Efficient Computation of Long Similar Subsequences

Abdullah N. Arslan and Ömer Eğecioğlu [⋆]

Department of Computer Science
University of California, Santa Barbara
Santa Barbara, CA 93106 USA
{arslan,omer}@cs.ucsb.edu

Abstract. Given sequences X of length n and Y of length m with $n \geq m$, let LAt^* and $NLAt^*$ denote the maximum ordinary, and maximum *length normalized* scores of local alignments with length at least a given threshold value t. The alignment length is defined as the sum of the lengths of the involved subsequences, and length normalized score of an alignment is the quotient of the ordinary score by the alignment length. We develop an algorithm which finds an alignment with ordinary score $\geq LAt^*$, and length $\geq (1 - \frac{1}{r})t$ for a given r, in time $O(rnm)$ and space $O(rm)$. The algorithm can be used to find an alignment with length normalized score $> \lambda$ for a given positive λ with the same time and space complexity and within the same approximation bounds. Thus this algorithm provides a length-approximate answer to a query such as "Do X and Y share a (sufficiently long) fragment with more than 70% of similarity?" We also show that our approach gives improved approximation algorithms for the *normalized local alignment* problem. In this case we can efficiently find an alignment with length $\geq (1 - \frac{1}{r})t$ which has a length normalized score $\geq NLAt^*$.

Keywords: Local alignment, normalized local alignment, approximation algorithm, dynamic programming, ratio maximization.

1 Introduction

Local sequence alignment aims to reveal similar regions in a given pair of sequences X and Y. The common notion of local similarity suffers from some well-known anomalies resulting from not taking into account the lengths of the subsequences involved in the alignments. The so-called *mosaic effect* in an alignment is observed when a very poor region is sandwiched between two regions with high similarity scores. *Shadow effect* is observed when a biologically important short alignment is not detected because it overlaps with a significantly longer yet biologically inadequate alignment with higher overall score. Several studies in the literature have aimed to describe methods to reduce these anomalies (Arslan and Eğecioğlu, 2002 [6], Arslan et al., 2001 [5], Zhang et al., 1999 [11], Zhang et al., 1998 [10], Altschul et al., 1997 [4]).

⋆ Supported in part by NSF Grants No. CCR–9821038 and No. EIA–9818320.

It is well-known that the statistical significance of local alignment depends on both its score and length (Altschul and Ericson, 1986 [2], 1988 [3]). Alexandrov and Solovyev, 1998 [1] proposed to normalize the alignment score by its length and demonstrated that this new approach leads to better protein classification. Arslan et al., 2001 [5] defined the *normalized local alignment problem* in which the goal is to find subsequences I and J that maximize $s(I,J)/(|I|+|J|)$ among all subsequences I and J with $|I|+|J| \geq t$, where $s(I,J)$ is the score, and t is a threshold for the overall length of I and J . The standard dynamic programming solution to this problem requires cubic time. By dropping the length constraint and changing the objective to the maximization of $s(I,J)/(|I|+|J|+L)$ for real parameter L, it is possible to have some control over the desired alignment lengths while keeping the computational complexity small [5].

In this paper we concentrate on the length constrained version of normalized local alignment. The problem is *feasible* if there is an alignment with positive normalized score and length at least t, where the *length* of an alignment is defined as the sum of the lengths of the subsequences involved in the alignment. We develop an algorithm which provides an approximate control over the total length of the resulting alignment while guaranteeing that the normalized score is maximum achievable by any alignment of length $\geq t$. The approximation ratio is controlled by a parameter r. For a feasible problem, the algorithm returns subsequences with total length $\geq (1-\frac{1}{r})t$. The computations take $O(rnm)$ time and $O(rm)$ space (Theorem 1, section 3). We subsequently revisit the two normalized local alignment algorithms proposed in [5]. In these algorithms we change the subproblems involving ordinary local alignments to those which have a length constraint, and we use the approximation algorithm we present in this paper to solve them. We show that this way we can obtain an alignment which has a normalized score no smaller than the optimum score of the original normalized local alignment problem with total length at least $(1-\frac{1}{r})t$ provided that the original problem is feasible (Theorem 2, section 4). In both resulting algorithms the space complexity is $O(rm)$. The number of subproblems that need to be solved is the same as in [5] : While one algorithm establishes that $O(\log n)$ invocations of our approximation algorithm is sufficient, experiments suggest that the other algorithm requires only $3-5$ iterations, resulting in observed $O(rnm)$ time complexity.

2 Background

Given two sequences $X = x_1 x_2 \ldots x_n$ and $Y = y_1 y_2 \ldots y_m$ with $n \geq m$, *alignment graph* $G_{X,Y}$ is used to represent all possible *alignments* between all subsequences of X and Y. It is a directed acyclic graph having $(n+1)(m+1)$ lattice points (u,v) as vertices for $0 \leq u \leq n$, and $0 \leq v \leq m$ (See for example, [9,6]). An *alignment path* for subsequences $x_i \cdots x_k$, and $y_j \cdots y_l$ is a directed path from vertex $(i-1, j-1)$ to (k,l) in $G_{X,Y}$ where $i \leq k$ and $j \leq l$. We will use the terms alignment and alignment path interchangeably.

The objective of sequence alignment is to quantify the similarity between two sequences. There are various scoring schemes for this purpose. In the *basic scoring scheme*, the arcs of $G_{X,Y}$ are assigned weights determined by non-negative reals δ (*mismatch penalty*) and μ (*indel* or *gap penalty*). We assume that $s(x_i, y_j)$ is the similarity score between the symbols x_i, and y_j which is 1 for a match ($x_i = y_j$) and $-\delta$ for a mismatch ($x_i \neq y_j$). The following is the classical dynamic programming formulation ([9]) to compute the maximum local alignment score $\mathcal{S}_{i,j}$ ending at each vertex (i, j):

$$\mathcal{S}_{i,j} = \max\{0,\ \mathcal{S}_{i-1,j} - \mu,\ \mathcal{S}_{i-1,j-1} + s(x_i, y_j),\ \mathcal{S}_{i,j-1} - \mu\} \qquad (1)$$

for $1 \leq i \leq n$, $1 \leq j \leq m$, with the boundary conditions $\mathcal{S}_{i,j} = 0$ whenever $i = 0$ or $j = 0$.

Let $\subseteq$ indicate the subsequence relation. The *local alignment* (LA) problem seeks subsequences $I \subseteq X$ and $J \subseteq Y$ with the highest similarity score. The optimum local alignment score $LA^*(X, Y)$ is defined as $LA^*(X, Y) = \max\{s(I, J) \mid I \subseteq X, J \subseteq Y\} = \max\limits_{i,j} \mathcal{S}_{i,j}$, where $s(I, J) > 0$, hereinafter, is the best local alignment score between I and J. LA^* can be computed using the Smith-Waterman algorithm [8] in time $O(nm)$ and space $O(m)$.

In what follows, for any optimization problem $\mathcal{P}$, we denote by $\mathcal{P}^*$ its optimum value, and sometimes drop the parameters from the notation when they are obvious from the context. We call $\mathcal{P}$ *feasible* if it has a solution with the given parameters.

As in [5] the objective of the *normalized local alignment* problem ($NLAt$) can be written as

$$NLAt^*(X, Y) = \max\{s(I, J)/(|I| + |J|) \mid I \subseteq X, J \subseteq Y, |I| + |J| \geq t\} \qquad (2)$$

In general optimal alignments for LA and $NLAt$ are different (see [5] for a detailed example).

3 Finding long alignments with high ordinary score

For a given t, we define the *local alignment with length threshold* score between X and Y as

$$LAt^*(X, Y) = \max\{s(I, J) \mid I \subseteq X, J \subseteq Y, \text{ and } |I| + |J| \geq t\} \qquad (3)$$

To solve LAt we can extend the dynamic programming formulation in (1) by adding another dimension. At each entry of the dynamic programming matrix we store optimum scores for all possible lengths up to $m+n$, increasing the time and space complexity to $O(n^2m)$ and $O(nm)$, respectively, which are unacceptably high in practice.

We give an approximation algorithm $APX\text{-}LAt$ which computes a local alignment whose score is at least LAt^*, and whose length is at least $(1 - \frac{1}{r})t$ provided that the LAt problem is feasible, i.e. $s(\widehat{I}, \widehat{J}) \geq LAt^*$ and $|\widehat{I}| + |\widehat{J}| \geq (1 - \frac{1}{r})t$.

For simplicity, we assume a basic scoring scheme. Our approximation idea is similar to that in Arslan and Eğecioğlu, 2002 [6]. Instead of a single score, we maintain at each node (i,j) of $G_{X,Y}$, a list of alignments with the property that for positive s where s is the optimum score achievable over the set of alignments with length $\geq t$ and ending at (i,j), at least one element of the list actives score s and length $t-\Delta$ where Δ is a positive integral parameter. We show that the dynamic programming formulation can be extended to preserve this property through the nodes. In particular, an alignment with score $\geq LAt^*$, and length $\geq t-\Delta$ will be observed in one of the nodes (i,j) during the computations.

We imagine the vertices of $G_{X,Y}$ as grouped into $\lfloor (n+m)/\Delta \rfloor$ diagonal slabs at distance Δ from each other as shown in Figure 1. The length of a diagonal arc is 2 while the length of each horizontal, or vertical arc is 1 . Each slab consists of $\lfloor \Delta/2 \rfloor + 1$ diagonals. Two consecutive slabs share a diagonal which we call a *boundary* . The *left* and the *right boundaries* of slab b are respectively the boundaries shared by the left and right neighboring slabs of b. As a subgraph, a slab contains all the edges in $G_{X,Y}$ incident to the vertices in the slab except for the horizontal and vertical edges incident to the vertices on the left boundary (which belong to the preceding slab), and the diagonal edges incident to the vertices on the first diagonal following the left boundary.

Now to a given diagonal d in $G_{X,Y}$, we associate a number of slabs as follows. Let *slab 0 with respect to diagonal* d be the slab that contains the diagonal d itself. The slabs to the left of *slab 0* are then ordered consecutively as *slab 1, slab 2,* ... with respect to d. In other words, *slab k* with respect to diagonal d is the subgraph of $G_{X,Y}$ composed of vertices placed inclusively between diagonals $\lfloor d/\Delta \rfloor$ and d if $k=0$, and between diagonal $(\lfloor d/\Delta \rfloor - k)\Delta$ and $(\lfloor d/\Delta \rfloor - k + 1)\Delta$, otherwise. Figure 1 includes sample slabs with respect to diagonal d, and alignments ending at some node (i,j) on this diagonal.

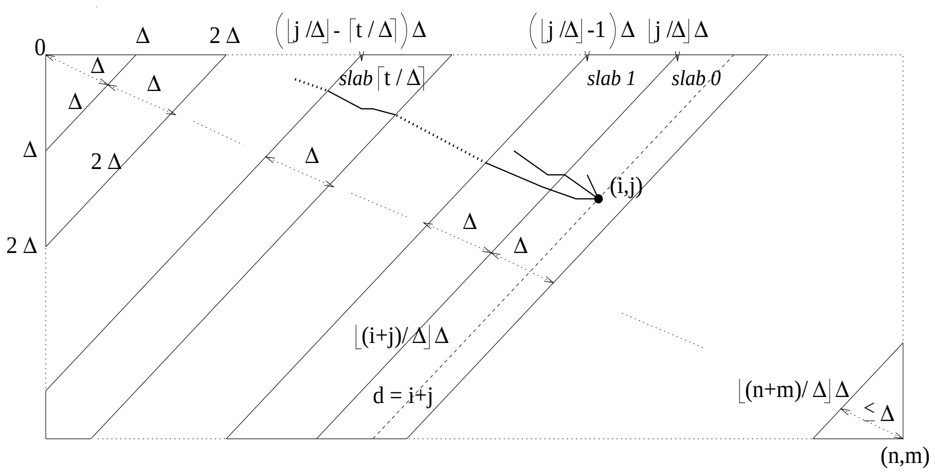


Fig. 1. Slabs with respect to diagonal d, and alignments ending at node (i,j) starting at different slabs.

Let $\mathcal{S}_{i,j,k}$ represent the optimum score achievable at (i,j) by any alignment starting at slab k with respect to diagonal $i+j$ for $0 \le k < \lceil t/\Delta \rceil$. For $k = \lceil t/\Delta \rceil$, $\mathcal{S}_{i,j,k}$ is slightly different: It is the maximum of all achievable scores by an alignment starting in or before slab k . Also let $\mathcal{L}_{i,j,k}$ be the length of an optimal alignment starting at slab k, and achieving score $\mathcal{S}_{i,j,k}$. A single slab can contribute at most Δ to the length of any alignment. We store at each node (i,j) $\lceil t/\Delta \rceil + 1$ score-length pairs $(\mathcal{S}_{i,j,k}, \mathcal{L}_{i,j,k})$ for $0 \le k \le \lceil t/\Delta \rceil$ corresponding to $\lceil t/\Delta \rceil + 1$ optimal alignments that end (i,j) . Figure 2 shows the steps of our approximation algorithm $APX\text{-}LAt$. The processing is done row-by-row starting with the top row $(i=0)$ of $G_{X,Y}$.

Step 1 of the algorithm performs the initialization of the lists of the nodes in the top row $(i=0)$. Step 2 implements computation of scores as dictated by the dynamic programming formulation in (1). Let maxp of a list of score-length pairs be a pair with the maximum score in the list. We obtain an optimal alignment with score $\mathcal{S}_{i,j,k}$ by extending an optimal alignment from one of the nodes $(i-1,j)$, $(i-1,j-1)$, or $(i,j-1)$. We note that extending an alignment at (i,j) from node $(i-1,j-1)$ increases the length by 2 and the score by $s(x_i,y_j)$, whereas from nodes $(i-1,j)$ or $(i,j-1)$ adds 1 to the length and $-\mu$ to the score of the resulting alignment. There are two cases:

(1) If the current node (i,j) is not on the first diagonal after a boundary then nodes $(i-1,j)$, $(i-1,j-1)$ and $(i,j-1)$ share the same slabs with node (i,j) . In this case $(\mathcal{S}_{i,j,k}, \mathcal{L}_{i,j,k})$ is calculated by using $(\mathcal{S}_{i-1,j,k}, \mathcal{L}_{i-1,j,k})$, $(\mathcal{S}_{i-1,j-1,k}, \mathcal{L}_{i-1,j-1,k})$, and $(\mathcal{S}_{i,j-1,k}, \mathcal{L}_{i,j-1,k})$ as shown in Step $2.b$ where $(\mathcal{S}_{i-1,j-1,k}, \mathcal{L}_{i-1,j-1,k}) \oplus (s(x_i,y_j),2) = (\mathcal{S}_{i-1,j-1,k} + s(x_i,y_j), \mathcal{L}_{i-1,j-1,k} + 2)$ if $\mathcal{S}_{i-1,j-1,k} > 0$ or $k=0$; and $(0,0)$ otherwise. This is because, by definition, every local alignment has a positive score, and it is either a single match, or it is an extension of an alignment whose score is positive. Therefore we do not let an alignment with no score be extended unless the resulting alignment is a single match in the current slab.

(2) If the current node is on the first diagonal following a boundary (i.e. $i+j \mod \Delta = 1$) then the slabs for the nodes involved in the computations for node (i,j) differ as shown in Figure 3. In this case slab k for node (i,j) is slab $k-1$ for nodes $(i-1,j)$, $(i-1,j-1)$ and $(i,j-1)$. Moreover any alignment ending at (i,j) starting at slab 0 for (i,j) can only include one of the edges $((i-1,j),(i,j))$ or $((i-1,j-1),(i,j))$ both of which have negative weight $-\mu$. Therefore, $(\mathcal{S}_{i,j,0}, \mathcal{L}_{i,j,0})$ is set to $(0,0)$. Steps $2.a.1$ and $2.a.2$ show the calculation of $(\mathcal{S}_{i,j,k}, \mathcal{L}_{i,j,k})$ respectively for $0 < k < \lceil t/\Delta \rceil$ and for $k = \lceil t/\Delta \rceil$.

The running maximum score $\widehat{LAt}$ is updated whenever a newly computed score for an alignment with length $\ge t - \Delta$ is larger than the current maximum which can only happen with alignments starting in or before slab $\lceil t/\Delta \rceil - 1$. The final value $\widehat{LAt}$ is returned in Step 3. The alignment position achieving this score may also be desired. This can be done by maintaining for each optimal alignment a start and end position information besides its score and length. In this case in addition to the running maximum score, the start and end positions of a maximal alignment should be stored and updated.

```
Algorithm $APX\text{-}LAt(\delta,\mu)$
1. Initialization:
   set $\widehat{LAt}=0$
   set $(\mathcal{S}_{0,j,k},\mathcal{L}_{0,j,k})=(0,0)$ for all $j,k$,  $0\le j\le m$, and $0\le k\le\lceil t/\Delta\rceil$
2. Main computations:
   for $i=1$ to $n$ do
   {
     set $(\mathcal{S}_{i,0,k},\mathcal{L}_{i,0,k})=(0,0)$ for all $k$, $0\le k\le\lceil t/\Delta\rceil$
     for $j=1$ to $m$ do
     {
       if $(i+j$ mod $\Delta=1)$ then
        {
         set $(\mathcal{S}_{i,j,0},\mathcal{L}_{i,j,0})=(0,0)$
         for $k=1$ to $\lceil t/\Delta\rceil-1$ do
2.a.1       set $(\mathcal{S}_{i,j,k},\mathcal{L}_{i,j,k})=\mathrm{maxp}\{$ $(0,0)$, $(\mathcal{S}_{i-1,j,k-1},\mathcal{L}_{i-1,j,k-1})+(-\mu,1)$,
                                        $(\mathcal{S}_{i-1,j-1,k-1},\mathcal{L}_{i-1,j-1,k-1})\oplus(s(x_i,y_j),2)$,
                                        $(\mathcal{S}_{i,j-1,k-1},\mathcal{L}_{i,j-1,k-1})+(-\mu,1)$ $\}$
         for $k=\lceil t/\Delta\rceil$
2.a.2       set $(\mathcal{S}_{i,j,k},\mathcal{L}_{i,j,k})=\mathrm{maxp}\{$ $(0,0)$, $(\mathcal{S}_{i-1,j,k-1},\mathcal{L}_{i-1,j,k-1})+(-\mu,1)$,
                                        $(\mathcal{S}_{i-1,j-1,k-1},\mathcal{L}_{i-1,j-1,k-1})\oplus(s(x_i,y_j),2)$,
                                        $(\mathcal{S}_{i,j-1,k-1},\mathcal{L}_{i,j-1,k-1})+(-\mu,1)$,
                                        $(\mathcal{S}_{i-1,j,k},\mathcal{L}_{i-1,j,k})+(-\mu,1)$,
                                        $(\mathcal{S}_{i-1,j-1,k},\mathcal{L}_{i-1,j-1,k})\oplus(s(x_i,y_j),2)$,
                                        $(\mathcal{S}_{i,j-1,k},\mathcal{L}_{i,j-1,k})+(-\mu,1)$ $\}$
        } else
          {
            for $k=0$ to $\lceil t/\Delta\rceil$ do
2.b             set $(\mathcal{S}_{i,j,k},\mathcal{L}_{i,j,k})=\mathrm{maxp}\{$ $(0,0)$, $(\mathcal{S}_{i-1,j,k},\mathcal{L}_{i-1,j,k})+(-\mu,1)$,
                                            $(\mathcal{S}_{i-1,j-1,k},\mathcal{L}_{i-1,j-1,k})\oplus(s(x_i,y_j),2)$,
                                            $(\mathcal{S}_{i,j-1,k},\mathcal{L}_{i,j-1,k})+(-\mu,1)$ $\}$
          }
        for $k=\lceil t/\Delta\rceil-1$ if $\mathcal{L}_{i,j,k}\ge t-\Delta$ then set $\widehat{LAt}=\max\{\widehat{LAt},\mathcal{S}_{i,j,k}\}$
        for $k=\lceil t/\Delta\rceil$ set $\widehat{LAt}=\max\{\widehat{LAt},\mathcal{S}_{i,j,k}\}$
     }
   }
3. Return $\widehat{LAt}$
```

Fig. 2. Algorithm *APX-LAt*.

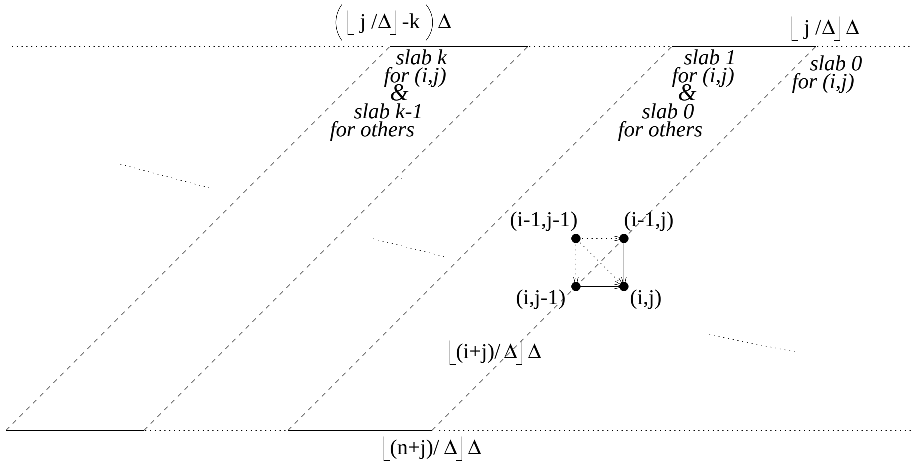


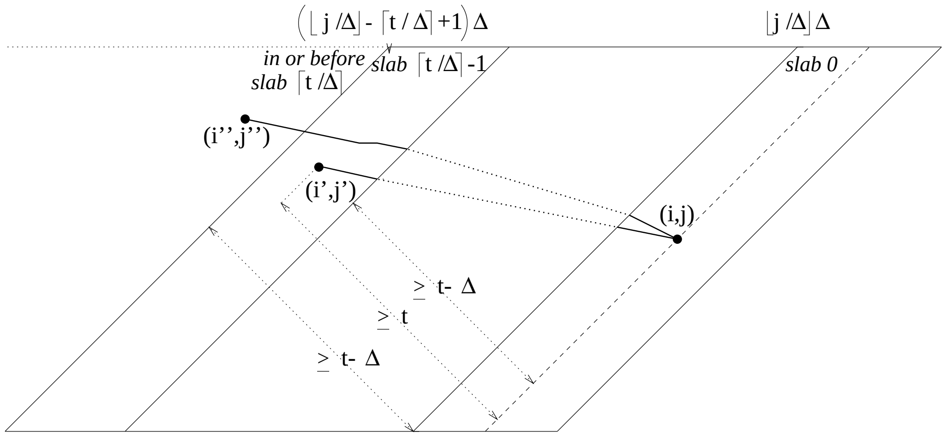
Fig. 3. Relative numbering of the slabs with respect to (i,j), $(i-1,j)$, $(i-1,j-1)$ and $(i,j-1)$ when node (i,j) is on the first diagonal following boundary $\lfloor (i+j)/\Delta \rfloor$.

We first show that $\mathcal{S}_{i,j,k}$ calculated by the algorithm is the optimum score achievable and $\mathcal{L}_{i,j,k}$ is the length of an alignment achieving this score over the set of all alignments ending at node (i,j) and starting with respect to diagonal $i+j$: 1) at slab k for $0 \le k < \lceil t/\Delta \rceil$, 2) in or before slab k for $k = \lceil t/\Delta \rceil$. This claim can be proved by induction. If we assume that the claim is true for nodes $(i-1,j)$, $(i-1,j-1)$ and $(i,j-1)$, and for their slabs, then we can easily see by following Step 2 of the algorithm that the claim holds for node (i,j) and its slabs.

Let optimum score LAt^* for the alignments of length $\ge t$ be achieved at node (i,j) . Consider the calculations of the algorithm at (i,j) at which an optimal alignment ends. There are two possible orientations of an optimal alignment as shown in Figure 4: 1) It starts at some node (i',j') of slab $k = \lceil t/\Delta \rceil - 1$. By our previous claim an alignment starting at slab k with score $\mathcal{S}_{i,j,k} \ge LAt^*$ is captured in Step 2. The length of this alignment $\mathcal{L}_{i,j,k}$ is at least $t - \Delta$ since the length of the optimal alignment is $\ge t$, and both start at the same slab and end at (i,j). 2) It starts at some node (i'',j'') in or before slab $k = \lceil t/\Delta \rceil$. Again by the previous claim an alignment starting in or before slab k with score $\mathcal{S}_{i,j,k} \ge LAt^*$ is captured in Step 2. The length of this alignment $\mathcal{L}_{i,j,k}$ is at least $t-\Delta$ since slab k is at distance $\ge t-\Delta$ from (i,j) . Therefore the final value $\widehat{LAt}$ returned in Step 3 is $\ge LAt^*$ and it is achieved by an alignment whose length is $\ge t - \Delta$. We summarize these results in the following theorem.

Theorem 1. *For a feasible LAt problem, Algorithm APX-LAt returns an alignment* $(\widehat{I},\widehat{J})$ *such that* $s(\widehat{I},\widehat{J}) \ge LAt^*$ *and* $|\widehat{I}| + |\widehat{J}| \ge (1-\frac{1}{r})t$ *for any* $r > 1$. *The algorithm's complexity is* $O(rnm)$ *time and* $O(rm)$ *space.*

Proof. Algorithm *APX-LAt* is similar to the Smith-Waterman algorithm except that at each node instead of a single score, $\lceil t/\Delta \rceil + 1$ entries for score-length pairs are stored and manipulated. Therefore the resulting complexity exceeds that



$(\lfloor j/\Delta \rfloor - \lceil t/\Delta \rceil + 1)\Delta$
$\lfloor j/\Delta \rfloor \Delta$
in or before slab $\lceil t/\Delta \rceil$
slab $\lceil t/\Delta \rceil - 1$
slab 0
(i'',j'')
(i',j')
(i,j)
$\geq$ t- Δ
$\geq$ t
$\geq$ t- Δ

Fig. 4. Two possible orientations of an optimal alignment of length $\geq t$ ending at (i, j): it starts either at some (i', j') at slab $\lceil t/\Delta \rceil - 1$, or (i'', j'') in or before slab $\lceil t/\Delta \rceil$.

of the Smith-Waterman algorithm by a factor of $\lceil t/\Delta \rceil + 1$. That is, the time complexity of $APX\text{-}LAt$ is $O(nmt/\Delta)$. The algorithm requires $O(mt/\Delta)$ space since we need the entries in the previous and the current row to calculate the entries in the current row. When the LAt problem is feasible, it is guaranteed that Algorithm $APX\text{-}LAt$ returns an alignment $(\widehat{I}, \widehat{J})$ such that $s(\widehat{I}, \widehat{J}) \geq LAt^* > 0$ and $|\widehat{I}| + |\widehat{J}| \geq t - \Delta$ for any positive Δ . Therefore setting $\Delta = \lfloor t/r \rfloor$ for a choice of r, $1 < r \leq t$, and using Algorithm $APX\text{-}LAt$ we can achieve the approximation and complexity results expressed in the theorem. We also note that for $\Delta = 1$ the algorithm becomes a dynamic programming algorithm extending the dimension by storing all possible alignment lengths.

4 Finding long alignments with high normalized score

We consider the problem Qt of finding two subsequences with normalized score higher than λ, and total length at least t . More formally

$$Qt: \text{ find } (I, J) \text{ such that } I \subseteq X, J \subseteq Y, \ \frac{s(I,J)}{|I| + |J|} > \lambda \text{ and } |I| + |J| \geq t \quad (4)$$

We note that Qt is feasible iff $NLAt^* > \lambda$. We present an approximation algorithm which provided that Qt is feasible finds two subsequences $\widehat{I} \subseteq X$, and $\widehat{J} \subseteq Y$ with normalized score higher than λ, and $|\widehat{I}| + |\widehat{J}| \geq (1 - \frac{1}{r})t$.

Let AVt denote the *set of alignment vectors* between X and Y where (x, y, z) is an *alignment vector* if there is an alignment between subsequences $I \subseteq X$ and $J \subseteq Y$ with x matches, y mismatches, and z indels such that $|I| + |J| \geq t$. Then $s(I, J)$ and length $|I| + |J|$ are linear functions over AVt . Problems LAt and $NLAt$ can be rewritten as follows :

$$\begin{array}{lll} LAt_{\delta,\mu} & : maximize\ x - \delta y - \mu z & \text{s.t. } (x,y,z) \in AVt \\ NLAt_{\delta,\mu} & : maximize\ \frac{x - \delta y - \mu z}{2x + 2y + z} & \text{s.t. } (x,y,z) \in AVt \end{array}$$

Also for a given λ, we have *the parametric local alignment with length threshold problem* $LAt(\lambda)$

$$LAt_{\delta,\mu}(\lambda) : maximize\ x - \delta y - \mu z \ - \ \lambda(2x + 2y + z) \quad \text{s.t. } (x, y, z) \in AVt$$

A parametric local alignment with length threshold problem can be described in terms of a local alignment with length threshold problem.

Proposition 1. *For $\lambda \neq \frac{1}{2}$, the optimum value $LAt^*(\lambda)$ of the parametric LAt problem can be formulated in terms of the optimum value LAt^* of an LAt problem.*

Proof. The optimum value of the parametric problem, when $\lambda \neq \frac{1}{2}$, is

$$LAt^*(\lambda) = (1 - 2\lambda) LAt^*_{\delta',\mu'} \text{ where } \delta' = \frac{\delta + 2\lambda}{1 - 2\lambda},\ \mu' = \frac{\mu + \lambda}{1 - 2\lambda}\ . \tag{5}$$

Thus, computing $LAt^*(\lambda)$ involves solving the local alignment problem $LAt_{\delta',\mu'}$, and performing some simple arithmetic afterward.

We assume without loss of generality that for any alignment the score does not exceed the number of matches. Therefore for any alignment vector, its normalized score $\lambda \leq \frac{1}{2}$. We consider $\lambda = \frac{1}{2}$ as a special case since it can only happen when the alignment is composed of matches only.

An optimal solution to a ratio optimization problem $NLAt$ can be achieved via a series of optimal solutions of the parametric problem with different parameters $LAt(\lambda)$. In fact $\lambda = NLAt^*$ iff $LAt^*(\lambda) = 0$. Details for a very similar result can be found in [5].

Proposition 2. *When solving $LAt(\lambda)$, Algorithm APX-LAt returns an alignment $(\widehat{I}, \widehat{J})$ with normalized score higher than λ, and $|\widehat{I}| + |\widehat{J}| \geq (1 - \frac{1}{r})t$ if Problem Qt is feasible.*

Proof. Assume that Problem Qt is feasible. Then $NLAt^* > \lambda$, and therefore $LAt^*(\lambda) > 0$ which implies that Algorithm APX-LAt with parameters δ' and μ' (of Proposition 1) returns an alignment $(\widehat{I}, \widehat{J})$ such that its score is positive (i.e. $s(\widehat{I}, \widehat{J}) - \lambda(|\widehat{I}| + |\widehat{J}|) > 0$, or equivalently its normalized score is higher than λ) and $|\widehat{I}| + |\widehat{J}| \geq (1 - \frac{1}{r})t$ by the approximation results of Algorithm APX-LAt .

Theorem 2. *If $NLAt^* > 0$ then an alignment with normalized score at least $NLAt^*$, and total length at least $(1 - \frac{1}{r})t$ can be computed for any $r > 1$ in time $O(rnm \log n)$, and using $O(rm)$ space.*

Proof. Algorithm $RationalNLAt$ given in Figure 5 accomplishes this. The algorithm is based on a binary search for optimum normalized score over an interval of integers. This takes $O(\log n)$ parametric problems to solve. The algorithm is similar to the $RationalNLA$ algorithm in [5], and the results are derived similarly.

```
Algorithm $APX$-$RationalNLAt$
If there is an exact match of size $(1-\frac{1}{r})t$ then return($\frac{1}{2}$) and exit
$\sigma \leftarrow \frac{1}{qs(m+n)^2}$ where $\delta = \frac{p}{q}$, and $\mu = \frac{r}{s}$
$[e,f] \leftarrow [0, \frac{1}{2}\sigma^{-1}]$
$\lambda^* \leftarrow 0$
While $(e+1<f)$ do
  $k \leftarrow \lceil (e+f)/2 \rceil$
  If $APX$-$LAt(k\sigma) > 0$ then $\{e \leftarrow k$, and $\lambda^* \leftarrow \frac{x-\delta y-\mu z}{2x+2y+z}$ for $(x,y,z)$ optimal $\}$
  else $f \leftarrow k$
End {while}
Return($\lambda^*$)
```

Fig. 5. Algorithm *APX-RationalNLAt* for rational scores.

If $NLAt^* > 0$ then we can also achieve the same approximation guarantee by using a Dinkelbach algorithm given in [5] as the template. The details of the resulting algorithm are presented in Figure 6. Solutions of the parametric problems through the iterations yield improved (higher) values to λ except for the last iteration. The resulting algorithm performs no more than $3-5$ iterations on the average as experiments suggest.

```
Algorithm Dinkelbach
If $APX$-$LAt(0) \leq 0$ then return(0) and exit
$\lambda^* \leftarrow \frac{x-\delta y-\mu z}{2x+2y+z}$ where $(x,y,z)$ is optimal for $APX$-$LAt(0)$
Repeat
   $\lambda \leftarrow \lambda^*$
   if $APX$-$LAt(\lambda) > 0$ then $\lambda^* \leftarrow \frac{x-\delta y-\mu z}{2x+2y+z}$ for $(x,y,z)$ optimal
Until $\lambda^* \leq \lambda$
Return($\lambda^*$)
```

Fig. 6. `Dinkelbach` algorithm for *NLAt*.

Our approximation and complexity results hold for two particularly important cases of scoring schemes: *affine gap penalties*, and *arbitrary scoring matrices*. We can develop variants of Algorithm *APX-LAt* for these scoring schemes with simple modifications. In the case of arbitrary scoring matrices, penalties depend on individual symbols involved in the operations. Varying penalties can easily be incorporated in the dynamic programming formulation. In the case of affine gap penalties, the total penalty of a *gap* (a block of insertions, or a block of deletions) of size k is $\alpha + k\beta$ where α, and β are the *gap open penalty*, and the *gap extension penalty*, respectively. Affine gap penalties require a slightly different dynamic programming formulation than the one given for basic scoring scheme (1). It can be described as follows ([9]) : Let $\mathcal{E}_{i,j} = \mathcal{F}_{i,j} = \mathcal{S}_{i,j} = 0$ when i or j

is 0 then define

$$\begin{aligned}\mathcal{E}_{i,j} &= \max\{\mathcal{S}_{i,j-1} - \alpha,\ \mathcal{E}_{i,j-1} - \beta\},\\ \mathcal{F}_{i,j} &= \max\{\mathcal{S}_{i-1,j} - \alpha,\ \mathcal{F}_{i-1,j} - \beta\},\\ \mathcal{S}_{i,j} &= \max\{0,\ \mathcal{S}_{i-1,j-1} + s(x_i, y_j),\ \mathcal{E}_{i,j},\ \mathcal{F}_{i,j}\} \qquad (6)\end{aligned}$$

Affine gap penalties do not increase the complexity of the local alignment problem, i.e. the problem can be solved in time $O(nm)$ and using $O(m)$ space. Figure 7 shows the variant of Algorithm *APX-LAt* for affine gap penalties. The approximation and complexity results expressed in Theorem 1 can be obtained by Algorithm *APX-LAt-AFFINE* for affine gap penalties.

We can verify that in both cases of these scoring schemes a parametric *LAt* problem can easily be formulated in terms of an *LAt* problem. We can develop variants of *NLAt* algorithms for them such that the same approximation and complexity results hold.

5 Implementation and test results

We have implemented versions of Algorithm *APX-LAt* and *Dinkelbach* for affine gap penalties and tested our *Dinkelbach* program on *bli-4* locus in *C. elegans* and *C.briggsae* for various values of parameters t and r. We have observed that the program performs $3-5$ invocations of *APX-LAt* implementation on the average. Therefore for reasonable choice of r its time requirement is $3r$ to $5r$ times that of a Smith-Waterman implementation on the average. In Figure 9, we include results for optimal alignments obtained as t runs from $1,000$ to $22,000$ in increments of $1,000$, and from $30,000$ to $90,000$ in increments of $10,000$, and for fixed $r = 5$. On a Beowulf class super-computer which is composed of a cluster of 42 linux-based 400-500 Mhz workstations it took about 8 days to complete the tests. We note that we could use a fast heuristic algorithm to solve the parametric local alignment problems and improve the running time by orders of magnitude, but then the approximation guarantee of the results no longer holds.

We have used a score of 1 for a match, -1 for a mismatch, and $-6 - 0.2k$ for a gap of length k. In Figure 9, we have multiplied the normalized scores by $10,000$ to be able to display them on the same scale as the ordinary scores. As expected in general, normalized scores steadily decrease with the increasing alignment lengths. The alignments whose lengths exceed $32,100$ include regions with very poor scores.

Test runs like this can generate important statistical information. For instance in this case we can infer from our approximation results and from the normalized score 0.33 of the alignment with length $16,048$ that 0.33 cannot be obtained by any alignment whose length exceeds $16,048/(1-1/5) \approx 20,000$.

```
Algorithm $APX\text{-}LAt\text{-}AFFINE(\delta,\alpha,\beta)$
1. Initialization:
   set $\widehat{LAt} = 0$
   set $(\mathcal{E}_{0,j,k},\mathcal{L}^{\mathcal{E}}_{0,j,k}) = (\mathcal{F}_{0,j,k},\mathcal{L}^{\mathcal{F}}_{0,j,k}) = (\mathcal{S}_{0,j,k},\mathcal{L}^{\mathcal{S}}_{0,j,k}) = (0,0)$
       for all $j,k$,  $0 \le j \le m$,  $0 \le k \le \lceil t/\Delta \rceil$
2. Main computations :
   for $i = 1$ to $n$ do {
    set $(\mathcal{E}_{i,0,k},\mathcal{L}^{\mathcal{E}}_{i,0,k}) = (\mathcal{F}_{i,0,k},\mathcal{L}^{\mathcal{F}}_{i,0,k}) = (\mathcal{S}_{i,0,k},\mathcal{L}^{\mathcal{S}}_{i,0,k}) = (0,0)$
       for all $k$,  $0 \le k \le \lceil t/\Delta \rceil$
    for $j = 1$ to $m$ do {
      if $(i+j \bmod \Delta = 1)$ then {
        set $(\mathcal{E}_{i,j,0},\mathcal{L}^{\mathcal{E}}_{i,j,0}) = (\mathcal{F}_{i,j,0},\mathcal{L}^{\mathcal{F}}_{i,j,0}) = (\mathcal{S}_{i,j,0},\mathcal{L}^{\mathcal{S}}_{i,j,0}) = (0,0)$
        for $k = 1$ to $\lceil t/\Delta \rceil - 1$ do {
          set $(\mathcal{E}_{i,j,k},\mathcal{L}^{\mathcal{E}}_{i,j,k}) = \max\{$ $(\mathcal{S}_{i,j-1,k-1},\mathcal{L}^{\mathcal{S}}_{i,j-1,k-1}) + (-\alpha,1)$,
                                    $(\mathcal{E}_{i,j-1,k-1},\mathcal{L}^{\mathcal{E}}_{i,j-1,k-1}) + (-\beta,1)$ $\}$
          set $(\mathcal{F}_{i,j,k},\mathcal{L}^{\mathcal{F}}_{i,j,k}) = \max\{$ $(\mathcal{S}_{i-1,j,k-1},\mathcal{L}^{\mathcal{S}}_{i-1,j,k-1}) + (-\alpha,1)$,
                                    $(\mathcal{F}_{i-1,j,k-1},\mathcal{L}^{\mathcal{F}}_{i-1,j,k-1}) + (-\beta,1)$ $\}$
          set $(\mathcal{S}_{i,j,k},\mathcal{L}^{\mathcal{S}}_{i,j,k}) = \max\{$ $(0,0)$,
                                    $(\mathcal{S}_{i-1,j-1,k-1},\mathcal{L}^{\mathcal{S}}_{i-1,j-1,k-1}) \oplus (s(x_i,y_j),2)$,
                                    $(\mathcal{E}_{i,j,k},\mathcal{L}^{\mathcal{E}}_{i,j,k})$,  $(\mathcal{F}_{i,j,k},\mathcal{L}^{\mathcal{F}}_{i,j,k})$ $\}$
        }
        for $k = \lceil t/\Delta \rceil$ do {
          set $(\mathcal{E}_{i,j,k},\mathcal{L}^{\mathcal{E}}_{i,j,k}) = \max\{$ $(\mathcal{S}_{i,j-1,k-1},\mathcal{L}^{\mathcal{S}}_{i,j-1,k-1}) + (-\alpha,1)$,
                                    $(\mathcal{E}_{i,j-1,k-1},\mathcal{L}^{\mathcal{E}}_{i,j-1,k-1}) + (-\beta,1)$,
                                    $(\mathcal{S}_{i,j-1,k},\mathcal{L}^{\mathcal{S}}_{i-1,j,k}) + (-\alpha,1)$,
                                    $(\mathcal{E}_{i,j-1,k},\mathcal{L}^{\mathcal{E}}_{i,j-1,k}) + (-\beta,1)$ $\}$
          set $(\mathcal{F}_{i,j,k},\mathcal{L}^{\mathcal{F}}_{i,j,k}) = \max\{$ $(\mathcal{S}_{i-1,j,k-1},\mathcal{L}^{\mathcal{S}}_{i-1,j,k-1}) + (-\alpha,1)$,
                                    $(\mathcal{F}_{i-1,j,k-1},\mathcal{L}^{\mathcal{F}}_{i-1,j,k-1}) + (-\beta,1)$,
                                    $(\mathcal{S}_{i-1,j,k},\mathcal{L}^{\mathcal{S}}_{i-1,j,k}) + (-\alpha,1)$,
                                    $(\mathcal{F}_{i-1,j,k},\mathcal{L}^{\mathcal{F}}_{i-1,j,k}) + (-\beta,1)$ $\}$
          set $(\mathcal{S}_{i,j,k},\mathcal{L}^{\mathcal{S}}_{i,j,k}) = \max\{$ $(0,0)$,
                                    $(\mathcal{S}_{i-1,j-1,k-1},\mathcal{L}^{\mathcal{S}}_{i-1,j-1,k-1}) \oplus (s(x_i,y_j),2)$,
                                    $(\mathcal{S}_{i-1,j-1,k},\mathcal{L}^{\mathcal{S}}_{i-1,j-1,k}) \oplus (s(x_i,y_j),2)$,
                                    $(\mathcal{E}_{i,j,k},\mathcal{L}^{\mathcal{E}}_{i,j,k})$,  $(\mathcal{F}_{i,j,k},\mathcal{L}^{\mathcal{F}}_{i,j,k})$ $\}$
        }
      }
 ...
```

Fig. 7. Algorithm *APX-LAt-AFFINE* . The algorithm continues in Figure 8 .

```
            else {
               for $k = 0$ to $\lceil t/\Delta \rceil$ do {
                  set $(\mathcal{E}_{i,j,k}, \mathcal{L}^{\mathcal{E}}_{i,j,k}) = \max\{ (\mathcal{S}_{i,j-1,k}, \mathcal{L}^{\mathcal{S}}_{i,j-1,k}) + (-\alpha, 1),$
                                                   $(\mathcal{E}_{i,j-1,k}, \mathcal{L}^{\mathcal{E}}_{i,j-1,k}) + (-\beta, 1) \}$
                  set $(\mathcal{F}_{i,j,k}, \mathcal{L}^{\mathcal{F}}_{i,j,k}) = \max\{ (\mathcal{S}_{i-1,j,k}, \mathcal{L}^{\mathcal{S}}_{i-1,j,k}) + (-\alpha, 1),$
                                                   $(\mathcal{F}_{i-1,j,k}, \mathcal{L}^{\mathcal{F}}_{i-1,j,k}) + (-\beta, 1) \}$
                  set $(\mathcal{S}_{i,j,k}, \mathcal{L}^{\mathcal{S}}_{i,j,k}) = \max\{ (0,0), (\mathcal{S}_{i-1,j-1,k}, \mathcal{L}^{\mathcal{S}}_{i-1,j-1,k}) \oplus (s(x_i, y_j), 2),$
                                                   $(\mathcal{E}_{i,j,k}, \mathcal{L}^{\mathcal{E}}_{i,j,k}), (\mathcal{F}_{i,j,k}, \mathcal{L}^{\mathcal{F}}_{i,j,k}) \}$
               }
            }
         for $k = \lceil t/\Delta \rceil - 1$ if $\mathcal{L}^{\mathcal{E}}_{i,j,k} \geq t - \Delta$ then set $\widehat{LAt} = \max\{\widehat{LAt}, \mathcal{S}_{i,j,k}\}$
         for $k = \lceil t/\Delta \rceil$ set $\widehat{LAt} = \max\{\widehat{LAt}, \mathcal{S}_{i,j,k}\}$
      }
   }
3. Return $LAt^*$
```

Fig. 8. Continuation of Algorithm *APX-LAt-AFFINE* from Figure 7 .

6 Conclusion

We have developed an algorithm for finding sufficiently long similar subsequences in two sequences of lengths n and m respectively, with $n \geq m$. Given thresholds λ and t the proposed algorithm finds an alignment with a normalized score higher than λ and with total length no smaller than $(1 - \frac{1}{r})t$, provided that the corresponding normalized local alignment problem is feasible. The length of the result can be made arbitrarily close to t by increasing r. This is done at the expense of allocating more resources as the time and space complexities depend on the parameter r as $O(rnm)$ and $O(rm)$ respectively.

Based on the techniques previously proposed in [5], and using the approximation algorithm we present in this paper, we have further developed ways to find an alignment with normalized score no smaller than the maximum normalized score achievable by alignments with length at least t. The alignment returned by the algorithm is guaranteed to have total length $\geq (1 - \frac{1}{r})t$. In our experiments we have observed that the time requirement of the Dinkelbach implementation is $O(rnm)$ on the average. This is better compared to the worst-case time complexity $O(n^2m)$ of the naive algorithm.

We believe that our approximation algorithms have made normalized scores a viable similarity measure in pairwise local alignment as they provide approximate control over the desired alignment lengths. Since the computed normalized score for a particular value of t is an upper bound for the actual normalized scores achievable by sequences of length at least t, these algorithms can also be used to collect statistics about scores of alignments versus length for a particular pair of input sequences.

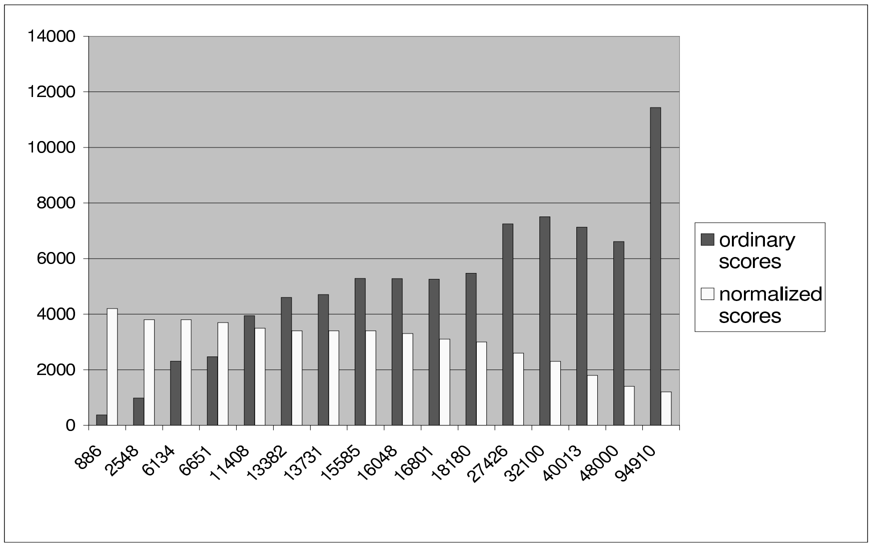


Fig. 9. Ordinary versus normalized scores on *bli-4* locus in *C. elegans* and *C.briggsae* when score of a match is 1, mismatch score is -1, and score for a gap of length k is $-6 - 0.2k$.

References

1. N. N. Alexandrov and V. V. Solovyev. (1998) Statistical significance of ungapped alignments. *Pacific Symposium on Biocomputing (PSB-98), (eds. R. Altman, A. Dunker, L. Hunter, T. Klein)*, pp. 463–472.
2. S. F. Altschul and B. W. Erickson. (1986) Locally optimal subalignments using nonlinear similarity functions. *Bull. of Math. Biology*, 48:633–660.
3. S. F. Altschul and B. W. Erickson. (1988) Significance levels for biological sequence comparison using nonlinear similarity functions. *Bull. of Math. Biology*, 50:77–92.
4. S. F. Altschul, T. L. Madden, A. A. Schaffer, J. Zhang, Z. Zhang, W. Miller, and D. J. Lipman. (1997) Gapped Blast and Psi-Blast: a new generation of protein database search programs. *Nucleic Acids Research*, 25:3389–3402.
5. A. N. Arslan, Ö. Eğecioğlu, and P. A. Pevzner. (2001) A new approach to sequence comparison: Normalized local alignment. *Bioinformatics*, 17(4):327–337.
6. A. N. Arslan and Ö. Eğecioğlu. (2001) Algorithms for local alignments with length constraints. *Technical Report TRCS2001-17*, Department of Computer Science, University of California at Santa Barbara.
7. N. Megiddo. (1979) Combinatorial optimization with rational objective functions. *Mathematics of Operations Research*, 4:414–424.
8. T.F. Smith and M.S. Waterman. (1981) The identification of common molecular subsequences. *J. Mol. Biol.*, 147, 195–197.
9. M. S. Waterman. (1995) *Introduction to computational biology.* Chapman & Hall.
10. Z. Zhang, P. Berman, and W. Miller. (1998) Alignments without low-scoring regions. *J. Comput. Biol.*, 5:197–200.
11. Z. Zhang, P. Berman, T. Wiehe, and W. Miller. (1999) Post-processing long pairwise alignments. *Bioinformatics*, 15:1012–1019.