A New Approach to Sequence Comparison : Normalized Sequence Alignment

Abdullah N. Arslan* and Ömer Eğecioğlu†
Department of Computer Science
University of California, Santa Barbara, Santa Barbara, CA 93106
{arslan, omer}@cs.ucsb.edu

Pavel A. Pevzner
Department of Computer Science and Engineering
University of California, San Diego, San Diego, CA 92093
ppevzner@cs.ucsd.edu

ABSTRACT

The Smith-Waterman algorithm for local sequence alignment is one of the most important techniques in computational molecular biology. This ingenious dynamic programming approach was designed to reveal the highly conserved fragments by discarding poorly conserved initial and terminal segments. However, the existing notion of local similarity has a serious flaw: it does not discard poorly conserved intermediate segments. The Smith-Waterman algorithm finds the local alignment with maximal *score* but it is unable to find local alignment with maximum *degree* of similarity (e.g., maximal percent of matches). Moreover, there is still no efficient algorithm that answers the following natural question: do two sequences share a (sufficiently long) fragment with more than 70% of similarity? As a result, the local alignment sometimes produces a mosaic of well-conserved fragments artificially connected by poorly-conserved or even unrelated fragments. This may lead to problems in comparison of long genomic sequences and comparative gene prediction as recently pointed out by Zhang et al., 1999 [33]. In this paper we propose a new sequence comparison algorithm (*normalized local alignment*) that reports the regions with maximum degree of similarity. The algorithm is based on fractional programming and its running time is $O(n^2 \log n)$. In practice, normalized local alignment is only 3-5 times slower than the standard Smith-Waterman algorithm.

*Supported in part by a UCSB–COR grant.
†Supported in part by NSF Grant No. CCR–9821038.

1. BACKGROUND

Gene prediction in human genome often amounts to using related proteins from other species as clues for finding exon-intron structures (Gelfand et al., 1996 [16], Birney et al., 1996 [11], Pachter et al., 1999 [22]). Recently, a related paradigm, motivated by availability of complete genomes, has emerged (Batzoglou et al., 2000 [10], Bafna and Huson, 2000 [9]). In this new approach, human genes are predicted based on other (e.g., mouse) un-annotated genomic sequences. The idea of this method is that similarity between nucleotide sequences of related human and mouse exons is 85% on average, while similarity between introns is 35% on average. This observation motivates the following simple approach: use local alignment algorithm (Smith and Waterman, 1981 [27]) to find the most similar segments in human and mouse genomic sequences and use these fragments as potential exons at the further stages.

Unfortunately, this approach faces serious difficulties. Smith-Waterman algorithm was developed 20 years ago for a different problem and it is not well suitable for sequence comparison at genomic scale. Surprisingly enough, we still don't have an efficient algorithm that finds the local alignment with the best degree of sequence similarity. The following example illustrates this point.

It is well-known that the statistical significance of the local alignment depends on both its score and length (Altschul and Ericson, 1986 [3], 1988 [4]). However, the score of a local alignment is not normalized over the length of the matching region. As a result, a local alignment with score 1,000 and length 10,000 (*long alignment*) will be chosen over a local alignment with score 998 and length 1,000 (*short alignment*), although the latter one is probably more important biologically. Moreover, if the corresponding alignment paths overlap, the more biologically important "short" alignment won't be detected even by suboptimal sequence alignment algo-

rithm (*shadow effect*). Another unfortunate property of the Smith-Waterman algorithm is that it was designed to exclude non-similar initial and terminal fragments in sequence alignment but it was not designed to exclude non-similar internal fragments. This flaw with Smith-Waterman local similarity approach (Figure 1) leads to inclusion of arbitrarily poor internal fragments (*mosaic effect*). As a result, applications of the Smith-Waterman algorithm to comparison of related genomes (particularly with short introns as *C. elegans* and *C. briggsae*) may lead to problems (Zhang et al., 1999 [33]).

The attempts to fix the problem of mosaic effect undertaken by Goad and Kanehisa, 1982 [17] (who introduced alignment with minimal mismatch density) and Sellers, 1984 [25] did not lead to successful algorithms and were later abandoned. The mosaic effect was first analyzed by Webb Miller (personal communication) and led to some studies trying to fix this problem at the post-processing stage (Huang et al., 1994 [18], Zhang et al., 1999 [33]). Zhang et al., 1999 [33] proposed to decompose a local alignment into sub-alignments that avoid the mosaic effect. However, the post-processing approach may miss the alignments with the best degree of similarity if the Smith-Waterman algorithm missed them. As a result, highly similar fragments may be ignored if they are not parts of larger alignments dominating other local similarities. Another approach to fixing the problems with the Smith-Waterman algorithm is based on the notion of X-*drop*, a region within an alignment that scores below X. The alignments that contain no X-drops are called X-*alignments*. Although X-alignments are expensive to compute in practice, Altschul et al., 1997 [5] and Zhang et al., 1998 [32] used some heuristics for searching databases with this approach. Other attempts to fix the problem of mosaic effect involve modifications of the local alignment algorithm that allow insertions of very long gaps.

Another deficiency of the local alignment was recently revealed by Alexandrov and Solovyev, 1998 [2]. They asked if the Smith-Waterman algorithm correctly finds the most biologically adequate relative in a benchmark sample of different protein families. The answer to this question was negative, and Alexandrov and Solovyev, 1998 [2] "blamed" it on the fact that the Smith-Waterman algorithm does not take into account the length of the alignment. They proposed to normalize the alignment score by its length and demonstrated that this new approach leads to better protein classification. However, computing normalized scores in alignments may be very expensive when there is a constraint on length.

The idea of normalization has been studied in the context of *edit distances* where the objective is a minimization defined over the set of sequence of edit operations transcribing one string to the other of the two given strings. We may think of adapting similar solutions to normalized local alignment problem where the objective is to reveal local similarities by maximizing the scores among the substrings of the original strings. The algorithm developed by Marzal and Vidal, 1993 [19] computes the *normalized edit distance* between two given strings. The normalized edit distance problem seeks for a sequence of edit operations with minimum amortized weight, i.e. the total weight divided by the number of edit operations. The algorithm in [19] uses dynamic programming to compute the minimum edit distances for all lengths. Similarly, we can modify the Smith-Waterman local alignment algorithm ([27]) to consider local scores of all lengths which requires cubic time and quadratic space, very high complexities for practical applications. Various parallel algorithms for normalized edit distance problem were developed by Eğecioğlu and Ibel, 1996 [15]. In this paper, we only consider serial computations. The algorithms by Oommen and Zhang, 1996 [21], Vidal et al., 1995 [29], Arslan and Eğecioğlu, 1999 [7], 2000 [8] do not aim to satisfy a constraint on the length, therefore they cannot directly be adapted to the computation of normalized scores when lengths are restricted.

In this paper, we propose a new practical algorithm that produces local alignment with maximum degree of similarity by extending the ideas presented in [7] and [8]. To reflect the length of the local alignment in scoring, the score $s(I,J)$ of local alignment involving substrings I and J may be adjusted by dividing $s(I,J)$ by the total length of the aligned regions: $s(I,J)/(|I|+|J|)$. The *normalized local alignment problem* is to find substrings I and J that maximize $s(I,J)/(|I|+|J|)$ among all substrings I and J with $|I|+|J| \geq T$, where T is a threshold for the minimal overall length of I and J. For the same problem with no restriction on overall length, the answer would be short substrings that are not biologically meaningful (in this case normalized score is maximized by a single match). We use a slightly different objective to normalized alignment. We aim to maximize $s(I,J)/(|I|+|J|+L)$ for a given parameter L . Our purpose is to provide a way of control over the degree of normalization by varying L, and at the same time still being able to use fractional programming technique for fast computation.

2. NORMALIZED LOCAL ALIGNMENT ALGORITHMS

Let $a = a_1 a_2 \cdots a_n$ and $b = b_1 b_2 \cdots b_m$ be two sequences of symbols over an alphabet Σ with $n \geq m$. The *Alignment Graph* $G_{a,b}$ (*Edit Graph* in the context of string editing) is used to represent all possible *alignments* (Waterman, 1995 [30]) between a and b . It is a directed acyclic graph having $(n+1)(m+1)$ lattice points (u,v) for $0 \leq u \leq n$, and $0 \leq v \leq m$ as vertices (Figure 2). The diagonal arcs are either matching ($a_u = b_v$), or mismatching ($a_u \neq b_v$).

Consider a directed path p between two vertices $(i-1, j-1)$ and (k,l) on $G_{a,b}$ where $i \leq k$ and $j \leq l$.

Figure 1: The inclusion of an arbitrarily poor region in an alignment (Zhang et al., 1999). If a region of negative score $-X$ is sandwiched between two regions scoring more than X, then the Smith-Waterman algorithm will join the three regions into a single alignment that may not be biologically adequate.

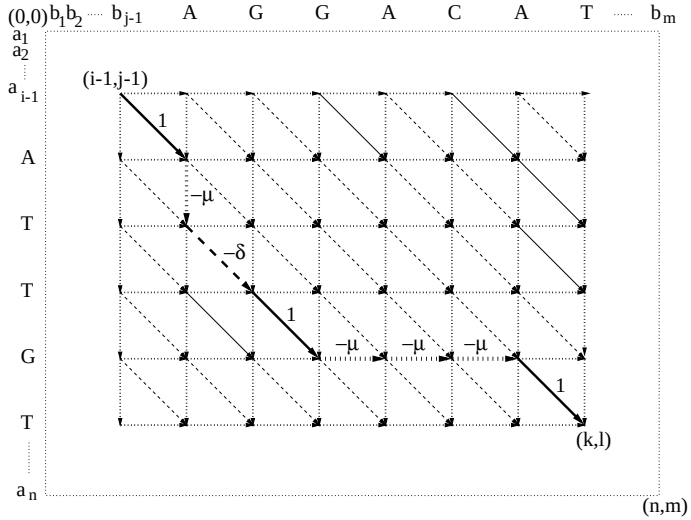

Figure 2: The alignment graph $G_{a,b}$ where $a_i \cdots a_k = ATTGT$ and $b_j \cdots b_l = AGGACAT$. Matching diagonal arcs are drawn as solid lines while mismatching diagonal arcs are shown by dashed lines. Dotted lines used for horizontal and vertical arcs correspond to indels. An example alignment path is shown. Only the weights of the arcs in this path are included.

We call each such path an *alignment path* since tracing the arcs of p, and performing the corresponding edit operations in $a_i \cdots a_k$, we obtain the segment $b_j \cdots b_l$ as follows : for a horizontal arc $((u, v-1),(u, v))$, insert b_v immediately before a_u; for a vertical arc $((u-1, v),(u, v))$ delete a_u; for a mismatching diagonal arc $((u-1, v-1),(u, v))$, substitute b_v for a_u. In the context of sequence alignment, insertions (horizontal arcs) and deletions (vertical arcs) are both called *indel*s, and the names *match*, and *mismatch*, are used to refer to matching diagonal, and mismatching diagonal arcs.

The objective of sequence alignment is to quantify the similarity between two strings. There are various scoring schemes for this purpose. In one simple such method, the arcs of $G_{a,b}$ have weights determined by positive reals δ (*mismatch penalty*) and μ (*indel* or *gap penalty*) as shown in Figure 2. We assume that a match has a score of 1, a mismatch penalty is δ, and an indel has a penalty of μ. Existence of an alignment path with a large total weight between the vertices $(i-1, j-1)$ and (k, l) indicates a high similarity between the segments $a_i \cdots a_k$ and $b_j \cdots b_l$.

For clarity of exposition, we assume this simple scoring scheme in setting up the definitions. We address the issue of extending the results to more complex scoring schemes in the next section.

We say that (x, y, z) is an *alignment vector for* $a_i \cdots a_k$ *and* $b_j \cdots b_l$, if there is an alignment path between the vertices $(i-1, j-1)$ and (k, l) in $G_{a,b}$ with x matches, y mismatches, and z indels. In Figure 2, $(3, 1, 4)$ is an alignment vector corresponding to the path shown in the figure. Let $AV_{i,j,k,l}(a, b)$ denote the set of all such alignment vectors, i.e. $AV_{i,j,k,l}(a, b) = \{(x, y, z) \mid (x, y, z)$ is an alignment vector for $a_i \cdots a_k$ and $b_j \cdots b_l\}$.

Similarly we call (x, y, z) an *alignment vector* if it is an alignment vector for some pair $a_i \cdots a_k$ and $b_j \cdots b_l$. We define $AV(a, b)$ as the set of all alignment vectors, over all $i \leq k$ and $j \leq l$. An alignment vector (x, y, z) has a score defined by δ, and μ :

$$SCORE(x, y, z) = x - \delta y - \mu z \qquad (1)$$

The maximum score between segments $a_i \cdots a_k$ and $b_j \cdots b_l$ is the score of an alignment vector whose score is the maximum among all the alignment vectors between these two sequences.

$$S_{\delta,\mu}(a_i \cdots a_k, b_j \cdots b_k) = \max\{SCORE(x, y, z) \mid (x, y, z) \in AV_{i,j,k,l}(a, b)\} \qquad (2)$$

In this paper, we denote by $\mathcal{P}^*$ the optimum value of problem $\mathcal{P}$. Local Alignment (LA) problem seeks for two segments with the highest similarity score:

$$LA^*_{\delta,\mu}(a,b) = \max_{\substack{i \le k,\\ j \le l}} \{S_{\delta,\mu}(a_i \cdots a_k, b_j \cdots b_k)\} \quad (3)$$

Let $LENGTH_L(a_i \cdots a_k, b_j \cdots b_l) = (k-i+1) + (l-j+1) + L$ where L is a positive constant. A normalized score NS_L of two segments $a_i \cdots a_k$, $b_j \cdots b_l$ is then

$$NS_{\delta,\mu,L}(a_i \cdots a_k, b_j \cdots b_l) = \frac{S_{\delta,\mu}(a_i \cdots a_k, b_j \cdots b_l)}{LENGTH_L(a_i \cdots a_k, b_j \cdots b_l)} \quad (4)$$

Normalized Local Alignment (NLA) problem seeks for two segments $a_i \cdots a_k$ and $b_j \cdots b_l$ for which the normalized score is the highest among all possible pairs of segments as expressed below:

$$NLA^*_{\delta,\mu,L}(a,b) = \max_{\substack{i \le k,\\ j \le l}} \{NS_{\delta,\mu,L}(a_i \cdots a_k, b_j \cdots b_l)\}$$

Observe that if (x,y,z) is an alignment vector for $a_i \ldots a_k$ and $b_j \ldots b_l$ then $(k-i+1)+(l-j+1) = 2x+2y+z$. Using this relation, we see that the function $LENGTH_L$ can be given on the set of alignment vectors $(x,y,z) \in AV(a,b)$ by the expression

$$LENGTH_L(x,y,z) = 2x + 2y + z + L \quad (5)$$

By using the definition of $AV(a,b)$ with (1), (2), (4), and (5) we express the objective of the NLA problem in the domain of alignment vectors as

$$NLA^*_{\delta,\mu,L}(a,b) = \max\left\{\frac{SCORE_{(x,y,z)}}{LENGTH_L(x,y,z)} \mid (x,y,z) \in AV(a,b)\right\} \quad (6)$$

Figure 3 shows some possible problem cases for LA for which NLA discriminates an alignment with higher percent matches from the one determined by the LA problem. Part (i) includes an example for the mosaic effect, and parts (ii), and (iii) have examples with non-overlapping and overlapping alignments respectively. For $L < 600$, in each case, the shorter alignment(s) with a score of 80 has a larger normalized score ($\frac{80}{200+L}$) than the longer alignment which has a score of 120 (whose normalized score is $\frac{120}{600+L}$) .

The local and normalized alignment problems we have defined by stating their objectives are clearly optimization problems of linear functions over the same domain. In other words, using equations (1) and (5), and definitions (3) and (6) we can rewrite LA and NLA as the following maximization problems :

$$\begin{aligned} LA_{\delta,\mu}(a,b): &\quad maximize\ x - \delta y - \mu z \\ &\quad \text{s.t.}(x,y,z) \in AV(a,b) \\ NLA_{\delta,\mu,L}(a,b): &\quad maximize\ \tfrac{x-\delta y-\mu z}{2x+2y+z+L} \\ &\quad \text{s.t.}(x,y,z) \in AV(a,b) \end{aligned}$$

For a given λ, we define a problem which we call *the parametric local alignment problem*

$$\begin{aligned} &LA_{\delta,\mu,L}(\lambda)(a,b): \\ &maximize\ x - \delta y - \mu z\ -\ \lambda(2x+2y+z+L) \\ &\text{s.t. } (x,y,z) \in AV(a,b) \end{aligned}$$

In order not to repeat the formal parameters in the problem descriptions when they are the same, in the rest of the paper we will use LA, NLA and $LA(\lambda)$ instead of $LA_{\delta,\mu,L}(a,b)$, $NLA_{\delta,\mu,L}(a,b)$, and $LA_{\delta,\mu,L}(\lambda)(a,b)$, respectively.

As we propose next, a parametric local alignment problem can be described in terms of local alignment problem.

PROPOSITION 1. *For a parameter λ ($< \frac{1}{2}$), the optimum value $LA^*(\lambda)$ of the parametric LA problem can be formulated in terms of the optimum value LA^* of an LA problem.*

PROOF. The objective of the parametric problem is

$$\begin{aligned} & LA^*(\lambda) \\ = \ & \max\{(1-2\lambda)x - (\delta+2\lambda)y - (\mu+\lambda)z - \lambda L\} \\ = \ & (1-2\lambda)\max\left\{x - \tfrac{\delta+2\lambda}{1-2\lambda}y - \tfrac{\mu+\lambda}{1-2\lambda}z\right\} - \lambda L \\ = \ & (1-2\lambda)LA^*_{\delta',\mu'}(a,b) - \lambda L \end{aligned}$$

where $\delta' = \frac{\delta+2\lambda}{1-2\lambda}$, $\mu' = \frac{\mu+\lambda}{1-2\lambda}$.

Thus, computing $LA^*(\lambda)$ involves solving the local alignment problem $LA_{\delta',\mu'}(a,b)$, and performing some simple arithmetic afterwards. □

Note that since δ, μ and L are positive, for any alignment vector (x',y',z'), if λ is its normalized score then

$$\lambda = \frac{x' - \delta y' - \mu z'}{2x' + 2y' + z' + L} < \tfrac{1}{2}$$

Dinkelbach's algorithm [14] can be used to solve NLA . Dinkelbach has developed a general algorithm which uses the *parametric method* of an optimization technique

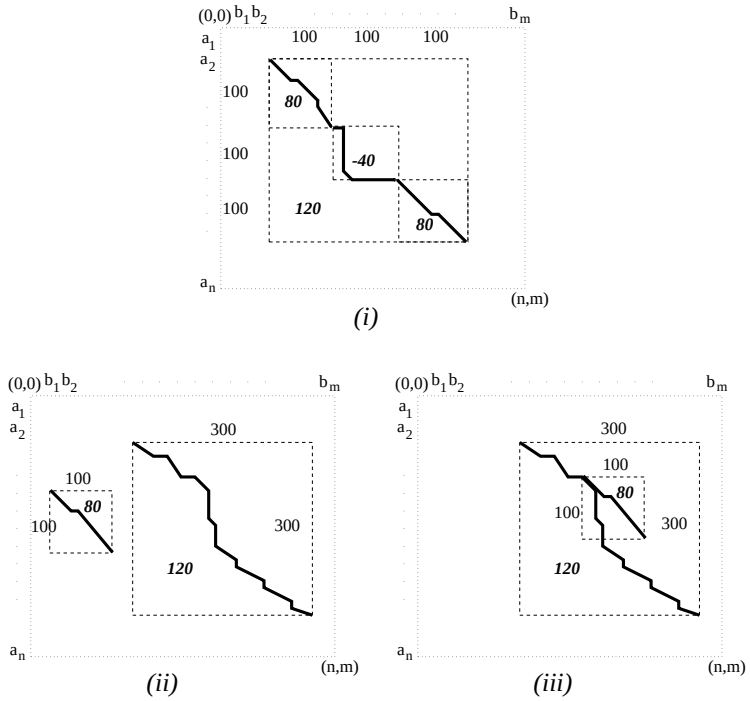

Figure 3: Mosaic and shadow effects. (i) **mosaic effect,** (ii) **shadow effect (non-overlapping alignments),** (iii) **shadow effect (overlapping alignments). The numbers written in italic are the scores of alignments identified by the corresponding rectangles. The other numbers are the side lengths of the rectangles.**

known as *fractional programming*. The algorithm is applicable to optimization problems which involve a ratio of two functions over the same domain where the function in the denominator is assumed to be positive. The thesis of the parametric method applied to the case of alignment maximization problems implies that the optimal solution to NLA can be achieved via a series of optimal solutions of $LA(\lambda)$ for different λ . The central result is that

$$\lambda = NLA^* \quad \text{iff} \quad LA^*(\lambda) = 0 \ .$$

That is, an alignment vector a has the optimum normalized score λ iff a is an optimal alignment vector for the parametric problem $LA(\lambda)$ whose optimum value is zero. A proof of this essential property of the parametric method is given by Sniedovich, 1992 [28]. Craven, 1988 [13] and Sniedovich, 1992 [28] explain various other interesting properties of Dinkelbach's algorithm and fractional programming.

Dinkelbach algorithm for NLA problem is shown in Figure 4. The algorithm starts with an initial value for λ and repeatedly solves $LA(\lambda)$. At each instance of the parametric problem, an optimal alignment vector (x, y, z) of $LA(\lambda)$ yields a ratio (normalized score) for NLA. This new ratio is either equal to λ, in which case it is optimum, or larger than λ . If it is equal to λ then the algorithm terminates. Note that in this case $LA^*(\lambda) = 0$ since the optimal alignment vector of the last iteration has the normalized score λ . Otherwise, the ratio is taken to be the new value of λ and $LA(\lambda)$ is solved again. When continued in this fashion, convergence to NLA^* is guaranteed. Another way to explain the behavior of the algorithm is as follows. It iteratively modifies the scores in such a way that the optimal non-normalized local alignment under the set of converged scores is also the optimal normalized alignment under the original scores.

```
Algorithm Dinkelbach

Pick an arbitrary alignment vector (x, y, z) in
AV(a, b) ,
                λ* ← (x − δy − μz) / (2x + 2y + z + L)
Repeat
   λ ← λ*
   Using Prop.1, solve LA(λ) and
   obtain an optimal alignment vector (x, y, z)
   λ* ← (x − δy − μz) / (2x + 2y + z + L)
Until λ* = λ

Return(λ*)
```

Figure 4: `Dinkelbach` **algorithm for** ***NLA*.**

The parametric problem in this algorithm can be solved using the Smith-Waterman algorithm. An optimal alignment vector (or alternatively its score and length values) needs to be computed along with optimal score for the parametric problem of the Dinkelbach algorithm. Position of an optimal alignment may also be desired. These can be done by extending the Smith-Waterman algorithm to include, at each entry of the score matrix, information about the alignment vector corresponding to an optimal alignment path which ends at that node, and the starting node-position of the path. This additional information can be carried over and updated along with the optimal score updates without an increase in the asymptotic space and time complexity. The resulting space complexity of solving *NLA* by this algorithm is $O(m)$. The resulting time complexity is the product of the number of iterations and, the time complexity of the Smith-Waterman algorithm. Although experimental results suggest that the number of iterations is small on average, no satisfactory theoretical average-case/worst-case bound for the growth of the number of iterations has been established.

We show next that a provably better time complexity result can be achieved by using Megiddo's technique based on an observation used in Arslan and Eğecioğlu, 2000 [8] for the computation of normalized edit distance. Even though it does not seem feasible to precompute candidate values for the optimum value of *NLA*, we can show that an efficient search (*a binary search*) for the optimum value is still possible by using the fact that any two distinct candidate values for NLA^* are not arbitrarily close to each other if the scores are rational. The resulting algorithm `RationalNLA` for the *NLA* problem with rational penalties is given in Figure 5. The properties of `RationalNLA` can be used to prove the following theorem whose proof is omitted.

THEOREM 1. *If algorithm A computes LA^* and obtains an optimal alignment vector with time complexity $T(n,m)$, then* NLA* *can be computed in time $O(T(n,m)\log n)$ and using (asymptotically) the same space required by algorithm A provided that δ and μ are rational.*

The Smith-Waterman algorithm can be used as algorithm A in `RationalNLA` to find the local alignment vectors and hence to solve the parametric local alignment problems invoked by `RationalNLA`. Therefore:

COROLLARY 1. *Normalized local alignment of sequences of length n and m can be computed in $O(nm\log n)$ time and $O(m)$ space.*

The ideas in the Dinkelbach algorithm or algorithm `RationalNLA` are not restricted to a particular scoring scheme. Under any given scoring scheme, provided that the parametric LA problems in these algorithms can be formulated in terms of an LA problem, these algorithms

```
Algorithm RationalNLA

σ ← 1/(qs(m+n+L)^2) where δ = p/q, and μ = r/s (This is the
gap lower bound)
[e, f] ← [0, (1/2)qs(m + n + L)^2]

While (e + 1 < f) do
  k ← ⌊(e + f)/2⌋
  Using Prop.1, solve LA(kσ)
   and let v be the optimum score obtained
  if v = 0 then return(kσ)
  else if v < 0 then f ← k
  else e ← k
End {while}

Using Prop.1, solve LA(fσ) and obtain an optimal
alignment vector (x, y, z)

Return((x − δy − μz)/(2x+2y+z+L))
```

Figure 5: *NLA* algorithm `RationalNLA` for rational scores.

can be modified so that they present a solution to *NLA* problem. Furthermore, if scores/penalties are rational, and solving a parametric problem and obtaining an optimal solution (alignment vector) take asymptotically the same time as that of the underlying LA algorithm, then the complexity results for `RationalNLA` of Theorem 1 hold. We address two particularly important cases of scoring schemes : *affine gap penalties*, and *arbitrary score matrices*.

Sometimes insertion or deletion of a block of symbols called a *gap* is treated differently than a stream of single-symbol indels. Affine gap penalty for a gap of length k is

$$\alpha + \mu k$$

where α is a *gap open penalty* and μ is an indel penalty. In this case, we may use a 4-tuple (x, y, z, g) to represent an alignment vector with which the new component g is the number of gaps. For example, $(3, 1, 4, 2)$ is the alignment vector for the alignment path shown in Figure 2. The alignment vector has two gaps one of which is a single delete, and the other is a block of three inserts. The definition of the length function $LENGTH_L$ does not change under this scoring scheme. The score of an alignment vector can be rewritten as

$$SCORE(x, y, z, g) = x - \delta y - \mu z - \alpha g$$

In some applications, score of a given operation varies depending on the individual symbols involved in the operation (e.g., protein sequence comparison). In this case, we may decide to define the alignment vector such that it includes as a component frequency of each operation. Let $i-$, $-i$ denote respectively the deletion and insertion of the ith symbol, and ij denote the substi-

tution of the jth symbol for the ith symbol of the alphabet Σ . For a given operation e, let s_e represent the score, and f_e represent the frequency of this operation. If $u = |\Sigma|$ then for a given alignment vector a where

$$\begin{aligned} a \;=\; < \;& f_{1-}, f_{2-}, \ldots, f_{u-}, \\ & f_{-1}, f_{-2}, \ldots, f_{-u}, \\ & f_{11}, f_{12}, \ldots, f_{1u}, \ldots, f_{u1}, f_{u2}, \ldots, f_{uu} \;> \;, \end{aligned}$$

the score and length functions can be defined as

$$SCORE(a) \;=\; \sum_{ij} s_{ij} f_{ij} + \sum_{i} s_{i-} f_{i-} + \sum_{i} s_{-i} f_{-i}$$

$$LENGTH_L(a) \;=\; 2\sum_{ij} f_{ij} + \sum_{i} f_{i-} + \sum_{i} f_{-i} + L$$

One can verify that in both of these cases, a parametric LA problem can easily be formulated in terms of an LA problem under that particular scoring scheme, and our results hold.

3. IMPLEMENTATION AND TEST RESULTS

We have chosen to implement the Dinkelbach algorithm for NLA computation (affine gap penalties) since this algorithm has a good performance in practice. We have modified the Smith-Waterman algorithm (for affine gaps) to obtain and carry along the alignment information through the nodes. In our implementation we have used $LENGTH_L$ value of the alignment vectors as a tie breaker. We select an alignment with the largest $LENGTH_L$ value in case there are more than one optimal alignments ending in the same node. That is, we favor the alignment with the largest $LENGTH_L$ value among the alignments with the same normalized score since for two alignments with the same normalized score, the one with larger $LENGTH_L$ value has the higher (non-normalized) score which may be preferred over others (The program can be obtained by contacting A.N.A.). In our tests, the algorithm never required more than 9 invocations of the Smith-Waterman algorithm, and in the majority of cases it took $3-5$ invocations to solve a single NLA problem.

Once optimal segments are found for one NLA problem, one may want to continue with more NLA computations after masking these segments in the two sequences. For this purpose, we have developed algorithm `RepeatedDinkelbach`. With each alignment between $a_i \ldots a_k$ and $b_j \ldots b_l$, we store a pair whose first component is the alignment vector (x, y, z, g) and second component is the alignment position (i, j, k, l) . We have used a queue Q to store alignments generated by the iterations of the Dinkelbach NLA algorithm so that a new NLA computation picks as the initial alignment the last alignment in Q which does not overlap with the alignment reported in the last iteration. This way we improve the average number of iterations per NLA computation. `RepeatedDinkelbach` continues generating alignments until no alignment whose normalized score is larger than a given threshold score T can be found in unmasked regions of the sequences. This termination condition is easy to implement since the normalized scores are decreasing as they are reported. Another alternative would be to let the algorithm run until there remains no more alignments with positive score. We have also implemented a version of the algorithm which first masks a set of regions as a pre-processing step. This allows us to explicitly stop the NLA computations at any time we want, and resume the computation of alignments from where it (almost) left using the second algorithm.

We have tested our algorithms with various values of L . We observe that if L is large we obtain alignments with high scores but low normalized scores, while if L is small then the resulting alignments have high normalized scores but they may be short and less interesting biologically. In other words, as the value of L increases our algorithm finds longer optimal alignments for a particular instance of the problem. It is difficult to determine a value for L which performs well in (almost) every case because a proper value is data-dependent. If the highest normalized score (with respect to the current value of L) belongs to an alignment that is too short to be biologically interesting then we need to increase the value of L to favor the longer (biologically interesting) alignments. For example for the alignments in Figure 3, L has to be at least 600 so that the longer alignment wins over the shorter one. If alignments returned as optimal do not have sufficiently high normalized scores then a smaller values of L should be tried. One needs to experiment various values for L for a particular instance of sequence alignment. Another way to get rid of unwanted short alignments can be to mask the corresponding regions and rerun the algorithm. If we decide to do so we need to be sure that these regions do not take part in desired alignments. As a common practice in sequence alignment, we first masked the repeats by RepeatMasker (http://ftp.genome.washington.edu/ RM/RepeatMasker.html) before running our algorithm. These biologically uninteresting regions may have high normalized scores. They may become part of unwanted short alignments. Therefore hiding repeats may help eliminate short alignments to be output as optimal by our algorithm. To visualize the difference among various approaches to sequence alignment, we represented every area of similarity as a rectangle rather than as a diagonal in conventional drawings of dot-matrices. Rectangles in the figures show the segments involved in the alignments. In Figures 6 and 7 the alignment regions returned by Smith-Waterman algorithm are shown using dotted lines whereas those determined by post-processing algorithm by Zhang et al., 1999 [33] are distinguished by dashed lines. Rectangles with thick lines are the ones obtained by our algorithm. We have included percent matches (number of matches divided by the average length of the segments) for the alignments we have found. Our algorithm captures the regions found

by these algorithms but provides more "granularity" in representing the most similar fragments of the aligned regions. To achieve even higher level of granularity one can either reduce the threshold T for reported alignments or vary L at different iterations of the algorithm. As expected, the regions not included in found normalized local alignments show little similarity: the degree of similarity "outside" the boxes in Figures 6 and 7 is usually below 35%.

4. CONCLUSIONS

The arrival of long genomic sequences raises new challenges in sequence comparison. In particular, the traditional tools for computing and representing alignments may not be suitable for genomic-scale sequence comparison. These challenges were recently addressed by Schwartz et al, 2000 [24] who introduced the *Percent Identity Plots* or *PIPs*. PIPs are compact and convenient substitutes for dot-matrices that, in addition to revealing similar segments, reflect the percent of similarity between different segments of compared sequences. Our normalized local approach is conceptually similar to this approach in an attempt to find the regions with the highest percent of similarity.

The undesirable properties of linear scoring in sequence alignment were first revealed by Altschul and Erickson, 1986 [3] who proposed different non-linear scoring functions. They also noticed that alignments with non-linear scoring functions are difficult to compute in practice. The deficiency of linear scoring functions are well-known in other application domains of dynamic programming. In particular, non-linear scoring functions lead to better practical algorithms for speech recognition and recognition of hand-written texts (Vidal et al., 1995 [29]).

Some sequence comparison practitioners have been using a few runs of the Smith-Waterman algorithm with varied gap penalties to arrive to a biologically adequate alignment. However, the choice of gap penalties in such searches remained largely heuristic. Our algorithm for normalized sequence alignment mimics this approach but provides a rigorous justification for choosing parameters in different runs of the Smith-Waterman algorithm.

Pearson, 1995 [23], Shpaer et al., 1996 [26] and Brenner et al.,1998 Brenner98 made the comparative analysis of FASTA, BLAST and the Smith-Waterman algorithm for functional protein classification. Abdueva et al. 2001 [1] used their test framework to study the effect of alignment length on sensitivity of database search. The preliminary results of this work demonstrate that normalization improves the functional protein classification.

Although the normalized local alignment approach proved to be successful in our preliminary tests, a number of questions remain unsolved. Most importantly, the statistics of normalized local alignment is poorly understood. The statistical questions associated with the classical local alignment are so complex (Arratia et al., 1990 [6], Waterman and Vingron, 1994 [31]) that we did not even dare to try estimating statistical significance of normalized local alignment. Another problem is that the rules governing the optimal choice of the parameter L are not yet well understood.

5. ACKNOWLEDGEMENTS

We are grateful to Diana Abdueva, Nickolai Alexandrov, Steven Altschul, Ben Koop, Martin Vingron, and Michael Waterman for helpful discussions.

6. REFERENCES

[1] D. Abdueva, V.V. Solovyev, M. Trukhan, P.A. Pevzner, and N. N. Alexandrov. Effect of alignment length on sensitivity of database search. *(in preparation).*

[2] N. N. Alexandrov and V. V. Solovyev. Statistical significance of ungapped alignments. *Pacific Symp. on Biocomputing (PSB-98)*, (eds. R. Altman, A. Dunker, L. Hunter, T. Klein), 463-472.

[3] S. F. Altschul and B. W. Erickson. Locally optimal subalignments using nonlinear similarity functions. *Bulletin of Mathematical Biology*, 48, 633-660, 1986.

[4] S. F. Altschul and B. W. Erickson. Significance levels for biological sequence comparison using nonlinear similarity functions. *Bulletin of Mathematical Biology*, 50, 77-92, 1988.

[5] S. F. Altschul, T.L. Madden, A.A. Schaffer, J. Zhang, Z. Zhang, W. Miller and D.J. Lipman. Gapped Blast and Psi-Blast: a new generation of protein database search programs. *Nucleic Acids Research*, 25, 3389-3402, 1997.

[6] R. Arratia and L. Gordon and M.S. Waterman. The Erdös-Renyi Law in distribution, for coin tossing and sequence matching. *The Annals of Statistics*, 18, 539-570, 1990.

[7] A. N. Arslan, Ö. Eğecioğlu. An Efficient Uniform-Cost Normalized Edit Distance Algorithm. *6th Symp. on String Processing and Info. Retrieval (SPIRE'99), IEEE Comp. Soc.*, 8-15, September 1999, Cancun, Mexico.

[8] A. N. Arslan, Ö. Eğecioğlu. Efficient Algorithms For Normalized Edit Distance. *Journal of Discrete Algorithms, Special Issue on Matching Patterns, Hermes Science Publications, (in press).*

[9] V. Bafna and D. Huson. The Conserved Exon Method for Gene Finding *Proc. of the Eight Int. Conf. on Intelligent Systems for Molecular Bio.*, La Jolla, California, 2000, 3-12

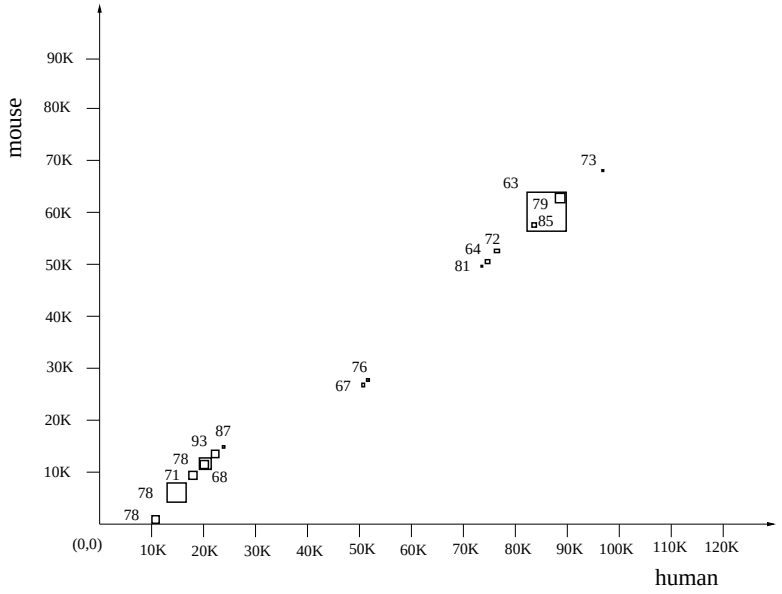

Figure 6: Normalized local alignments of orthologous human (GenBank Acc. No. AF030876) and mouse (GenBank Acc. No. AF121351) genomic sequences ($L = 2000$, $\delta = 1$, $\alpha = 6$, and $\mu = 0.2$).

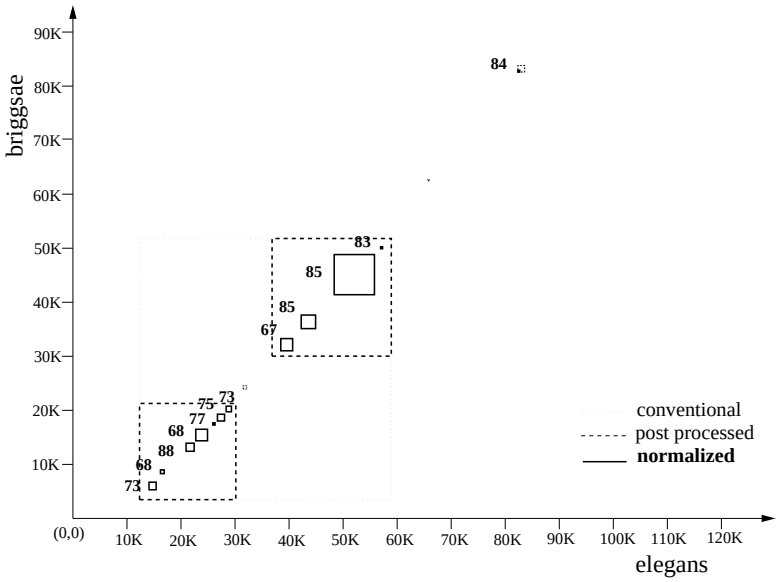

Figure 7: Comparison of normalized local alignments of *bli-4* locus in *C. elegans* and *C.briggsae* with conventional local alignments and post-processed local alignments as described in Zhang et al., 1999 ($L = 2000$, $\delta = 1$, $\alpha = 6$, and $\mu = 0.2$).

[10] S. Batzoglou and L. Pachter and J. Mesirov and B. Berger and E. Lander Comparative analysis of mouse and human DNA and applications to exon prediction. *Proc. of the Fourth Annual Int. Conf. on Computational Molecular Biology(RECOMB-99))*, April 2000, Tokyo, Japan, 46-53

[11] E. Birney and J.D. Thompson and T.J. Gibson PairWise and SearchWise: finding the optimal alignment in a simultaneous comparison of a protein profile against all DNA translation frames. *Nucleic Acids Res.*, 24, 2730-2739, 1996.

[12] S.E. Brenner and C. Chotia and T.J. Hubbard Assessing sequence comparison methods with reliable structurally identified distant evolutionary relationships. *Proc Natl Acad Sci USA*, 95, 6073-6078, 1998.

[13] B. D. Craven. *Fractional Programming.* Helderman Verlag, Berlin, 1988.

[14] W. Dinkelbach. On nonlinear fractional programming. *Management Science*, 13:492-498, 1967.

[15] Ö. Eğecioğlu and M. Ibel. Parallel algorithms for fast computation of normalized edit distances. *Proc. of the Eighth IEEE Symp. on Par. and Dist. Proc. (SPDP'96)*, 496-503, October 1996.

[16] M.S. Gelfand and A.A. Mironov and P.A. Pevzner. Gene recognition via spliced sequence alignment. *Proc. Natl. Acad. Sci. USA.* 93, 9061-9066, 1996.

[17] W. B. Goad and M. I. Kanehisa. Pattern recognition in nucleic acid sequences. I. A general method for finding local homologies and symmetries. *Nucleic Acid Research*, 10:247-163, 1982.

[18] X. Huang and P.A. Pevzner and W. Miller. Parametric recomputing in alignment graph. *Proc. of the 5th Annual Symp. on Comb. Pat. Matching*, 87-101, Asilomar, CA, June, 1994.

[19] A. Marzal and E. Vidal. Computation of normalized edit distances and applications. *IEEE Trans. on PAMI*, 15(9):926-932, September 1993.

[20] N. Megiddo. Combinatorial optimization with rational objective functions. *Mathematics of Operations Research*, 4:414-424, 1979.

[21] B. J. Oommen and K. Zhang. The normalized string editing problem revisited. *IEEE Trans. on PAMI*, 18(6):669-672, June 1996.

[22] L. Pachter and S. Batzoglou and V.I. Spitkovsky and E.S. Lander and B. Berger and D.J. Kleitman. A Dictionary Based Approach for Gene Annotation. *Proc. of the Third Annual Int. Conf. on Comp. Mol. Bio. (RECOMB-99))* , 285-194, April 1999, Lyon, France.

[23] W.R. Pearson. Comparison of methods for searching protein sequence databases. *Protein Sci.*, 4, 1145-1160, 1995.

[24] S. Schwartz and Z. Zhang and K. A. Fraser and A. Smit and C. Riemer and J. Bouck and R. Gibson and R. Hardisson and W. Miller. PipMaker - A Web server for aligning two genomic DNA sequences. *Genome Research*, 10, 577-586, 2000.

[25] P. H. Sellers. Pattern recognition in genetic sequences by mismatch density. *Bull. of Math. Bio.*, 46, 501-504, 1984.

[26] E.G. Shpaer, M. Robinson, D. Yee, J.D. Candlin, R. Mines, and T. Hunkapiller T. Sensitivity and selectivity in protein similarity searches: a comparison of Smith-Waterman in hardware to BLAST and FASTA. *Genomics*, 38, 179-191, 1996.

[27] T.F. Smith and J. M.S. Waterman. The identification of common molecular subsequences. *J. Mol. Biol.*, 147, 195-197, 1981.

[28] M. Sniedovich. *Dynamic Programming.* Marcel Dekker, New York, 1992.

[29] E. Vidal, A. Marzal, and P. Aibar. Fast computation of normalized edit distances. *IEEE Trans. on PAMI*, 17:899-902, 1995.

[30] M.S. Waterman. Introduction to Computational Biology. Chapman and Hall, 1995.

[31] M.S. Waterman and M. Vingron. Rapid and accurate estimates of statistical significance for sequence database searches *Proc. Natl. Acad. Sci. USA*, 91, 4625-4628, 1994.

[32] Z. Zhang and P. Berman and W. Miller. Alignments without low-scoring regions. *J. Comput. Biol.*, 5, 197-200, 1998.

[33] Z. Zhang and P. Berman and T. Wiehe and W. Miller. Post-processing long pairwise alignments. *Bioinformatics*, 15, 1012-1019, 1999.