

Estimating seed sensitivity on homogeneous alignments

Gregory Kucherov
LORIA/INRIA-Lorraine,
615, rue du Jardin Botanique,
B.P. 101, 54602
Villers-lès-Nancy France,
Gregory.Kucherov@loria.fr

Laurent Noé
LORIA/INRIA-Lorraine,
615, rue du Jardin Botanique,
B.P. 101, 54602
Villers-lès-Nancy France,
Laurent.Noé@loria.fr

Yann Ponty
LRI, UMR CNRS 8623,
Bât 490 Université Paris-Sud
91405 Orsay, France,
Yann.Ponty@lri.fr

Abstract

We address the problem of estimating the sensitivity of seed-based similarity search algorithms. In contrast to approaches based on Markov models [18, 6, 3, 4, 10], we study the estimation based on homogeneous alignments. We describe an algorithm for counting and random generation of those alignments and an algorithm for exact computation of the sensitivity for a broad class of seed strategies. We provide experimental results demonstrating a bias introduced by ignoring the homogeneous condition.

1. Introduction

Comparing nucleic acid or protein sequences remains by far the most common bioinformatics application. The classical local alignment problem consists in computing most significant similarities between two sequences, or between a sequence and a database. The significance of an alignment is measured by a *score*, commonly defined using an additive principle by assigning a positive score to each matching character, and a negative score (penalty) to each mismatch and each contiguous gap.

Best-scoring local alignments can be computed by the well-known Smith-Waterman dynamic programming algorithm [23], however for large-scale sequence comparison this computation becomes too time-consuming. Several heuristic algorithms have been designed to speed up the computation of local alignments, at the price of possibly missing some alignments or computing their sub-optimal variants. BLAST [1] is the most prominent representative of this family, Fasta [17] is another example. Both these programs are based on the common principle: similarity regions are assumed to share one or several short fragments, called *seeds*, that are used to detect potential similarities. More recently, it has been understood that using non-contiguous (called also *spaced* or

gapped) seeds instead of contiguous substrings can considerably improve the sensitivity/selectivity trade-off. PatternHunter [18] was the first method that used carefully designed spaced seeds to improve the sensitivity of DNA local alignment. Spaced seeds have been also shown to improve the efficiency of lossless filtration for approximate pattern matching [20, 7]. Earlier, random spaced seeds were used in FLASH program [8] to cover sequence similarities, and the sensitivity of this approach was recently studied in [5]. For the last two years, spaced seeds received an increasing interest [6, 3, 4, 10, 9] and have been used in new local alignment tools [22, 19].

Coming back to the concept of alignment score, note that all heuristic algorithms typically try to output alignments with a score greater than some user-defined bound. However, computing *all* such alignments would be an ill-defined task. Usually, the alignments of interest are those which, on the one hand, do not contain in them big regions of negative score (in which case the alignment should probably be split into two or more alignments of higher score) and on the other hand, are not too short to be a part of a larger high-scoring similarity. This is captured by the Xdrop heuristics, a part of BLAST algorithm: once a seed has been identified, the Xdrop algorithm extends it in both directions into so-called High-scoring Segment Pair (HSP), as long as the “running score” does not decrease more than by a certain value. All other seed-based algorithms apply a similar approach in that they extend the found seed (or group of seeds) outside as far as the total score does not undergo a prohibitive drop.

The alignments found in such a way are formalized through the notion of *maximal scoring segment*¹ [21]. Consider a *gapless alignment* which can be naturally translated into a sequence of match/mismatch scores. We call this alignment (or sequence) *homogeneous* if it does not contain

¹ We use the term *segment* instead of *subsequence* [21] as the latter usually does not require the elements to be contiguous.

a proper contiguous subalignment (segment) whose score is greater than that of the whole alignment. Given an alignment (or sequence), a homogeneous subalignment is called a *maximal scoring subalignment (segment)* if it is extended to the right and left as far as the homogeneity property is verified. In other words, a maximal scoring segment is not included in a larger homogeneous segment.

Abstracting from possible gaps, alignments found by similarity search programs are maximal scoring segments. On the other hand, maximal scoring segments capture a biologically relevant notion of alignment: if an alignment is not homogeneous then it is likely to be a merge of two alignments that should be considered as distinct, and if an alignment is not maximal, it is likely to be a part of a larger interesting alignment.

In this context, the main motivation of this work can be summarized by the following claim: *Since homogeneous alignments are those which are really found and intended to be found by similarity search algorithms, then the efficiency of those algorithms has to be measured on homogeneous alignments rather than on arbitrary alignments.*

With this motivation in mind, we propose an approach to analyze the sensitivity of similarity search algorithms. Using this approach, we demonstrate that measuring the sensitivity of algorithms on general alignments instead of homogeneous alignments (as it is usually done) leads to biased results, more specifically to an underestimation of the sensitivity.

In this paper, we propose a dynamic programming algorithm to compute the sensitivity estimator (hit probability) with respect to homogeneous alignments. The algorithm, which is an extension of algorithms proposed in [15, 6, 3], works for a wide range of seed definitions (contiguous or spaced seeds, single- double- or multiple-seed approaches, etc) and therefore can guide the choice of the seed strategy. It is based on the enumeration of homogeneous alignments. On the other hand, the enumeration allows us to obtain an efficient random generation algorithm for homogeneous alignments. The latter, in turn, can be used for estimating the hit probability experimentally by testing the chosen seed criterion on a large number of random homogeneous alignments.

Finally, note that the homogeneous sequences have other applications in biological sequence analysis. Karlin, Altschul and other authors [14, 13] studied long homogeneous (and maximal scoring) segments in protein sequences, where each residue is assigned a score according to a certain scoring function. It has been demonstrated that those segments are often biologically significant, and may, for example, correspond to transmembrane domains [13]. Therefore, the methods proposed in our paper can potentially apply to other bioinformatics problems than the problem of measuring the sensitivity of local align-

ments programs considered here.

The paper is organized as follows. Section 2 presents algorithms, based on enumeration techniques, for the uniform random generation of homogeneous sequences. Section 3 describes an exact algorithm to compute the seed detection probability on homogeneous sequences. Section 4 is devoted to experiments, and demonstrates the bias induced by considering general sequences rather than homogeneous ones. Finally, Section 5 discusses possible extensions and directions for future work.

2. Enumeration and random generation of homogeneous sequences

Our main object of study is the *gapless alignment* of two DNA sequences. We represent it by a binary sequence $A = (b_1, \dots, b_n)$, $b_i \in \{0, 1\}$, where 1 stands for a match and 0 for a mismatch. We assume that each match is assigned a constant positive integer score s , and each mismatch a constant negative integer score (penalty) $-p$, regardless of the mismatching letters. This is the case for many nucleotide scoring systems, for example a popular BLAST default scoring system assigns 1 to each match and -3 to each mismatch.² An alternative representation of A is then the sequence of individual scores, called *score sequence*, $X_A = (x_1, \dots, x_n)$, $x_i \in \{s, -p\}$ and $x_i = -p + b_i(s + p)$, and the score of the whole alignment is $S(X) = \sum_{k=1}^n x_k$.

Definition 1. A binary sequence A (and the associated score sequence X_A) is called *homogeneous* if $S(X_A)$ is strictly greater than $S(X_A[i..j])$ for all proper segments $X_A[i..j] = (x_i, \dots, x_j)$ ($i > 1$ or $j < n$).

This section is devoted to the following question: How to uniformly generate random homogeneous sequences of a given length n , or of given length n and score S ? Here the uniformity condition means that each sequence of the considered class has the same probability to occur.

The interest of this question is twofold. First, being able to randomly generate homogeneous sequences would allow to measure the sensitivity of a similarity search algorithm in the experimental way, by testing it against a sample of randomly drawn homogeneous sequences. Second, as it is of-

² More accurate scoring systems assign different penalties to different mismatches. In particular, it is very useful to distinguish between transitions (substitutions $A \leftrightarrow G$ and $C \leftrightarrow T$) and transversions (other substitutions), since transition mutations occur with a greater relative frequency than transversions, particularly in coding sequences. This distinction is also useful in seed design, as it allows to consider transition-constrained mismatches, as done by BLASTZ [22] or YASS [19] for example. A further step would be to make finer partitions of all nucleotide pairs, depending on statistical properties of analyzed sequences. In our setting, this would imply the modeling of alignments by sequences over three or more letters. We will discuss this extension in Section 5.

ten the case for combinatorial objects, the question of random generation of homogeneous sequences is closely related to the question of their enumeration, and the latter will be used in the next section to obtain an exact algorithm for computing the sensitivity.

For a binary sequence $A = (b_1, \dots, b_n)$ and the associated score sequence $X_A = (x_1, \dots, x_n)$, consider the evolution of the prefix score $\sum_{i=1}^k x_i$ for $k = 1..n$. The evolution can be represented as a walk on \mathbb{Z}^2 starting from the origin $(0, 0)$ and evolving through two possible vectors $(1, s)$ and $(1, -p)$. The one-to-one relation between a binary se-

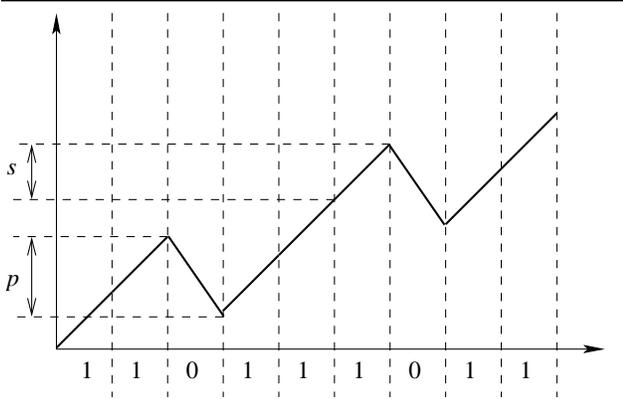


Figure 1. An alignment uniquely associated with a walk. 1 stands for a match and 0 for a mismatch

quence and a walk is illustrated in Figure 1.

Homogeneous sequences correspond to walks C on \mathbb{Z}^2 starting at $(0, 0)$, ending at (n, S) , and verifying two additional conditions:

- C is *positive*: $\forall (k, y) \in C, k > 0 \Rightarrow y > 0$,
- C is *culminating*: $\forall k < n, (k, y) \in C \Rightarrow y < S$.

A walk verifying both conditions is called a *culminating positive walk* (CPW). It is easily seen that the condition for a walk to be both positive and culminating is equivalent to the homogeneity of the underlying sequence.

We will be interested in two cases, depending on whether the total score (culminating point) S is fixed or not. We start with the case of sequences of arbitrary score S and then show how the algorithm is modified to the case of fixed score. Let C_n be the set of all CPWs of size n and arbitrary total score S .

A classical approach to the random generation of sequences of a given length drawn from a language L is based on counting suffixes of those sequences [24]. It allows to generate sequences incrementally from left to right. In our

case, this approach is preferable to the generation by rejection. The latter consists in generating sequences uniformly among all possible sequences and discarding those that do not meet the constraints. Although this method also yields a uniform distribution, and generating each candidate sequence can be done in linear time, the time complexity of the rejection method heavily depends on s and p parameters. It would be efficient if the constraints are not too strong – e.g. when the probability of rejection is $O(1/n)$. In our setting, if s is smaller than p (which is often the case in practice), the rejection probability tends to 1 with an exponential speed as n grows, thus leading to an expected exponential time complexity of generation.

The counting approach is based on the following general scheme. Let L be a set of sequences over an alphabet $\Sigma = \{a_1, \dots, a_m\}$ and L_n be the sequences of L of size n . Let w_p be a prefix of some sequence of L_n . We call $P_a(w_p)$ the probability that w_p can be followed by $a \in \Sigma$ to form a sequence of L_n :

$$P_a(w_p) = \frac{|\{w' | w_p a w' \in L_n\}|}{|\{w'' | w_p w'' \in L_n\}|}. \quad (1)$$

Lemma 2. Given values $P_a(w_p)$ for all $a \in A$ and all prefixes w_p , one can generate sequences of L_n uniformly.

Proof. Starting from ε , issue consecutively letters $\alpha_1, \dots, \alpha_n$ with probabilities $P_{\alpha_1}(\varepsilon), P_{\alpha_2}(\alpha_1), P_{\alpha_3}(\alpha_1 \alpha_2), \dots, P_{\alpha_n}(\alpha_1 \alpha_2 \dots \alpha_{n-1})$. The probability of issuing a sequence $w = \alpha_1 \alpha_2 \alpha_3 \dots \alpha_n$ is

$$\begin{aligned} P(\alpha_1 \dots \alpha_n) &= \\ &= P_{\alpha_1}(\varepsilon) P_{\alpha_2}(\alpha_1) P_{\alpha_3}(\alpha_1 \alpha_2) \dots P_{\alpha_n}(\alpha_1 \alpha_2 \dots \alpha_{n-1}) = \\ &= \frac{|\{w' | \alpha_1 w' \in L_n\}|}{|L_n|} \frac{|\{w' | \alpha_1 \alpha_2 w' \in L_n\}|}{|\{w' | \alpha_1 w' \in L_n\}|} \dots \frac{|\{w' | \alpha_1 \dots \alpha_n w' \in L_n\}|}{|\{w' | \alpha_1 \dots \alpha_{n-1} w' \in L_n\}|} = \\ &= \frac{1}{|L_n|} \end{aligned} \quad (2)$$

as $\{w' | \alpha_1 \dots \alpha_n w' \in L_n\} = \{\varepsilon\}$. Therefore, this yields a uniform generation procedure. \square

In general, one has to precompute up to $|\Sigma|^n$ values $P_\alpha(w_p)$. However, in the case of homogeneous sequences, it is not necessary to process each prefix w_p individually, as the only relevant information of w_p is the maximal ordinate h reached by w_p and the current ordinate y (see Figure 2). Therefore, we introduce the concept of (h, y) -initialized walk.

Definition 3. For $h, y \geq 0, y \leq h$, an (h, y) -initialized CPW is a CPW starting at $(0, y)$ and culminating at some point (n, S) , such that $S > h$.

Let $C_{y,h,n}$ be the set of (h, y) -initialized CPWs of length n .

Lemma 4. Assume that w_p is a positive walk from $(0, 0)$ to (k, y) and h is its maximal ordinate. Then $\{w' | w_p w' \in C_n\} = C_{y,h,n-|w_p|}$.

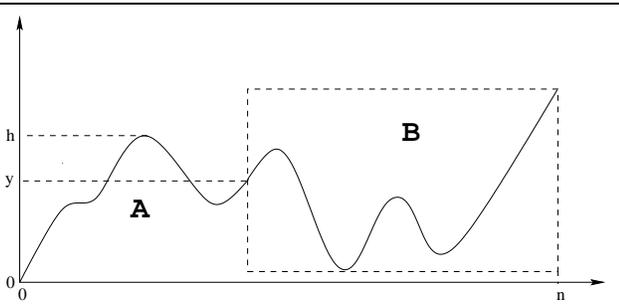


Figure 2. Suffixes B associated to a prefix A of C_n depend only of the maximal ordinate h and current ordinate y

A proof is immediate and is illustrated on Figure 2.

To count the (h, y) -initialized CPWs of size n for all compatible values of h, y and n , we use the following recursive decomposition of (h, y) -initialized CPWs. A CPW is represented below as a sequence of vectors $\{(1, s), (1, -p)\}$.

Lemma 5. For $y, h \geq 0$,

$$C_{y,h,1} = \begin{cases} (1, s) & \text{if } y + s > h \\ \emptyset & \text{otherwise} \end{cases} \quad (3)$$

and for $k > 1$,

$$C_{y,h,k} = \begin{cases} (1, s) \cdot C_{y+s, \max(h, y+s), k-1} \\ \cup (1, -p) \cdot C_{y-p, h, k-1} & \text{if } y > p, \\ (1, s) \cdot C_{y+s, \max(h, y+s), k-1} & \text{otherwise.} \end{cases} \quad (4)$$

Proof. A one-step CPW cannot be a $(1, -p)$ -step, as it would not be culminating. It can only be a $(1, s)$ -step provided that $h < y + s$.

The general case is a union of two cases, depending on whether the first step is $(1, s)$ or $(1, -p)$. The latter is possible only if $y > p$. If the first step is $(1, s)$, then the new maximum is set to $\max(h, y + s)$. \square

Note that the decomposition of $C_{y,h,n}$ is unambiguous, which means that the union operation is a disjoint union. Therefore, Lemma 5 gives a recursive formula for computing the number of CPWs $C_{y,h,k}$ for different values $1 \leq k \leq n$, $0 \leq y, h \leq s \cdot n$. A dynamic programming implementation of the recursion leads to an $O(n^3)$ time and space complexity.

Lemma 4 allows to count the number of possible suffixes for any prefix w_p of a walk of C_n . Let $k = n - |w_p|$, y be the ordinate of the final point of w_p and h the maximal ordinate reached by w_p . Then, the probability $P_1(w_p)$ that w_p is followed by a step $(1, s)$ is $|C_{y+s, \max(h, y+s), k-1}| / |C_{y,h,k}|$. As soon as the values $|C_{y,h,k}|$ are computed, a sequence of C_n is generated incrementally in time $O(n)$ using Lemma 2.

We now modify the method to generate homogeneous sequences of *fixed* score S , which amounts to generating CPWs with a fixed cumulating point. This case is simpler, as there is only a finite number of possible intermediate scores (states). Therefore the set of sequences becomes a regular language, for which there exists a linear-time random generation algorithm (including the preprocessing time) [12, 11]. In our case, it is sufficient to precompute an $S \times n$ table storing the number of CPW suffixes for each point of the rectangle specified by corner points $(0, 0)$ and (n, S) . Those can be seen as walks inside each rectangle with corners $(0, y)$ and (k, S) . Let $D_{y,k}^S$ be the set of such walks. The following lemma establishes the corresponding recurrence.

Lemma 6. For $y \geq 0$,

$$D_{y,1}^S = \begin{cases} (1, s) & \text{if } y + s = S \\ \emptyset & \text{otherwise} \end{cases} \quad (5)$$

and for $k > 1$,

$$D_{y,k}^S = \begin{cases} (1, s) \cdot D_{y+s, k-1}^S \\ \cup (1, -p) \cdot D_{y-p, k-1}^S & \text{if } p < y < S - s, \\ (1, s) \cdot D_{y+s, k-1}^S & \text{if } y \leq p < S - s \\ (1, -p) \cdot D_{y-p, k-1}^S & \text{if } p < S - s \leq y \\ \emptyset & \text{otherwise.} \end{cases} \quad (6)$$

Again, the union is disjoint, and therefore the recurrence can be used for counting the cardinality of each $D_{y,k}^S$. Using Lemmas 2 and 4, this gives a uniform generation algorithm of $O(S \cdot n)$ space and time complexity.

3. Computing the hit probability

We present now an algorithm for computing the hit probability on a random homogeneous sequence, that can be applied to different seed strategies such as single, double or multiple seeds, contiguous or spaced seeds, etc. The algorithm can be seen as an extension of the dynamic programming algorithm of [15] for computing the hit probability for a single seed, under the Bernoulli model of the sequence. The algorithm of [15] has been extended in several ways: [3] proposed an extension to the (Hidden) Markov Models of the sequence; another technique, proposed in [6], allows to deal with Markov Models of the sequence and with multiple seeds; finally, in [4], the algorithm of [15] was extended to another seed model, called *vector seeds*. A similar-style dynamic programming algorithm was proposed in [7] in a purely combinatorial setting, for computing the so-called *optimal threshold*, which is the minimal number of seed occurrences for given sequence length and number of substitution errors (see also [20]).

The extension we propose here is of different nature, as our probabilistic space is not specified by a probabilistic model of the alignment, but by a set of possible alignments and the condition of the uniform distribution. In other

words, here we impose *global constraints* on the alignments (to be homogeneous and to have a given score) rather than specifying their local properties, as Markov models do. The key of the construction is the representation from the previous section of those sequences as random culminating walks on the plane, together with counting formulas (5), (6).

For the sake of simplicity of presentation, we describe the algorithm for a single spaced seed. At the end of this section, we explain how the algorithm can be extended to multiple-seed strategies.

Recall that a seed π is a string over $\{0, 1\}$, where 1 stands for 'match' and 0 for a don't care symbol. The length l of π is called the *span* of π and the number of 1's in π its *weight*. A seed π *matches* a string $u \in \{0, 1\}$ of length l , if for each position i , $\pi[i] = 1$ implies $u[i] = 1$. A seed π *detects* a sequence (gapless alignment) $A \in \{0, 1\}^*$ if π matches some substring of A .

We now describe a dynamic programming algorithm for computing the exact probability that a given seed π of span l , detects a random homogeneous sequence A of length n and score S under a scoring system $(s, -p)$.

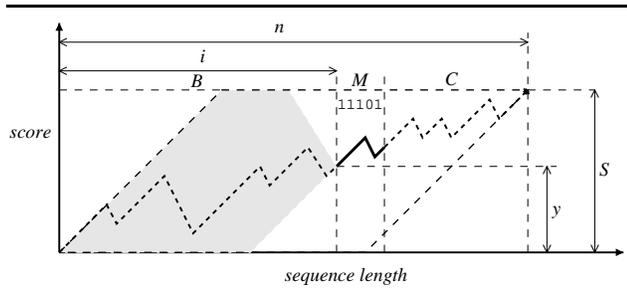


Figure 3. Computing the seed detection probability on homogeneous sequences

Consider a prefix of a random homogeneous sequence A and assume that this prefix ends with a suffix M . That is, let $A = BMC$ and let $|B| = i$ and $S(B) = y$. Let $P(i, M, y)$ be the probability that π detects the prefix BM of a random sequence A (see Figure 3). Thus, our goal is to compute the probability $P(n, \varepsilon, S)$ using a set of recursive equations that we define now. The following are initial conditions of the recursion:

- (i) $P(i, M, y) = 0$, if $i + |M| < l$,
- (ii) $P(i, M, y) = 1$, if $|M| = l$ and π detects M .

The following conditions insure that the probabilistic space is respected. Condition (iii) says that the sequences under consideration cannot have a negative score or a score greater than S . Condition (iv) is optional but allows to cut off at earlier stages some infertile branches of the computation. It in-

sure that the walks are inside the diagonal band defined by extremal points (see Figure 3).

- (iii) $P(i, M, y) = 0$, if $y \geq S$ and $i < n$, or $y \leq 0$ and $n > 0$,
- (iv) $P(i, M, y) = 0$, if $y > i \cdot s$ or $y < S - (n - i) \cdot s$.

The following conditions describe main recursion steps.

- (v) if π does not detect $1^{l-|Mb|}Mb$ ($b \in \{0, 1\}$), then $P(i, Mb, y) = P(i, M, y)$,
- (vi) if $|M| < l$, then $P(i, M, y) = P_1 P(i - 1, 1.M, y - s) + P_0 P(i - 1, 0.M, y + p)$, where P_1 and P_0 are computed using formulas (5), (6):

$$P_1 = \frac{|D_{S-y+s, i-1}^S|}{|D_{S-y, i}^S|}, \quad P_0 = \frac{|D_{S-y-p, i-1}^S|}{|D_{S-y, i}^S|} \quad (7)$$

Condition (v) says that if Mb is not a suffix of any match of π , then the last letter b can be dropped out. Condition (vi) is the most tricky one. It says that if M is shorter than l , then the probability is decomposed into the sum of two terms corresponding to two possible states of the walk right before the start of M (Figure 3). A way to compute the probability p_0 and p_1 of each of those states is to "flip over" the whole picture and to think of the walk as coming from point (n, S) to $(0, 0)$. Then, P_0 and P_1 are computed by formula (1) using the counting technique described in the previous section. The walks that contribute to probabilities P_0 and P_1 are those located inside the shadowed zone in Figure 3.

The recursive decomposition of $P(n, \varepsilon, S)$ goes as follows: by applying (vi), the size of M increases up to length l , then by alternating (vi) and (v), the size of M alternates between l and $(l - 1)$ while i decreases unless conditions (i)-(iv) apply.

The worst-case complexity of the algorithm is $O(2^l \cdot S \cdot n)$ both in time and space. The time complexity can be improved by introducing a preprocessing step and exploiting the structure of the seed π , following a general method described in [15, 3]. If w is the weight of the seed π , the time complexity can be made $O(l \cdot 2^{l-w} \cdot (l^2 + S \cdot n))$. We refer to [3] for details.

The algorithm presented above can be extended to certain multi-seed detection strategies, when a *hit* is defined as two or more proximate occurrences of the seed. A multi-seed strategy is used in Gapped BLAST [2] (two non-overlapping seed occurrences), BLAT [16] (two or more non-overlapping occurrences), PatternHunter [18] (two possibly overlapping occurrences), YASS [19] (any number of possibly overlapping occurrences, with additional restriction on the overlap size).

To extend the algorithm to the case of K seeds without constraints on the overlap, it is sufficient to perform the

recursion on the probability $P(i, M, y, k)$, where the additional parameter k , $0 \leq k \leq K$, means that k distinct occurrences of the seed are assumed to occur in BM (see Figure 3). The modification will mainly concern relation (ii), which will read as $P(i, M, y, k) = P(i, M^-, y, k - 1)$, where M^- means word M without the rightmost letter. If the overlap between two successive seeds is upper-bounded by some constant Δ (possibly zero, in which case no overlap is allowed), the modification still holds, except that M^- should be set to M without Δ rightmost letters. If the detection strategy imposes an upper bound on the distance between two neighboring seeds, the recursion gets more complex, as yet another parameter should be introduced to “store” the distance between the closed seed on the left.

4. Experimental results

To demonstrate the bias introduced by ignoring the property of homogeneity, we compared the detection probabilities of different seed strategies on homogeneous vs non-constrained alignments of a given score and different lengths. The probabilities for homogeneous alignments were computed by the algorithm of Section 3. For general alignments, a simpler version of the algorithm was used, that does not account for the homogeneity constraint (details are left out).

Figure 4 shows the results. Each plot represents the probability of a certain seed to detect homogeneous/arbitrary alignments of a certain score as a function of their length. All experiments reported in this section use the default BLAST +1/-3 scoring system. Left and right plots correspond to score 16 and 32 respectively. The upper row corresponds to the seed 110100110010101111 of weight 11 and span 18 (implemented in PatternHunter [18]), while the lower row corresponds to the contiguous seed of weight 11 (implemented in BLAST 1 [1]).

For all settings, the results clearly show that ignoring the condition of homogeneity leads to a considerable underestimation of the sensitivity. The fraction of homogeneous alignments missed is, in most cases, at least two times less than what one would expect out of measurements on arbitrary alignments.

One of the most important applications of measuring the sensitivity is the design of optimal spaced seeds for the detection of alignments of a given type. Therefore, we made another group of experiments aiming at comparing the most sensitive seeds for homogeneous vs arbitrary alignments. Some results are shown in Table 1.

We computed *optimal seeds* for detecting homogeneous and arbitrary alignments of length 40, for several score values (between 12 and 24). Some results are shown in Table 1. They have been obtained by an exhaustive search through all seeds of span up to 20. The probabilities were computed

by the algorithm of Section 3. The table shows that for the same parameters (alignment score and seed weight), the optimal seeds are different, depending on whether the optimality is defined with respect to all alignments (probability P_a) or only homogeneous ones (probability P_h). In each case, the highest probability is shown in slanted characters.

(n, S)	w	$\pi_w^{\{h a\}}(n, S)$	P_h	P_a
(40, 12)	9	1110010110111	0.986271	0.902372
(40, 12)	9	111001001010111	0.983516	0.917869
(40, 16)	9	1110010110111	0.998399	0.988887
(40, 16)	9	1100110101111	0.998353	0.989535
(40, 16)	10	11101100101111	0.98742	0.938499
(40, 16)	10	110110010101111	0.98740	0.942769
(40, 20)	10	11101001110111	0.999172	0.996303
(40, 20)	10	110110010101111	0.999065	0.996555
(40, 20)	11	111011101001111	0.975462	0.993076
(40, 24)	11	111010011110111	0.999891	0.999661

Table 1. Optimal seeds for homogeneous vs arbitrary alignments

Furthermore, we have performed a similarity search based on seeds from Table 1, using YASS software [19]. In particular, we compared the number of alignments found using the seed optimized on homogeneous alignments vs the one for arbitrary alignments. Table 2 shows the results for the seeds from Table 1 of weight 9 and 10 ($\pi_w^h(n, S)$, respectively $\pi_w^a(n, S)$, stands for the optimal seed from Table 1 computed on homogeneous and all alignments respectively). The experiments were made on comparing full chromosomes IV (1560kb), V (580kb), IX (450kb), XVI (960kb) of *Saccharomyces Cerevisiae* against each other or against themselves. Both strands of each chromosome has been processed in each experiment (-r 2 option of YASS). The search was done with the “group size” parameter 10 and 11 for seed weight respectively 9 and 10 (option -s of YASS). The results show that, in most cases, the seed tuned for homogeneous alignments allows to identify more relevant similarities than the seed optimized for all alignments.

5. Discussion

In this paper we presented an approach to measure the sensitivity of seed-based similarity search strategies. The main point is to compute the hit probability over homogeneous alignments, rather than arbitrary alignments. The property of homogeneity requires that the alignment does not contain significant negative-score segments occurring either inside the alignment (in which case the alignment can be decomposed into alignments of higher score), or at

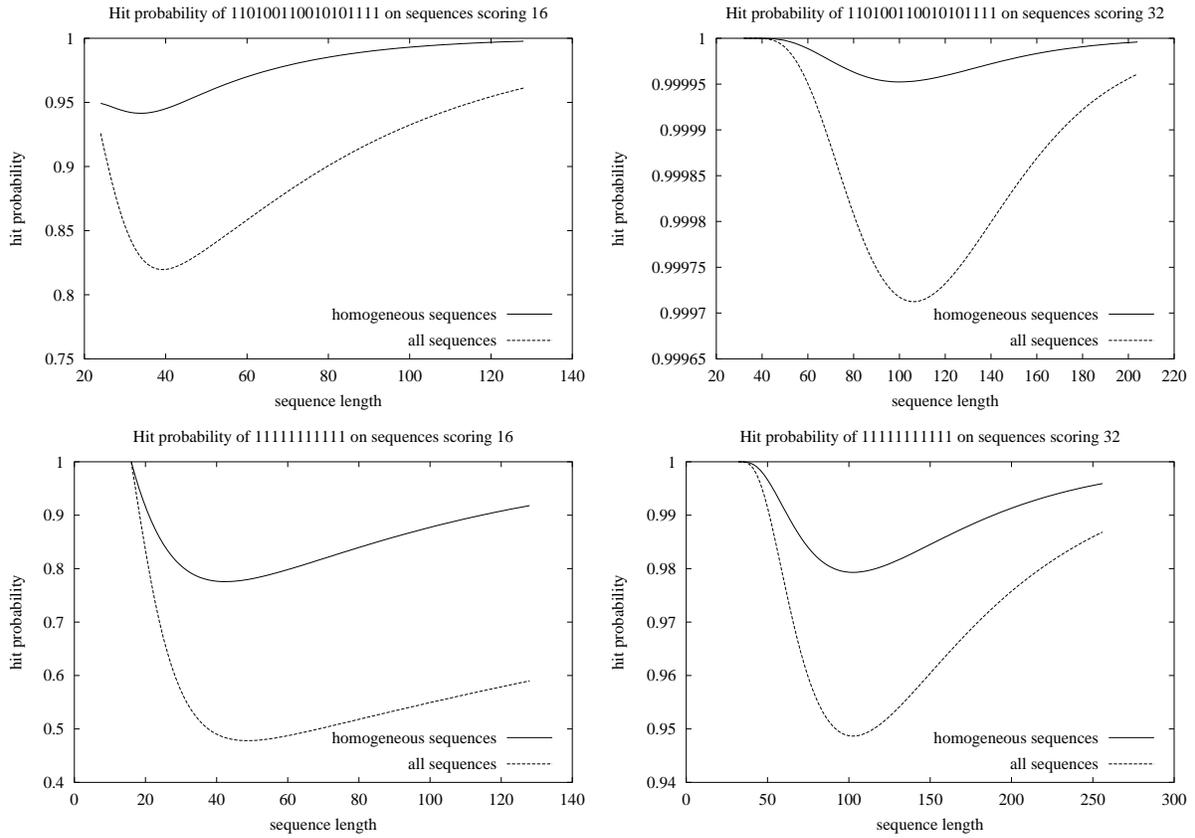


Figure 4. Seed detection probability on homogeneous vs arbitrary alignments

sequences	weight 9		weight 10	
	$\pi_9^h(40,12)$	$\pi_9^a(40,12)$	$\pi_{10}^h(40,20)$	$\pi_{10}^a(40,20)$
IX	519	519	502	496
IX / V	364	356	342	329
IX / XVI	408	387	383	348
IX / IV	523	521	488	473
V	500	487	477	466
V / XVI	961	955	937	891
V / IV	1273	1258	1248	1192
XVI	539	554	545	510
XVI / IV	1429	1448	1452	1368
IV	1542	1539	1510	1461

Table 2. Number of high-scoring alignments found with a seed optimized for homogeneous alignments (left column) vs that optimized for all alignments (right column)

the edges (in which case a subalignment should be considered). In this paper, we showed that ignoring this property leads to a bias in estimating the detection capacity of seeds.

Recently proposed approaches to estimate the seed de-

tection probability [6, 3, 4] assume a Markov model of alignment, that specifies its *local* composition. The approach of this paper is complementary, as we only impose global constraints (homogeneity, total score) and abstract from local properties of the alignment. If we want to account for local properties, the assumption that all fixed-score homogeneous sequences are equiprobable would be no longer justified.

Note that one of the drawbacks of the Markov model approach of [6, 3, 4] is that the alignment score is taken into account only indirectly, through the expected composition of the alignment (or *identity rate*, in case of the match/mismatch model). This is a serious disadvantage if one has to measure the probability on alignments of different length (as in [19]), since in this case the same score generally corresponds to different identity rates. The approach proposed in this paper is based on the score rather than on the identity rate.

Our analysis has been based on the match/mismatch model of alignment. However, sometimes it is very useful to distinguish between different mismatches, for example between transitions and transversions (see the footnote in Section 2). This approach leads to modeling alignments by se-

quences on non-binary alphabets (e.g. a three-letter alphabet of match/transition/transversion). From a pure computer science point of view, the results of Sections 2,3 can be extended to the non-binary alphabet. However, from the biological point of view, the assumption of the uniform distribution over all sequences becomes unrealistic in this setting, as letters are obviously no more “equivalent”, and properties of alignment composition must be added to the model. To conclude, the ultimate approach should take into account both global properties of the alignment and its local and compositional properties. Designing such an approach remains a challenging problem.

Acknowledgments We are grateful to Alain Denise for his help. We also thank Mikhail Roytberg for comments and helpful discussions. G. Kucherov and L. Noé have been supported by the french *Actions Spécifiques “Algorithmes et Séquences”* and *“Indexation de texte et découverte de motifs”* of CNRS. Yann Ponty has been supported by the french IMPG group *“Algorithmique, combinatoire et statistiques des séquences génomiques”*

References

- [1] S. Altschul, W. Gish, W. Miller, E. Myers, and D. Lipman. Basic Local Alignment Search Tool. *Journal of Molecular Biology*, 215:403–410, 1990.
- [2] S. Altschul, T. Madden, A. Schäffer, J. Zhang, Z. Zhang, W. Miller, and D. Lipman. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Research*, 25(17):3389–3402, 1997.
- [3] B. Brejova, D. Brown, and T. Vinar. Optimal spaced seeds for Hidden Markov Models, with application to homologous coding regions. In M. C. R. Baeza-Yates, E. Chavez, editor, *Proceedings of the 14th Symposium on Combinatorial Pattern Matching, Morelia (Mexico)*, volume 2676 of *Lecture Notes in Computer Science*, pages 42–54. Springer, June 2003.
- [4] B. Brejova, D. Brown, and T. Vinar. Vector seeds: an extension to spaced seeds allows substantial improvements in sensitivity and specificity. In R. P. G. Benson, editor, *Proceedings of the 3rd International Workshop in Algorithms in Bioinformatics (WABI), Budapest (Hungary)*, volume 2812 of *Lecture Notes in Computer Science*. Springer, September 2003.
- [5] J. Buhler. Efficient large-scale sequence comparison by locality-sensitive hashing. *Bioinformatics*, 17(5):419–428, 2001.
- [6] J. Buhler, U. Keich, and Y. Sun. Designing seeds for similarity search in genomic DNA. In *Proceedings of the 7th Annual International Conference on Computational Molecular Biology (RECOMB03), Berlin (Germany)*, pages 67–75. ACM Press, April 2003.
- [7] S. Burkhardt and J. Kärkkäinen. Better filtering with gapped q -grams. *Fundamenta Informaticae*, 56(1-2):51–70, 2003. Preliminary version in *Combinatorial Pattern Matching 2001*.
- [8] A. Califano and I. Rigoutsos. Flash: A fast look-up algorithm for string homology. In *Proceedings of the 1st International Conference on Intelligent Systems for Molecular Biology*, pages 56–64, July 1993.
- [9] K. Choi, F. Zeng, and L. Zhang. Good spaced seeds for homology search. *Bioinformatics*, 2003. accepted.
- [10] K. Choi and L. Zhang. Sensitivity analysis and efficient method for identifying optimal spaced seeds. *Journal of Computer and System Sciences*, 2003. to appear.
- [11] A. Denise. Génération aléatoire et uniforme de mots de langages rationnels [Uniform random generation of words of regular languages]. *Theoretical Computer Science*, 159(1):43–63, 1996.
- [12] T. Hickey and J. Cohen. Uniform random generation of strings in a context-free language. *SIAM. J. Comput.*, 12(4):645–655, 1983.
- [13] S. Karlin and S. Altschul. Applications and statistics for multiple high-scoring segments in molecular sequences. *Proc Natl Acad Sci USA*, 90(12):5873–7, June 15 1993.
- [14] S. Karlin and V. Brendel. Chance and significance in protein and DNA sequence analysis. *Science*, 257:39–49, 1992.
- [15] U. Keich, M. Li, B. Ma, and J. Tromp. On spaced seeds for similarity search. to appear in *Discrete Applied Mathematics*, 2002.
- [16] W. J. Kent. BLAT—the BLAST-like alignment tool. *Genome Research*, 12:656–664, 2002.
- [17] D. Lipman and W. Pearson. Improved tools for biological sequence comparison. *Proc. Natl. Acad. Sci. USA*, 85:2444–2448, 1988.
- [18] B. Ma, J. Tromp, and M. Li. PatternHunter: Faster and more sensitive homology search. *Bioinformatics*, 18(3):440–445, 2002.
- [19] L. Noé and G. Kucherov. YASS: Similarity search in DNA sequences. Technical Report RR-4852, INRIA, juin 2003. <http://www.inria.fr/rrrt/rr-4852.html>.
- [20] P. Pevzner and M. Waterman. Multiple filtration and approximate pattern matching. *Algorithmica*, 13:135–154, 1995. Preliminary version in *Combinatorial Pattern Matching 1993*.
- [21] W. L. Ruzzo and M. Tompa. A linear time algorithm for finding all maximal scoring subsequences. In *Seventh International Conference on Intelligent Systems for Molecular Biology*, pages 234–241, Heidelberg, Germany, 1999.
- [22] S. Schwartz, J. Kent, A. Smit, Z. Zhang, R. Baertsch, R. Hardison, D. Haussler, and W. Miller. Human–mouse alignments with BLASTZ. *Genome Research*, 13:103–107, 2003.
- [23] T. Smith and M. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147(195–197), Mar 1981.
- [24] H. S. Wilf. A unified setting for sequencing, ranking, and selection algorithms for combinatorial objects. *Advances in Mathematics*, 24:281–291, 1977.