

A General Framework for Local Pairwise Alignment Statistics with Gaps

Pasi Rastas

Department of Computer Science & HIIT Basic Research Unit
P.O. Box 68 (Gustaf Hällströmin katu 2b)
FIN-00014 University of Helsinki, Finland
`firstname.lastname@cs.helsinki.fi`

Abstract. We present a novel dynamic programming framework that allows one to compute tight upper bounds for the p -values of gapped local alignments in pseudo-polynomial time. Our algorithms are fast and simple and unlike most earlier solutions, require no curve fitting by sampling. Moreover, our new methods do not suffer from the so-called edge effects, a by-product of the common practice used to compute p -values. These new methods also provide a way to get into very small p -values, that are needed when comparing sequences against large databases. Based on our experiments, accurate estimates of small p -values are difficult to get by curve fitting.

1 Introduction

The most basic and fundamental sequence analysis task is to find out if two sequences are related [1]. This can be accomplished by aligning sequences, to get the alignment score as a sequence similarity measure. Based on the score, it is decided whether the gained alignment occurs because of the sequences are related or just by chance. The most important similarity measure of two sequences is their best local alignment score [2], used for example in the BLAST software [3, 4]. The score of a local alignment is the sum of the scores of aligned characters subtracted by the gap penalties. Local alignment is used routinely to compare new DNA and protein sequences against large sequence databases.

In this paper, a dynamic programming framework is sketched that allows one to compute tight upper bounds for the p -values of gapped local alignment scores. The gap models include linear, affine, and constant penalty models. Our algorithms use dynamic programming and unlike most earlier solutions, require no curve fitting. Moreover, our new methods do not suffer from the so-called edge effects, a by-product of the common practice to reduce the two dimensional alignment problem to a one-dimensional case [5]. Our methods provide a way to get into any meaningful p -value, including very small ones needed when comparing sequences against large databases. Based on our experiments, accurate estimates of small p -values are difficult to get by curve fitting. We present the framework for the simple Bernoulli distribution, but so other distributions would be possible as well. For example, one of the strings could be fixed and only the

other could be drawn from the null distribution. Also a fixed order Markov chain could be used as the null model.

The organization of this paper is following. Section 2 gives some basic definitions of local alignments and Section 3 recalls the problem of statistical significance of alignment scores. Section 3 also reviews of some previous methods to compute the significance of local alignment scores. In Section 4 we present our framework for the computation of gapped alignment statistics. The computational hardness is considered in Section 5 followed by experimental results in Section 6. Last, the discussion and future work part concludes this paper in Section 7.

2 Definition

Let $x = x_1, \dots, x_n$ and $y = y_1, \dots, y_m$ be two strings of length n and m from a finite alphabet Σ . Let $s(a, b)$ be the score of aligning, or matching residues or letters a and b from the alphabet. Furthermore, let d be the linear gap penalty, i.e. the penalty of deleting or inserting a single character.

A local alignment of x and y can be described as a path in the alignment grid (i, j) , $i = 0, \dots, n$, $j = 0, \dots, m$, taking vertical, diagonal or horizontal steps. A diagonal step from (i, j) to $(i + 1, j + 1)$ corresponds to aligning residues x_{i+1} and y_{j+1} and has score $s(x_{i+1}, y_{j+1})$, a horizontal step from (i, j) to $(i + 1, j)$ deletes character x_{i+1} and has score $-d$, and a vertical step from (i, j) to $(i, j + 1)$ inserts character y_{j+1} and has score $-d$. The total score of the alignment given by the path is the sum of the scores of individual steps.

As an example, consider the alignment of $x = ACGT$ and $y = ACCT$:

x: AC-G
y: ACCT

The score of this alignment is $s(A, A) + s(C, C) - d + s(G, T)$. The path taken in this example is $(0, 0) \mapsto (1, 1) \mapsto (2, 2) \mapsto (2, 3) \mapsto (3, 4)$.

Let z be a path in the alignment grid from (i', j') to (i, j) ($0 \leq i' \leq i \leq n$ and $0 \leq j' \leq j \leq m$). Denote by $\text{score}(x, y, z)$ the score of x and y locally aligned along path z . The best local alignment score of x and y , $\text{score}(x, y)$, is defined as $\max_z \text{score}(x, y, z)$. The value of $\text{score}(x, y)$ can be computed by Smith–Waterman algorithm in time $O(mn)$ [2, 6, 1].

3 Statistical Significance of a Local Alignment

Having computed that the best (local) alignment score of some x and y is t , a natural question to ask is how significant the alignment is. To answer this, we use the classical statistical framework and compute a p -value of getting a alignment score as good or better than t , given the null hypothesis or the null model.

The null model used for sequence x (and for y) from the alphabet Σ is the random (Bernoulli) model, i.e. the probability of x in the null model is

$P(x) = \prod_{i=1}^n q_{x_i}$, where q_a is the (arbitrary) probability of an alphabet symbol a . The model fixes probabilities only for fixed length sequences, i.e. the length distribution of x (nor y) is not modeled.

Let D_t consist of fixed length string pairs (x, y) with best score t or better, i.e. $D_t = \{(x, y) \mid \text{score}(x, y) \geq t\}$. The significance measure, i.e. the p -value of alignment score t , is $p_t = \sum_{(x, y) \in D_t} P(x)P(y)$.

3.1 Significance of Alignments without Gaps

The significance of alignments without gaps is typically solved approximately as a one-dimensional problem. A single string of length mn is created from the two dimensional alignment problem. Strings x and y are aligned (globally) in all $m + n - 1$ different ways and resulting string pairs are put together to form a string $Y = Y_1, \dots, Y_{mn}$ of mn letter pairs ($Y_i \in \Sigma \times \Sigma$). Then a probability $q_a q_b$ is assigned for each letter pair (a, b) . This dimension reduction can also be thought as concatenating diagonals of the alignment grid (but skipping first row and column) to get a single linear chain.

The best one-dimensional local alignment score is the maximal segment score

$$M(Y) = \max_{1 \leq i \leq j \leq mn} \sum_{k=i}^j s(Y_k). \quad (1)$$

This dimension reduction does not give an exact solution because of the so-called edge effects. Some alignments can overlap the concatenation points, i.e. these alignments do not match to any real alignments. Moreover, the mn letter pairs of Y are not independent, as they depend on the two underlying strings x and y . In fact, if a string of mn letter pairs is positionally independent, there are Σ^{2mn} different strings of independent mn pairs of letters, while there are only Σ^{m+n} different Y as the number of Y 's equals the number of different string pairs (x, y) .

Karlin-Altschul Statistics. The most common way to approximate the distribution of $M(Y)$ for Bernoulli distributed x and y , used in BLAST as well [3], is to use the Karlin-Altschul statistics [5, 7]. Assuming that expected score $\sum_{a,b} q_a q_b s(a, b)$ is negative and there is some letter pair (a, b) with $s(a, b) > 0$, and m and n are about equal size, the p -value for the alignment can be approximated as

$$P(M(Y) > t) \approx 1 - \exp(-K m n e^{-\lambda t}), \quad (2)$$

for some K and λ [8]. In the gapless case, parameters K and λ can be solved analytically [5, 8]. The term $K m n e^{-\lambda t}$ is the E -value or E -score of BLAST [3]. Equation (2) can also be used for alignments with gaps [4].

Exact One-Dimensional Solution. In [9], a pseudo-polynomial time algorithm is given for evaluating the probability of $P(M(Y) \geq t)$ exactly [9]. The method is quite simple and uses a Markov chain illustrated in Figure 1. In the

Markov chain, there is a state for each score value $0, 1, \dots, t$ (possible scores must be integers). The transition probability from state i to state j is $P(s(a, b) = j - i)$, if $i, j = 1, \dots, t - 1$. Transitions from and to states 0 and t are handled as shown in Figure 1. After taking nm steps in the Markov chain, the probability of being in the state t is $P(M(Y) \geq t)$.

A straightforward implementation of this Markov chain approach runs in time $O(mnt) = O(n^3)$ (assuming the absolute values of $s(\cdot, \cdot)$ are bounded by a constant). The algorithm of [9] runs in time $O(t^{2.376} \log(mn))$.

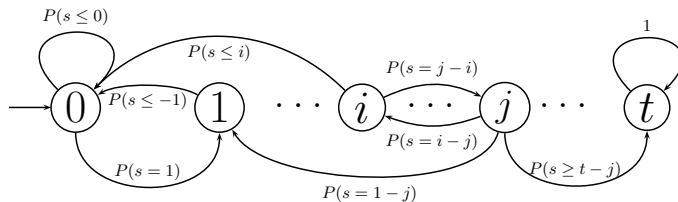


Fig. 1. A Markov chain used in [9] to compute $P(M(Y) \geq t)$. The number on each state denotes the alignment score. For clarity, all transitions have not been drawn.

3.2 Gapped Significance

As already mentioned, Formula (2) is also used for alignments with gaps. Many experimental studies show that the distribution of gapped alignment scores follow equation (2) reasonable accurately, e.g. [10]. However, to our best knowledge, there is no analytic, general way to solve K and λ with gaps. Some steps have been made towards analytic solution for these parameters [11].

4 Dynamic Programming Framework for Gapped Significance

In this section, we present our general framework, which allows us to compute expected number of alignments with various types of distributions for the sequences. This expectation is an upper bound for the p -value when the sequences are distributed according to the null model.

Let X_t be a random variable counting the number of alignments of x and y with score $\geq t$, when x and y are distributed according to the null model. The p -value p_t is $P(X_t \geq 1)$. The expectation of X_t , $E(X_t)$, can be computed by summing over all possible alignment paths z and adding up the probability that the alignment score of x and y is $\geq t$ along each z . To justify this, consider a set A_t^z of string pairs (x, y) with best score t or better on a given local alignment z , i.e. $A_t^z = \{(x, y) \mid \text{score}(x, y, z) \geq t\}$. Now the random variable X_t can be written as a sum of indicator functions of A_t^z , i.e. $X_t = \sum_z 1_{A_t^z}$. As the expectation is linear, $E(X_t) = E(\sum_z 1_{A_t^z}) = \sum_z E(1_{A_t^z}) = \sum_z P(A_t^z)$.

We can compute $E(X_t)$ by dynamic programming. Let $L'(i, j, s)$ be the expected number of paths with score s and ending at (i, j) for a sequence pair with lengths i and j . Consider the linear gap model and assume that we somehow

know the values of $L'(i-1, j, \cdot)$, $L'(i, j-1, \cdot)$, and $L'(i-1, j-1, \cdot)$. How could we then compute $L'(i, j, s)$? It is not hard to see that

$$L'(i, j, s) = L'(i, j-1, s+d) + L'(i-1, j, s+d) + \sum_{a,b} q_a q_b L'(i-1, j-1, s-s(a, b)).$$

This gives our framework to work on. We can utilize this idea to compute value $L(i, j, s) = L'(i, j, s) - L'(i, j, s+1)$, by dynamic programming as follows (the unassigned values are zeros):

$$\begin{aligned} L(0, 0, 0) &= L(0, j, 0) = L(i, 0, 0) = L(i, j, 0) = 1 \\ L(i, j, s) &= L(i-1, j, s+d) + L(i, j-1, s+d) \\ &+ \sum_{a,b \in \Sigma} q_a q_b L(i-1, j-1, s-s(a, b)), \end{aligned} \quad (3)$$

where $i = 1, \dots, n$, $j = 1, \dots, m$, and $s = 1, \dots, (n \max_{a,b} s(a, b))$. Then the expectation $E(X_t)$ can be computed as $E(X_t) = \sum_{i,j} \sum_{s \geq t} L(i, j, s)$. We have obtained an upper bound for the p -value, as by using Markov inequality $p_t = P(X_t \geq 1) \leq E(X_t)$.

This framework is a generalization of the one in [12] used to count and enumerate (sub)optimal alignments, explained also in [13] (chapter 13). The presented framework is pseudo-polynomial on the score values; if the possible score values are not bounded, the algorithms become very slow. All values of $s(\cdot, \cdot)$ and the gap penalties (d and e) are assumed to be integers, whose absolute values are bounded by B . Assuming that the (possible exponential) large numbers used in the computation fit to a fixed size floating point representation with sufficient accuracy, the time complexity of directly evaluating (3), assuming $n \leq m$, is $O(mn^2B)$ and space $O(n^2B)$ suffices. Later on, when time and space complexities are considered, B is assumed to be a small constant. This assumption seems reasonable in the bioinformatics domain. Moreover, the values of m and n are assumed to be of about equal size, i.e. $n = \Theta(m)$. Thus, evaluating (3) needs $O(n^3)$ time and $O(n^2)$ space.

For $E(X_t)$ to be meaningful, the expected score, $\sum_{a,b \in \Sigma} q_a q_b s(a, b)$, must be negative and there must be some characters $a, b \in \Sigma$ for which $s(a, b) > 0$. Moreover, useful thresholds t are ≥ 1 .

Equation (3) generalizes to affine gaps as well. Let d and e be the penalties for gap opening and gap extension, respectively. The dynamic programming becomes as follows:

$$\begin{aligned} M(0, 0, 0) &= M(0, j, 0) = M(i, 0, 0) = M(i, j, 0) = 1 \\ M(i, j, s) &= \\ &\sum_{a,b \in \Sigma} q_a q_b \left(I(i-1, j-1, s-s(a, b)) + M(i-1, j-1, s-s(a, b)) \right) \quad (4) \\ I(i, j, s) &= M(i, j-1, s+d) + M(i-1, j, s+d) \\ &+ I(i, j-1, s+e) + I(i-1, j, s+e), \end{aligned}$$

where $i = 1, \dots, n$, $j = 1, \dots, m$ and $s = 1, \dots, (n \max_{a,b} s(a, b))$. The value of $E(X_t)$ is now $\sum_{i,j} \sum_{s \geq t} (M(i, j, s) + I(i, j, s))$.

This is the simplest formulation for affine gaps from [1]. However, with these equations insertion of k symbols followed by k' deletions gets penalty $d + (k + k' - 1)e$, not $2d - (k + k' - 2)e$ as it should. This can be fixed by splitting table I into two tables I_x , and I_y , where I_x (I_y) accounts insertions to x (y). The time and space complexities of evaluating (4) are $O(n^3)$ and $O(n^2)$, respectively.

4.1 Faster in Linear Space

Both dynamic programming algorithms (3) and (4) compute $O(n^3)$ values of $E(X_t)$, as this expectation is computed for each t and for each sequence lengths up to m and n . In this sense they have an optimal time complexity of $O(n^3)$. However, in practice one is usually interested in only one t for some fixed m and n . Moreover, the quadratic space consumption of the above methods is quite restrictive. Next we show how to compute a single $E(X_t)$ faster using only linear space.

First, consider a global gapless alignment of strings x and y of equal length n . The path of this alignment travels along the diagonal of the alignment graph or grid. The partial score from 1 to i of this alignment is simply $\sum_{j=1}^i s(x_j, y_j)$. The probability that partial score stays positive and is s at position i , denoted by $H(i, s)$, can be computed by dynamic programming:

$$\begin{aligned} H(0, 0) &= 1, H(0, s) = 0 \\ H(i, s) &= \sum_{a,b \in \Sigma} q_a q_b H(i-1, s - s(a, b)), \end{aligned} \tag{5}$$

where $i = 1, \dots, n$ and $s = 1, \dots, n \max_{a,b} s(a, b)$. All values of this table H can be computed in $O(n^2)$ time. The space consumption is linear, as we can compute $H(i, \cdot)$ by keeping only $H(i-1, \cdot)$ in the memory.

If the scoring function $s(a, b)$ is s_1 when $a = b$ and $s(a, b) = s_2$ otherwise, the table H can be computed more efficiently, even when the absolute values s_1 and s_2 are not bounded by a constant. This is because for each i , at most n different values for each $H(i, \cdot)$ need to be stored and evaluated.

4.2 Linear Gaps

With table H , it is possible to compute $E(X_t)$ with linear gaps in $O(n^2)$ time and in $O(n)$ space. The trick is to use the linearity of expectation, and compute the expectation for each $m' \times n'$ submatrix only once, and multiply each expectation by $(n - n' + 1)(m - m' + 1)C$, where C is the number of alignments on each submatrix.

The number of alignments C for a submatrix size $m' \times n'$ is $\binom{n'}{k} \binom{m'}{k}$, where k is the number of matched characters, i.e. this the number of ways to choose k positions from both n' and m' possibilities independently. Term $(n - n' + 1)(m -$

$m' + 1$) is the number of ways to choose $m' \times n'$ sized submatrix (box) inside a matrix (box) of size $m \times n$.

By denoting $H'(i, t) = \sum_{s' \geq t} H(i, s')$, we obtain the following formula

$$E(X_t) = \sum_{k=0}^{\min\{m,n\}} \sum_{n'=k}^n \sum_{m'=k}^m (m-m'+1)(n-n'+1) \binom{n'}{k} \binom{m'}{k} H'(k, t+d(n'+m'-2k)).$$

By noticing that the value of $H'(k, t + d(n' + m' - 2k))$ stays the same for all n', m' for which $n' + m' = l$, we get

$$E(X_t) = \sum_k \sum_l D(k, l) H'(k, t + d(l - 2k)),$$

where $D(k, l) = \sum_{m', n': m'+n'=l} (m - m' + 1)(n - n' + 1) \binom{n'}{k} \binom{m'}{k}$.

Lemma 1. [14] $\sum_{0 \leq k \leq l} \binom{l-k}{m} \binom{q+k}{n} = \binom{l+q+1}{m+n+1}$

Lemma 2. $(m+1) \binom{m}{k} = (k+1) \binom{m+1}{k+1}$

Now we can write $D(k, l)$ as $\sum_{m'} \binom{l-m'}{k} \binom{m'}{k} \left(-(m'+2)(m'+1) + (l-n+m+3)(m'+1) + (m+2)(n-l) \right)$. By using Lemma 2 three times and then using Lemma 1 we get form $D(k, l) =$

$$(k+1)(-(k+2) \binom{l+3}{2k+3} + (l-n+m+3) \binom{l+2}{2k+2}) + (m+2)(n-l) \binom{l+1}{2k+1},$$

which can be evaluated incrementally on l in $O(1)$ time. Thus, we have an algorithm to compute $E(X_t)$ in $O(n^2)$ time and in linear space.

Note that the actual value computed in this way is slightly bigger than one would get from Equation (3). This is due to the fact that gaps are handled as they would occur at the end of the alignment. Depending on how the gaps are put, some alignments with a total score $\geq t$, have a non-positive partial score at some point. This difference can be made smaller by taking into account the fact that the first gap cannot occur at the first position. To take this into account, the term $\binom{n'}{k} \binom{m'}{k}$ is replaced by $\binom{n'-1}{k-1} \binom{m'-1}{k-1}$. Actually, depending on the scoring model and gap penalties, some k' first (and without loss of generality k' last) positions must be matched. This could also be taken into account.

4.3 Affine Gaps

With linear gaps the number of alignments was easy to compute as $\binom{n}{k}$ for length n (sub)sequence with k matches, i.e. positions selected to be aligned. With affine gaps, one has to compute the number of alignments with certain number of consecutive gaps. Next we count the number of these alignments.

Consider a string x to be aligned with k positions selected for matches. Then there are at most $k+1$ gap openings, and exactly $k+1$ gap openings if no two

matches are adjacent and no match occurs at the beginning or at the end of x . Consider that the length of x is n and k positions have been selected from x to be matched. Further, consider that there are two additional matches, one just before position 1, and the other just after position n (at positions 0 and $n + 1$). Now there are $k + 1 - l$ gap openings if there are l adjacent matches. Figure 2 illustrates these gap openings.

Let us now focus on a related problem of computing the number of binary strings of length n with k ones and l adjacent ones, denoted as $\mathcal{N}(n, k, l)$. Ones in this string corresponds to matched positions and zeros are gaps. Further on, $\mathcal{N}(n, k, l)$ can be composed as a sum $\mathcal{Z}(n, k, l) + \mathcal{O}(n, k, l)$, where \mathcal{O} counts strings that end to one and \mathcal{Z} those that end to zero. Formulas for \mathcal{Z} and \mathcal{O} are

$$\mathcal{Z}(n, k, l) = \binom{n-k}{k-l} \binom{k-1}{l}, \text{ and } \mathcal{O}(n, k, l) = \binom{n-k}{k-l-1} \binom{k-1}{l},$$

defining $\binom{n}{k}$ to be non-zero only when $0 \leq k \leq n$. These formulas are derived (and can be proved) from the fact that $\mathcal{O}(n, k, l) = \mathcal{Z}(n-1, k-1, l) + \mathcal{O}(n-1, k-1, l-1)$ and $\mathcal{Z}(n, k, l) = \mathcal{Z}(n-1, k, l) + \mathcal{O}(n-1, k, l)$.

With \mathcal{O} , one can compute the number ways to choose k positions from x with l adjacent positions. This number is $C'(n, k, l) = \mathcal{O}(n+2, k+2, l) - \mathcal{O}(n+1, k+2, l)$, as it is the number of strings of length $n+2$, with $k+2$ ones, from which l are adjacent, and with ones at both ends. When $n-k-1 \geq 0$, C' can be simplified to $\binom{n-k-1}{k-l} \binom{k+1}{l}$, and thus

$$C'(n, k, l) = \begin{cases} \binom{n-k-1}{k-l} \binom{k+1}{l} & , \text{ if } n > k \\ 1 & , \text{ if } n = k \text{ and } l = k + 1 \\ 0 & , \text{ otherwise.} \end{cases}$$

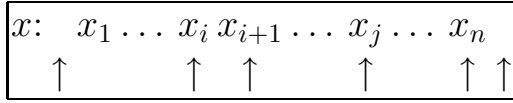


Fig. 2. A string x with four (4) matches, three (3) gap openings and $n - 7$ gap extensions. Positions 0 and $n + 1$ are marked as matches with arrows.

Constant Gap Penalty. The constant gap penalty model is a special case of affine gaps, in which the gap extension penalty is zero. Next we show, that $E(X_t)$ can be computed in $O(n^2 \log n)$ time with constant gap penalties.

As earlier, value $E(X_t)$ can be computed as $\sum_k \sum_{l_x=0}^{k+1} \sum_{l_y=0}^{k+1} \sum_{n'} \sum_{m'} (m - m' + 1)(n - n' + 1)C'(n', k, l_x)C'(m', k, l_y)H'(k, t + (2k + 2 - l_x - l_y)d)$ using variables l_x and l_y to account the number of adjacent matches in x and y . Regrouping gives us

$$\sum_k \sum_{l_x} \sum_{l_y} C(m, k, l_y)C(n, k, l_x)H'(k, t + (2k + 2 - l_x - l_y)d),$$

where $C(m, k, l) = \sum_{m'=k}^m (m - m' + 1)C'(m', k, l)$. Using Lemmas 2 and 3, the value of C can be written as

$$C(m, k, l) = \binom{k+1}{l} \left((m-k+1) \binom{m-k}{k-l+1} - (k-l+1) \binom{m-k+1}{k-l+2} \right).$$

Thus, this formulation gives an $O(n^3)$ time algorithm.

Lemma 3. $\sum_{i=0}^m \binom{i}{k} = \binom{m+1}{k+1}$

This algorithm can be improved by noticing that $H'(k, t + (2k + 2 - l_x - l_y)d)$ is the same for all l_x and l_y that sum to l . This can be utilized by writing $E(X_t)$ as

$$\sum_k \sum_l C(m, n, k, l) H'(k, t + (2k + 2 - l)d),$$

where $C(m, n, k, l) = \sum_{l_x, l_t: l_x + l_t = l} C(m, k, l_y) C(n, k, l_x)$. The value of $C(m, n, k, l)$ for all l can be computed as a convolution $f * g$ of two vectors f and g , $f[l'] = C(m, k, l')$ and $g[l'] = C(n, k, l')$ (As $(f * g)[l] = \sum_{l'} f[l'] g[l - l']$). The convolution can be computed in $O(n \log n)$ time using fast Fourier transformation (FFT) techniques [15]. With these techniques, one achieves an $O(n^2 \log n)$ time algorithm with linear space to compute $E(X_t)$ with constant penalty gaps. In our experiments, we observed that for n and m less than 2000, it is faster to use integer multiplication algorithms [16], like the Karatsuba's algorithm ($O(n^{1.585})$), instead of FFT ($O(n \log n)$).

General Affine Gaps. In this part, we consider general gap penalty model with penalties d for opening and e for extending a gap. By combining all tricks encountered so far allows us to write $E(X_t)$ as

$$\sum_k \sum_{n'} \sum_{m'} \sum_{l_x} \sum_{l_y} (m - m' + 1)(n - n' + 1) C'(n', k, l_x) C'(m', k, l_y) H'(k, t + (2k + 2 - l_x - l_y)(d - e) + e(m' + n' - 2k)).$$

Then, using FFT based convolution, we get $O(n^4 \log n)$ time with linear space. This algorithm is too slow, so with general affine gaps one should use (4). It is left as an open problem, whether it is possible to improve the above algorithm's time complexity, by using only linear space.

4.4 Relation Between $E(X_t)$ and E -score

The E -score of BLAST [3] is similar to expectation $E(X_t)$. However, the E -score is the expected number of high-scoring segment pairs, HSPs. A HSP is a maximal alignment that cannot be improved by extension or trimming [3].

A necessary condition for an alignment path z to be maximal, is that the score decreases if matches at either ends of z are added. This can be utilized in Formula (3) to get an upper bound for p_t that is tighter than $E(X_t)$. The idea is that a match with negative score is added to both ends of each path z , if possible. However, value $c^2 E(X_t)$ is also a good approximation of this new bound, where $c = P(s(a, b) < 0)$. Moreover, $c^2 E(X_t)$ is also approximately equal to the E -score of BLAST.

The E -score E of BLAST can be changed into $1 - e^{-E} \approx p$ -value. Similarly, $e^{-E(X_t)} = e^{-\sum_z P(A_t^z)} = \prod_z e^{-P(A_t^z)} \approx \prod_z (1 - P(A_t^z))$. If events $(x, y) \in A_t^z$ would be independent (they are not), then $e^{-E(X_t)} \approx P(X_t = 0) = 1 - p_t$.

5 Computational Hardness

It is not a coincidence that all presented algorithms are pseudo-polynomial on the score values. It turns out that the problem of computing $E(X_t)$ is NP-hard.

Without loss of generality, consider the expected number of alignments with the score being exactly t , i.e. the values $E(X'_t) = E(X_t) - E(X_{t+1})$. Now, it is NP-hard to decide whether $E(X'_t) > 0$.

The polynomial-time reductions are from the NP-complete Subset Sum [17].

Subset Sum:

Instance: Positive integers a_1, \dots, a_n and a positive integer A .

Question: Is there a subset $I \subset \{1, \dots, n\}$, such that $\sum_{i \in I} a_i = A$?

The alignment instance with string lengths n : First, define alignment scores as

$$s(i, j) = \begin{cases} 2^n a_i + 2^{i-1} & \text{if } i = j \\ 2^{i-1} & \text{if } i \equiv j + 1 \pmod{n} \\ 0 & \text{otherwise.} \end{cases}$$

The background probability is uniform over the alphabet $\Sigma = \{1, \dots, n\}$, i.e. $q_i = \frac{1}{n}$. Further, define the gap penalty $d = 2^n \sum_{i=0}^n a_i + 1$. We do not have to consider gaps as even a single gap would introduce higher penalty than any match of n alphabet pairs.

Now, there is a subset I with $\sum_{i \in I} a_i = A$, if and only if $E(X'_{2^n(A+1)-1}) > 0$. Due to the space limitations, the proof is omitted. □

Similar result can be found for $p'_t = p_t - p_{t+1}$, if $E(X'_{2^n(A+1)-1}) > 0$ is replaced by $p'_{2^n(A+1)-1} > 0$.

6 Experimental Results

We have sampled 10^7 random DNA sequence pairs of equal lengths of 125, 250, 500, 1000, and 2000. The distribution for each A, C, G and T was uniform (we have successfully tested some slightly non-uniform distributions as well). For each sampled pair, we have computed the maximum local alignment score using Smith-Waterman algorithm [2, 6]. The scoring schema is the same that is used in BLASTn, i.e. the score of match is +1 and mismatch -3, gap opening penalty d is 5 and gap extension has penalty $e = 2$. For the linear gap model we have used gap penalty d of 5.

The experimental results in Table 1 show that when $E(X_t) < \frac{1}{2}$, the ratio of $E(X_t)$ and p_t is about 2. This is very close to value $c^{-2} = \frac{16}{9}$ suggested in Subsection 4.4. In Table 2, we have compared small p -values computed by the Karlin-Altschul statistics, by the method of [9], and by our methods. The p -values computed by the Karlin-Altschul statistics seem to vary quite much, depending on the sample size used to fit parameters K and λ . Also the p -values computed with Karlin-Altschul are probably sometimes larger, and sometimes smaller than the actual p_t (suggested by the fact $p_t \leq E(X_t)$). For the experiments of Table 2,

t	$m = n =$		250		500		1000		2000	
	\hat{p}_t	$E(X_t)$	\hat{p}_t	$E(X_t)$	\hat{p}_t	$E(X_t)$	\hat{p}_t	$E(X_t)$	\hat{p}_t	$E(X_t)$
6	0.9332	5.083	1	21.32	1	87.31	1	353.3	1	1422
7	0.4894	1.267	0.9405	5.363	1	22.06	1	89.46	1	360.3
8	0.1542	0.3156	0.5079	1.349	0.9458	5.572	1	22.65	1	91.31
9	0.04092	0.07862	0.1631	0.3392	0.5208	1.408	0.9498	5.734	1	23.14
10	0.0103	0.01958	0.04379	0.0853	0.1693	0.3556	0.5306	1.452	0.9528	5.866
11	0.00257	0.004877	0.01121	0.02145	0.0458	0.08983	0.174	0.3675	0.5385	1.487
12	6.34e-4	0.001214	0.002811	0.005394	0.01172	0.02269	0.04731	0.09305	0.1777	0.3768
13	1.576e-4	3.023e-4	7.077e-4	0.001356	0.002991	0.005732	0.01219	0.02356	0.04831	0.09549
14	3.74e-05	7.524e-5	1.764e-4	3.41e-4	7.482e-4	0.001448	0.003116	0.005964	0.01245	0.0242
15	9.8e-6	1.872e-5	4.16e-5	8.574e-5	1.889e-4	3.658e-4	7.938e-4	0.00151	0.003183	0.006134
16	2.3e-6	4.659e-6	1.08e-5	2.156e-5	4.78e-5	9.24e-5	1.972e-4	3.822e-4	8.145e-4	0.001555
17	4e-7	1.159e-6	2.3e-6	5.419e-6	1.4e-5	2.334e-5	4.73e-5	9.677e-5	2.142e-4	3.940e-4
18	1e-7	2.883e-7	5e-7	1.362e-6	2.5e-6	5.895e-6	1.12e-5	2.45e-5	5.37e-5	9.986e-5
19	0	7.168e-8	1e-7	3.425e-7	8e-7	1.489e-6	2.9e-6	6.203e-6	1.34e-5	2.531e-5
20	0	1.782e-8	0	8.609e-8	1e-7	3.761e-7	5e-7	1.57e-6	3.4e-6	6.415e-6
21	0	4.43e-9	0	2.164e-8	0	9.501e-8	1e-7	3.975e-7	1.1e-6	1.626e-6
22	0	1.101e-9	0	5.44e-9	0	2.4e-8	0	1.006e-7	2e-7	4.12e-7

Table 1. Comparison of sampled p -values (\hat{p}_t) and $E(X_t)$ for affine gaps for each sequence length from 125 to 2000. Sampled values are accurate when $p_t \geq 10^{-5}$. In the equivalent table with linear gaps, the numbers were almost the same (not shown).

Method	$m = n =$				
	125	250	500	1000	2000
KA-100	1.086e-12	1.454e-10	9.196e-12	1.348e-8	2.615e-9
KA-1000	4.906e-12	2.241e-11	2.919e-10	1.687-9	4.963e-9
KA-10000	3.009e-11	2.936-11	2.168e-10	8.975e-10	3.760e-9
Mercier et al.	1.337e-11	5.353e-11	2.142e-10	8.569e-10	3.428e-9
$E(X_t)$	1.687e-11	8.636e-11	3.868e-10	1.633-9	6.708-9
$c^2 E(X_t)$	9.491e-12	4.858e-11	2.176e-10	9.186e-10	3.773e-9

Table 2. Comparison of p -values computed by the Karlin-Altschul statistics (KA- N), the method of Mercier et al. [9] (without gaps), and by our expectation $E(X_t)$ for score $\geq t = 25$ and with varying sequence lengths. Value $c^2 E(X_t)$ is included as the closest estimate of p_t (implied by the Table 1). In KA- N , parameters K and λ are fitted to a sample of N random alignment scores (as the learning is done for each sequence length separately, learned K and λ should reduce the effect of edges). Affine gaps were used.

the true p_t was not known, as the sampling would require about 10^{13} sequence pairs, but based on Table 1, it seems plausible that the value $c^2 E(X_t)$ is very close to the actual p_t .

7 Discussion and Future Work

The results with our framework seem promising and might have implications we do not yet know. The framework is best suited for linear gap penalties, as then the algorithm needs quadratic time and only linear space. Thus, the algorithm is as fast as the Smith–Waterman algorithm [2], that computes the best local alignment score between sequences of interest. For our java implementation, it took about 2.8 seconds to compute the upper bound for linear gapped p -value of two sequences of length 10000 on a standard desktop computer.

It was left as an open problem, how to improve the computation time of $E(X_t)$ with affine gaps. Moreover, the null model used in the framework could be order or position dependent instead of position independent Bernoulli. The problem of having one string fixed and the other being distributed according to the null model is another interesting problem to study in the future.

We have also tried our framework for global alignment score statistics, but it turned out that $E(X_t)$ does not give a good approximation of p_t in this case.

Acknowledgements. The author would like to thank Professor Esko Ukkonen and the anonymous referees for useful comments on this paper.

References

1. Durbin, R., Eddy, S., Krogh, A., Mitchison, G.: *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. first edn. Cambridge University Press (1998)
2. Smith, T., Waterman, M.: Identification of common molecular subsequences. *J.Mol.Biol.* **147** (1981) 195 – 197
3. Altschul, S., Gish, W., Miller, W., Myers, E., Lipman, D.: Basic local alignment search tool. *J.Mol.Biol.* **215** (1990) 403–410
4. Altschul, S., Madden, T., Schaffer, A., Zhang, J., Zhang, Z., Miller, W., Lipman, D.: Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucl. Acids Res.* **25** (1997) 3389–3402
5. Karlin, S., Altschul, S.F.: Methods for assessing the statistical significance of molecular sequence features by using general scoring schemes. *PNAS* **87** (1990) 2264–2268
6. Gotoh, O.: An improved algorithm for matching biological sequences. *J.Mol.Biol.* **162** (1982) 705 – 708
7. Karlin, S., Dembo, A., Kawabata, T.: Statistical composition of high-scoring segments from molecular sequences. *The Annals of Statistics* **18** (1990) 571–581
8. Karlin, S.: Statistical signals in bioinformatics. *Proc Natl Acad Sci U S A* **102** (2005) 13355–13362
9. Mercier, S., Cellier, D., Charlot, F., Daudin, J.J.: Exact and asymptotic distribution of the local score of one i.i.d. random sequence. In: *Proceedings of JOBIM.* (2000) 74–83
10. Pearson, W.: Empirical statistical estimates for sequence similarity searches. *J.Mol.Biol.* **276** (1998) 71 – 84
11. Mitrophanov, A., Borodovsky, M.: Statistical significance in biological sequence analysis. *Briefings in Bioinformatics* **7** (2006) 2–24
12. Naor, D., Brutlag, D.: On suboptimal alignments of biological sequences. In: *Proceedings of CPM, Springer-Verlag* (1993) 179–196
13. Gusfield, D.: *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology*. first edn. Cambridge University Press (1997)
14. Graham, R., Knuth, D., Patashnik, O.: *Concrete mathematics: A Foundation for Computer Science*. second edn. Addison-Wesley (1994)
15. Cooley, J., Tukey, J.: An algorithm for the machine calculation of complex Fourier series. *Mathematics of Computation* **19** (1965) 297–301
16. Bernstein, D.: *Multidigit multiplication for mathematicians* (2001)
17. Garey, M., Johnson, D.: *Computers and Intractability: A Guide to the Theory on NP-Completeness*. W. H. Freeman and Company (1979)