is higher than when seeding only randstrobes while, at the same time, the sequence coverage and expected island size were better compared to randstrobes. Altstrobes outperform randstrobes, particularly on simulated data (Suppl. Fig. S4). On simulated data, we observe that altstrobes have the most desired performance (highest sequence coverage and smallest island size), followed by mixedstrobes at about 80% strobe fraction, agreeing with our sensitivity analysis. For the biological data (Fig. 3), mixedstrobes at 80% strobe fraction is comparable to altstrobes. We believe this is because biological errors are less uniform, which may be beneficial for k-mers.

To quantify the specificity of altstrobes and mixedstrobes, we computed the E-hits on human chromosome 21 to randstrobes and k-mers (Fig. 4). We observe that altstrobes and mixedstrobes have higher specificity (lower E-hits) than k-mers. Additionally, we compared the fraction of unique seeds for k-mers to altstrobes and other strobemer techniques of orders 2 and 3, yielding similar results (Suppl. Fig. S7).

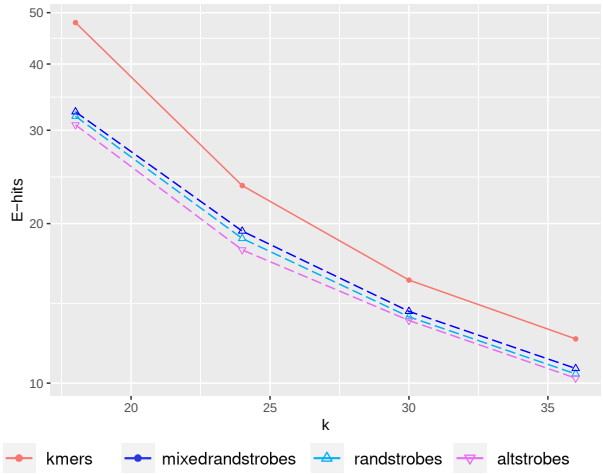


Fig. 4: Seed uniqueness as expected number of hits (E-hits) from a seed randomly drawn from Human chromosome 21. Chromosome 21 of the human GRCh38 assembly was seeded with k-mers, randstrobes $(2, k/2, 25, 50)$, mixedstrobes $(2, k/2, 25, 50, 80)$ and altstrobes $(2, (k/3)/(2k/3), 25, 50)$, whereby the number of extracted nucleotides ($k = 30$) was the same for all seeding techniques.

3.3 Time and memory to construct altstrobes and mixedstrobes

We implemented altstrobes and mixedstrobes in StrobeMap [37] in C++ (Suppl. Fig. S8). K-mers are 3.5 times and 2.5 times faster than altstrobes and mixedstrobes, respectively. However, the time difference becomes negligible when looking at the total indexing time (including, *e.g.*, sorting seeds and adding to hash table), especially when also taking into account that indexing is not the time limiting factor in most applications. The size on the index is nearly identical (Suppl. Fig. S8).

3.4 Minimap2 implementation

We implemented subsampled randstrobes, mixedstrobes, and altstrobe seeds in minimap2 [30] (see Suppl. section S7) to benchmark speed and accuracy of our seeding techniques and aligned simulated reads at various error rates to CHM13 (for details see Suppl. section S4). Our results indicate that altstrobes (2,9/18, 25,50) and mixedstrobes (2,14,25,50,80) have slightly more (0.2%) correctly mapped reads, and slightly faster alignment time (up to 30%) compared to k-mers ($k = 28$) with a similar number of extracted seeds and peak RAM usage (Suppl. Fig. S8). Altstrobes and mixedstrobes also speed up alignment up to 3.5 times compared to default setting ($k = 15$). However, the number of correctly mapped reads remains lower than default setting. This is expected as using much smaller seeds is beneficial for sensitivity, at the cost of computing time. Also, minimap2's search and extend parameters are highly optimized for exact k-mer seeds and minimizer subsampling, which is distorts the entropy of altstrobe and mixedstrobe seeds and is not something we modelled in our analysis.

3.5 Implementation and Software Availability

All the scripts used for the analysis and evaluation, as well as our seed implementations in StrobeMap and minimap2 are available at GitHub (https://github.com/benjamindominikmaier/mixedstrobes_altstrobes).

Data analysis was conducted in R 4.1.3 [35] and Python 3.10 [46]. Figures were produced using the package ggplot2 3.3.5.[48] in tidyverse 1.3.1 [49] (R); and matplotlib 3.5.1 [23] and seaborn 0.11.2 [47] (Python).

4 Discussion

To our knowledge, we believe that we have provided the first study on seed sensitivity analysis for fuzzy seeds that employ a stochastic sampling decision. We discovered a relationship between the stochasticity of a seed construct (through the entropy of a seed cover) and its effect on seed sensitivity, and we have experimentally verified that this relationship exists by modeling the entropy of a seed cover (Fig. 2, Suppl. Fig. S1). We have also discussed the cases where our model disagrees with our observations and explained why our model is incomplete. We have also expanded the strobemer family with mixedstrobes (combining k-mers and strobemers) and altstrobes (alternated strobe lengths). We experimentally verified that for most parameter settings where mixedstrobes and altstrobes have higher entropy than randstrobes (the previously best performing strobemer), they also produce higher seed sensitivity. We further validated the benefit of using mixedstrobes (80% randstrobe-fraction) and altstrobes as seeds using several metrics from [37] (Suppl. Fig. S4 and Fig. 3) and also showed that altstrobes have higher specificity than, e.g., randstrobes (Fig. 4 and Suppl. Fig. S7). We also showed that mixedstrobes and altstrobes are fast to construct (Suppl. Fig. S8-9) and do not constitute a bottleneck in mapping applications. Finally, we implemented randstrobes, mixedstrobes, and altstrobes in mininmap2 [30]. Minimap2 employs subsampling of seeds which distorts the relative entropies. Also, minimap2 implements chaining and other search-based cut-offs centered around minimizers. Nevertheless, we observed that using subsampled randstrobes and mixedstrobes within minimap2 for the most divergent sequence (10% mutation rates) both reduced runtime compared to k-mers of the same size with 25-30% and resulted in 0.2% more correctly mapped reads on CHM13 (Suppl. Fig. S9).

4.1 Future work

We believe that our work opens up for future work in several directions. Firstly, we believe that incorporating probabilities of error-free runs [3] will improve our model, which is currently only modeling entropy. Secondly, we showed that our investigated seed constructs have a large gap to maximum entropy, suggesting that there could be substantially more sensitive seed constructs. We may use our work's insights to produce even better seed constructs. Thirdly, subsampling is popular because of the reduced memory footprint and processing time, so we are interested in adapting our model to incorporate subsampling. It is clear that when using subsampling, the advantage that more random seed constructs have (e.g., strobemers) over k-mers reduces [37], as the high overlap of k-mers is removed. Nevertheless, it would, for example, be beneficial to understand what subsampling density is needed to make protocols similar in performance. Fourthly, since the minimap2 implementation is centered around minimizers, it is possible that aligners customized for, e.g., strobemers or other fuzzy seeds may enjoy an even more substantial performance gain, as shown for short-read alignment [38].

5 Conclusion

Our contributions are both theoretical and practical. Inspired by the many spaced seed studies on how to design optimal spaced seeds, we formalize and analyze more general fuzzy seeds, including, e.g., strobemers. To our knowledge, this is the first study on seed performance for fuzzy seeds that employ a stochastic sampling decision which makes them indel-tolerant. We discovered that seed sensitivity could be relatively well predicted by modeling the entropy of positions sampled by a set of neighboring seeds (denoted seed cover in this study). We discuss potential improvements to the model where it does not give accurate predictions. Furthermore, we provide two seed constructs, mixedstrobes, and randstrobes, that are fast to construct and practical to use and can, for some parametrizations, improve over randstrobes, the currently most sensitive strobemer seed.

Funding

Kristoffer Sahlin was supported by the Swedish Research Council (SRC, Vetenskapsrådet) under Grant No. 2021-04000.

References

1. Altschul, S.F., Gish, W., Miller, W., Myers, E.W., Lipman, D.J.: Basic local alignment search tool. Journal of Molecular Biology **215**(3), 403–410 (1990). https://doi.org/https://doi.org/10.1016/S0022-2836(05)80360-2, https://www.sciencedirect.com/science/article/pii/S0022283605803602
2. Au, K.F., Underwood, J.G., Lee, L., Wong, W.H.: Improving pacbio long read accuracy by short read alignment. PLOS ONE **7**(10), 1–8 (10 2012). https://doi.org/10.1371/journal.pone.0046679, https://doi.org/10.1371/journal.pone.0046679
3. Blanca, A., Harris, R.S., Koslicki, D., Medvedev, P.: The statistics of k-mers from a sequence undergoing a simple mutation process without spurious matches. Journal of Computational Biology **29**(2), 155–168 (2022). https://doi.org/10.1089/cmb.2021.0431, https://doi.org/10.1089/cmb.2021.0431, pMID: 35108101
4. Blassel, L., Medvedev, P., Chikhi, R.: Mapping-friendly sequence reductions: going beyond homopolymer compression. Cell iScience, in press (2022)
5. Bloom, B.H.: Space/time trade-offs in hash coding with allowable errors. Communications of the ACM **13**(7), 422–426 (1970)
6. Burkhardt, S., Crauser, A., Ferragina, P., Lenhof, H.P., Rivals, E., Vingron, M.: q-gram based database searching using a suffix array (quasar). In: Proceedings of the third annual international conference on Computational molecular biology. pp. 77–83 (1999)
7. Břinda, K., Sykulski, M., Kucherov, G.: Spaced seeds improve k-mer-based metagenomic classification. Bioinformatics **31**(22), 3584–3592 (07 2015). https://doi.org/10.1093/bioinformatics/btv419, https://doi.org/10.1093/bioinformatics/btv419
8. Charikar, M.S.: Similarity estimation techniques from rounding algorithms. In: Proceedings of the Thiry-Fourth Annual ACM Symposium on Theory of Computing. p. 380–388. STOC '02, Association for Computing Machinery, New York, NY, USA (2002). https://doi.org/10.1145/509907.509965, https://doi.org/10.1145/509907.509965
9. Chin, C.S., Khalak, A.: Human genome assembly in 100 minutes. bioRxiv (2019). https://doi.org/10.1101/705616, https://www.biorxiv.org/content/early/2019/07/17/705616
10. DeBlasio, D., Gbosibo, F., Kingsford, C., Marçais, G.: Practical universal k-mer sets for minimizer schemes. In: Proceedings of the 10th ACM International Conference on Bioinformatics, Computational Biology and Health Informatics. pp. 167–176 (2019)
11. Delcher, A.L., Kasif, S., Fleischmann, R.D., Peterson, J., White, O., Salzberg, S.L.: Alignment of whole genomes. Nucleic Acids Res **27**(11), 2369–2376 (Jun 1999)
12. Edgar, R.: Syncmers are more sensitive than minimizers for selecting conserved k-mers in biological sequences. PeerJ **9**, e10805–e10805 (Feb 2021). https://doi.org/10.7717/peerj.10805, https://pubmed.ncbi.nlm.nih.gov/33604186, 33604186[pmid]
13. Ekim, B., Berger, B., Orenstein, Y.: A randomized parallel algorithm for efficiently finding near-optimal universal hitting sets. In: Schwartz, R. (ed.) Research in Computational Molecular Biology. pp. 37–53. Springer International Publishing, Cham (2020)
14. Ekim, B., Berger, B., Chikhi, R.: Minimizer-space de bruijn graphs: Whole-genome assembly of long reads in minutes on a personal computer. Cell Syst **12**(10), 958–968 (Oct 2021)
15. Ferragina, P., Manzini, G.: Proceedings of the 41st annual symposium on foundations of computer science. focs'00 (2000)
16. Filion, G.J.: Analytic combinatorics for computing seeding probabilities. Algorithms **11**(1), 3 (2018)
17. Filion, G.J., Cortini, R., Zorita, E.: Calibrating seed-based heuristics to map short reads with sesame. Frontiers in Genetics **11** (2020). https://doi.org/10.3389/fgene.2020.00572, https://www.frontiersin.org/articles/10.3389/fgene.2020.00572
18. Firtina, C., Park, J., Alser, M., Kim, J.S., Cali, D.S., Shahroodi, T., Ghiasi, N.M., Singh, G., Kanellopoulos, K., Alkan, C., Mutlu, O.: Blend: A fast, memory-efficient, and accurate mechanism to find fuzzy seed matches (2021). https://doi.org/10.48550/ARXIV.2112.08687, https://arxiv.org/abs/2112.08687
19. Frith, M.C., Noé, L., Kucherov, G.: Minimally-overlapping words for sequence similarity search. Bioinformatics **36**(22-23), 5344–5350 (Dec 2020)
20. Frith, M.C., Shaw, J., Spouge, J.L.: How to optimally sample a sequence for rapid analysis. bioRxiv (2022). https://doi.org/10.1101/2022.08.18.504476, https://www.biorxiv.org/content/early/2022/08/22/2022.08.18.504476
21. Giladi, E., Healy, J., Myers, G., Hart, C., Kapranov, P., Lipson, D., Roels, S., Thayer, E., Letovsky, S.: Error tolerant indexing and alignment of short reads with covering template families. Journal of Computational Biology **17**(10), 1397–1411 (2010). https://doi.org/10.1089/cmb.2010.0005, https://doi.org/10.1089/cmb.2010.0005, pMID: 20937014
22. Gotea, V., Veeramachaneni, V., Makałowski, W.: Mastering seeds for genomic size nucleotide BLAST searches. Nucleic Acids Research **31**(23), 6935–6941 (12 2003)
23. Hunter, J.D.: Matplotlib: A 2d graphics environment. Computing in Science & Engineering **9**(3), 90–95 (2007). https://doi.org/10.1109/MCSE.2007.55

24. Ilie, L., Ilie, S.: Multiple spaced seeds for homology search. Bioinformatics **23**(22), 2969–2977 (09 2007). https://doi.org/10.1093/bioinformatics/btm422, https://doi.org/10.1093/bioinformatics/btm422
25. Jain, C., Rhie, A., Hansen, N.F., Koren, S., Phillippy, A.M.: Long-read mapping to repetitive reference sequences using winnowmap2. Nature Methods **19**(6), 705–710 (2022)
26. Keich, U., Li, M., Ma, B., Tromp, J.: On spaced seeds for similarity search. Discrete applied mathematics **138**(3), 253–263 (2004)
27. Kong, Y.: Generalized correlation functions and their applications in selection of optimal multiple spaced seeds for homology search. J Comput Biol **14**(2), 238–254 (Mar 2007)
28. Lederman, R.: A random-permutations-based approach to fast read alignment. BMC Bioinformatics **14**(5), S8 (2013). https://doi.org/10.1186/1471-2105-14-S5-S8, https://doi.org/10.1186/1471-2105-14-S5-S8
29. leonbloy: Entropy of a binary random vector with exactly k non-zeros. Mathematics Stack Exchange (2018), https://math.stackexchange.com/q/2682723, uRL:https://math.stackexchange.com/q/2682723 (version: 2018-03-09)
30. Li, H.: Minimap2: pairwise alignment for nucleotide sequences. Bioinformatics **34**(18), 3094–3100 (05 2018)
31. Ma, B., Tromp, J., Li, M.: PatternHunter: faster and more sensitive homology search . Bioinformatics **18**(3), 440–445 (03 2002). https://doi.org/10.1093/bioinformatics/18.3.440, https://doi.org/10.1093/bioinformatics/18.3.440
32. Marchet, C., Boucher, C., Puglisi, S.J., Medvedev, P., Salson, M., Chikhi, R.: Data structures based on k-mers for querying large collections of sequencing data sets. Genome research **31**(1), 1–12 (Jan 2021). https://doi.org/10.1101/gr.260604.119, https://pubmed.ncbi.nlm.nih.gov/33328168, 33328168[pmid]
33. Nip, K.M., Hafezqorani, S., Gagalova, K.K., Chiu, R., Yang, C., Warren, R.L., Birol, I.: Reference-free assembly of long-read transcriptome sequencing data with rna-bloom2. bioRxiv (2022). https://doi.org/10.1101/2022.08.07.503110, https://www.biorxiv.org/content/early/2022/08/07/2022.08.07.503110
34. Noé, L., Martin, D.E.K.: A coverage criterion for spaced seeds and its applications to support vector machine string kernels and k-mer distances. J Comput Biol **21**(12), 947–963 (Dec 2014)
35. R Core Team: R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria (2022), https://www.R-project.org/
36. Roberts, M., Hayes, W., Hunt, B.R., Mount, S.M., Yorke, J.A.: Reducing storage requirements for biological sequence comparison. Bioinformatics **20**(18), 3363–3369 (07 2004). https://doi.org/10.1093/bioinformatics/bth408, https://doi.org/10.1093/bioinformatics/bth408
37. Sahlin, K.: Effective sequence similarity detection with strobemers. Genome research **31**(11), 2080–2094 (Nov 2021). https://doi.org/10.1101/gr.275648.121, https://pubmed.ncbi.nlm.nih.gov/34667119, 34667119[pmid]
38. Sahlin, K.: Flexible seed size enables ultra-fast and accurate read alignment. bioRxiv (2022). https://doi.org/10.1101/2021.06.18.449070, https://www.biorxiv.org/content/early/2022/05/25/2021.06.18.449070
39. Sahlin, K.: "How to get the most random randstrobes in compiled languages?" (2022), https://github.com/ksahlin/strobemers/issues/8
40. Sahlin, K., Baudeau, T., Cazaux, B., Marchet, C.: A survey of mapping algorithms in the long-reads era. bioRxiv (2022). https://doi.org/10.1101/2022.05.21.492932, https://www.biorxiv.org/content/early/2022/07/20/2022.05.21.492932
41. Sahlin, K., Medvedev, P.: Error correction enables use of oxford nanopore technology for reference-free transcriptome analysis. Nature Communications **12**(1), 2 (2021)
42. Shannon, C.E.: A mathematical theory of communication. The Bell System Technical Journal **27**(3), 379–423 (1948). https://doi.org/10.1002/j.1538-7305.1948.tb01338.x
43. Shaw, J., Yu, Y.W.: Theory of local k-mer selection with applications to long-read alignment. Bioinformatics (11 2021)
44. Sović, I., Šikić, M., Wilm, A., Fenlon, S.N., Chen, S., Nagarajan, N.: Fast and sensitive mapping of nanopore sequencing reads with graphmap. Nature Communications **7**(1), 11307 (Apr 2016). https://doi.org/10.1038/ncomms11307, https://doi.org/10.1038/ncomms11307
45. Stefanov, V., Robin, S., Schbath, S.: Waiting times for clumps of patterns and for structured motifs in random sequences. Discrete Applied Mathematics **155**(6), 868–880 (2007). https://doi.org/https://doi.org/10.1016/j.dam.2005.07.016, https://www.sciencedirect.com/science/article/pii/S0166218X0600374X, computational Molecular Biology Series, Issue V
46. Van Rossum, G., Drake, F.L.: Python 3 Reference Manual. CreateSpace, Scotts Valley, CA (2009)
47. Waskom, M.L.: seaborn: statistical data visualization. Journal of Open Source Software **6**(60), 3021 (2021). https://doi.org/10.21105/joss.03021, https://doi.org/10.21105/joss.03021
48. Wickham, H.: ggplot2: Elegant Graphics for Data Analysis. Springer-Verlag New York (2016), https://ggplot2.tidyverse.org
49. Wickham, H., Averick, M., Bryan, J., Chang, W., McGowan, L.D., François, R., Grolemund, G., Hayes, A., Henry, L., Hester, J., Kuhn, M., Pedersen, T.L., Miller, E., Bache, S.M., Müller, K., Ooms, J., Robinson, D., Seidel, D.P., Spinu, V., Takahashi, K., Vaughan, D., Wilke, C., Woo, K., Yutani, H.: Welcome to the tidyverse. Journal of Open Source Software **4**(43), 1686 (2019). https://doi.org/10.21105/joss.01686

50. Xin, H., Shao, M., Kingsford, C.: Context-aware seeds for read mapping. Algorithms for Molecular Biology **15**(1), 10 (2020)
51. Zheng, H., Kingsford, C., Marçais, G.: Sequence-specific minimizers via polar sets. Bioinformatics **37**(Supplement$_1$), $i187 - -i195$(072021)