Aligning Coding DNA in the Presence of Frame-Shift Errors

Lars Arvestad *

Department of Numerical Analysis and Computing Science
Royal Institute of Technology
S-100 44 Stockholm, Sweden.

Abstract. The problem of aligning two DNA sequences with respect to the fact that they are coding for proteins is discussed. Criteria for a good alignment of coding DNA, together with an algorithm that satisfies them, are presented. The algorithm is robust against frame-shifts and forgiving towards silent substitutions. The important choice of objective function is examined and several variants are proposed.

1 Introduction

In this paper we discuss how to align two DNA sequences that come from a coding region, i.e. the DNA is translated to an amino acid sequence, which is something we should take note of when aligning the sequences.

The traditional pairwise sequence alignment algorithm, as found by Sellers in 1974 [13] and independently by others (see Waterman [15] and Sankoff and Kruskal [12] for a thorough treatment), aims at minimizing the amount of *change* (substitutions/replacements, insertions and deletions) between two biological sequences. However, change in DNA does not always have an obvious interpretation.

If the sequences are DNA coding for proteins, we do not necessarily want to count silent substitutions that are often numerous in the third position of a codon. Also, some amino acids are coded by codons that differ in each position. Matching two such codons looks bad on the DNA level, but should not result in a poor score since the proteins may be identical when the DNA is translated to amino acids.

Another common problem [9] with implementations of the traditional aligning algorithm, when applied to DNA, is that stretches of insertions/deletions, gaps, are not constrained to biologically reasonable lengths. Since gaps of a length that is not 0 modulo 3 change the reading-frame, they are very unlikely to occur. If the algorithms do not take this into account, unsatisfactory alignments are computed.

Frame-shift errors further complicate our task at hand. They typically occur from bad gel-readings (compressions) or other sequencing problems. Thus,

* E-mail: arve@nada.kth.se.

aligning newly sequenced DNA often involves investigating the source of error, correcting, and computing a new alignment. It would be desirable to do this automatically. Not only off-the-gel sequences contain frame-shift errors; it has been observed [8] that many sequences found in databases (e.g. EMBL) are faulty. An algorithm that is robust against these problems is therefore useful to many researchers.

Frame-shift errors also invalidate a natural line of attack, namely to translate the DNA to amino acids for each combination of reading frames. Because the translation will soon be obscured by a frame-shift, such an algorithm is very sensitive.

A related problem, the question of how to align a coding DNA sequence with a protein, has been discussed in a range of papers [4, 8, 11, 14] and mentioned applications are database searching as well as error-checking sequences. The problem we focus on in this paper has been addressed by Hein in [5], where an $O(N^4)$ algorithm is given. In this paper, we present a quadratic time algorithm for aligning protein-coding DNA in the presence of frame-shift errors.

Section 2 introduces notation and requirements for a good objective function. The new algorithm is explained in section 3 and variation on the objective function is found in section 4. To conclude, experimental results and a discussion are given in sections 5 and 6.

2 A new scoring scheme

In this section we address the question of how to choose a better scoring scheme for coding DNA sequences. Our intention is to introduce language needed for the new approach and give an abstract scoring scheme in order to present the new algorithm. Especially, notice that we do not yet make adjustments for frame-shifts, but leave that for section 3.

2.1 Definitions and notation

Definition 1. A *frame-shift* is an insertion or deletion of length one or two.

Definition 2. A *gap* is an insertion or deletion whose length is a multiple of three.

If a gap has a length that is not 0 modulo 3, then we regard it as a combination of a frame-shift and a gap. The intuition is that a frame-shift corresponds to something less likely such as a sequencing error or a rare evolutionary event, while a gap typically corresponds to lost or inserted amino acids.

The terms “cost” and “score” are used intermixed in this paper. We want to minimize the cost but maximize the score. Higher scores are assumed to be used for preferred matches.

Denote the cost of inserting a frame-shift γ . In this paper we only consider affine gap costs where opening a gap is associated with a cost α and every triplet

of three indels has an additional cost β . So for a gap of length l the cost is $\alpha + l\beta$. We can now make a new definition, slightly different from current practice, of an alignment.

Definition 3. An alignment of two sequences a and b is a pair (a', b') where a' and b' are derived by inserting frame-shifts and gaps in a and b .

Let $\mathcal{A}_{a,b}$ be the set of alignments of sequences a and b . (This notation is also abused to allow a and b to be sets of sequences.) The quality of an alignment is captured in the score of the alignment, thus we need to produce a reasonable scoring function. Let $\mathbf{N} = \{A, C, G, T\}$, the set of nucleotides and $\mathbf{N}^+ = \mathbf{N} \cup \{-\}$, the set of nucleotides together with the indel symbol. Similarly, let $\mathbf{A}$ be the set of amino acids and let $\mathbf{A}^+ = \mathbf{A} \cup \{-\}$ be the same set with indels included. The set of r -tuples over a set X is written X^r .

Definition 4. A function $S_N : (\mathbf{N}^+ \times \mathbf{N}^+)^* \mapsto \mathcal{R}$ is called a *nucleotide scoring function* and $S_A : (\mathbf{A}^+ \times \mathbf{A}^+)^* \mapsto \mathcal{R}$ is an *amino acid scoring function*.

We write the translation of a DNA sequence x , whose length is a multiple of three, to amino acids as $aa(x)$. If x contains frame-shifts or ambiguity symbols, the result may be a set of translations. Writing $aa(a, b)$ is short for $(aa(a), aa(b))$.

2.2 Requirements for a good alignment

Since we want to align coding DNA, we argue that a nucleotide scoring function should have the property that for an alignment (a', b') of a and b ,

$$S_N(a', b') = \max_{(\tilde{a}, \tilde{b}) \in \mathcal{A}_{a,b}} (S_N(\tilde{a}, \tilde{b})) \iff S_A(aa(a', b')) = \max_{(\tilde{a}, \tilde{b}) \in \mathcal{A}_{aa(a,b)}} (S_A(\tilde{a}, \tilde{b})). \quad (1)$$

That is, the optimal nucleotide sequence alignment is also, when translated to amino acids, the optimal amino acid sequence alignment, and *vice versa*.

We want the scoring function to work with codons, so let

$$G = \{(x, y) : x, y \in \mathbf{N}^3 \cup \{---\}\}$$

be the set of codon matchings and introduce the following to make S_N easier to define in terms of codons.

Definition 5. A function $s : G \mapsto \mathcal{R}$ is called a *codon scoring function*.

In words, s takes two triplets and assigns the matching of them a score. For equation (1) to be fulfilled, we choose s such that $s(x_1, x_2) = s_A(aa(x_1, x_2))$, where s_A is an amino acid scoring scheme, e.g. from PAM or BLOSSUM matrices, $s(---, x) = s(x, ---) = \beta$. Matchings of type gap-gap, $(---, ---)$, are not of interest to us and hence they have an associated score of $-\infty$.

We can now define an appropriate nucleotide scoring function.

Definition 6. Let x be the number of gaps. The score of an alignment (a', b') is then

$$S_N(a', b') = x\alpha + \sum_{i=0}^{|a'|/3} s(a'_{3i}a'_{3i+1}a'_{3i+2}, b'_{3i}b'_{3i+1}b'_{3i+2}). \quad (2)$$

This function is easy to optimize, and the only things that differ from traditional nucleotide alignment are that we are inserting indel triplets (instead of single indels) and assigning scores based on triplets.

3 Aligning in the presence of frame-shifts

The above scoring function only works when we have correctly sequenced sequences and when we disregard evolutionary relations that have come from accidental frame-shifts during evolution. For maximal flexibility, we want to be able to insert and delete frame-shifts. Inserting frame-shifts is simply a matter of inserting indels and deleting frame-shifts is to be interpreted as ignoring nucleotides.

Therefore, we change the definition of an alignment slightly.

Definition 7. An alignment of two sequences a and b is a pair (a', b') where a' and b' are derived by inserting and removing frame-shifts and adding gaps in a and b .

Observe that removing frame-shifts does not necessarily mean that nucleotides are removed from the sequences or even from the presentation of the alignment, but simply that they are ignored in the objective function.

Adapting to this definition, define G' as

$$G' = \{(x, y) : x, y \in \mathbb{N}^{+3}\}.$$

In this set of matchings, we have all possible ways of constructing codons from full nucleotide triplets as well as from codon fragments and frame-shift symbols. Also included are all matchings against gap symbols.

It is noteworthy that contrary to common practice, columns in our alignments may actually contain only indels. The approach taken here is that the sequences are tried to be reconstructed in parts where the reading-frame is confused. This is more natural when frame-shifts are thought to be sequence errors, than when they are evolutionary events. However, since we may look at alignments as tools for reconstructing sequence ancestors, frame-shifts have the advantage that we are able to guess the dropped nucleotide(s).

There is now an immediate extension of the nucleotide scoring function.

Definition 8. Let x and z be the number of gaps and frame-shifts, respectively. The score of an alignment (a', b') is then

$$S_N(a', b') = x\alpha + z\gamma + \sum_{i=0}^{|a'|/3} s(a'_{3i}a'_{3i+1}a'_{3i+2}, b'_{3i}b'_{3i+1}b'_{3i+2}).$$

In the former, how to choose s was quite immediate, but it is less clear how to do that now. We defer that discussion to section 4.

3.1 A new requirement

Requirement (1) is hard to relate to the current version of the problem since we had not defined a scoring function that could incorporate frame-shifts when it was formulated. The requirement can be restated to include frame-shifts in a slightly more complex way.

$$\begin{aligned} S_N(a', b') &= \max_{(\tilde{a}, \tilde{b}) \in \mathcal{A}_{a,b}} (S_N(\tilde{a}, \tilde{b})) \iff \\ &\iff S_A(aa(a', b')) = \max_{\substack{z \in \mathcal{N} \\ (\tilde{a}, \tilde{b}) \in \mathcal{A}_{aa(a,b,z)}}} (S_A(\tilde{a}, \tilde{b}) + z\gamma). \end{aligned} \quad (3)$$

We have added translations from the DNA pair to amino acid sequences which may include frame-shifts; The set $aa(a, b, z)$ is the set of translations of the DNA sequences a and b using z frame-shifts. The effect is that we recognize, both in the DNA alignment and in the DNA to amino acid translation, that our sequences might have frame-shifts.

3.2 The new algorithm

We can now present the new algorithm and we state it in a theorem.

Theorem 9. *We can find the optimal nucleotide alignment with frame-shifts in time $O(|a||b|)$.*

Proof. The result follows by applying generalized substitutions (see Sankoff-Kruskal [12]) and Gotoh's technique for linear gap functions [3]. Let $\mathbf{C} = \mathbb{N}^{1,2,3,4,5}$ be the set of codon fragments and let $M = \{(x, y) : x, y \in \mathbf{C}\}$ be the set of generalized substitutions. That is, M could be derived from G' by removing any gap-matchings (on the form $(---, x)$ for any x) and for each element in G' removing the indel symbols and create copies with one and two extra frame-shifting nucleotides inserted. Let $d_{a,b}$ denote the score of the optimal alignment for the sequences a and b ending with a (reconstructed) codon-matching. Let $d_{a,b}^-$ and $d_{a,b}^+$ denote the optimal score of the alignments on a and b that ends with a gap in a' and b' respectively.

To score substitutions from M , we define $\tilde{s}$ from s as

$$\tilde{s}(x, y) = \max_{\substack{x' \in I(x) \\ y' \in I(y)}} s(x', y') \quad (4)$$

where I is defined as

$$\begin{aligned} I(a) &= \{-a, -a-, a-\} & I(ab) &= \{-ab, a-b, ab-\} \\ I(abc) &= \{abc\} & I(abcd) &= \{abc, abd, acd, bcd\} \\ I(abcde) &= \{abc, abd, acd, bcd, abe, ace, bce, ade, bde, cde\} \end{aligned} \quad (5)$$

are the sets of three-letter strings, codons, created from inserting indels into, or removing nucleotides from, the arguments. We are assuming that s is defined on codons containing frame-shifts (with the associated cost γ included), s on a codon-gap matching scores β (unless the codon contains frame-shifts), and that $s(---, ---) = -\infty$.

Consider now the following recursion with $\circ$ as the concatenation operator.

$$d_{\epsilon, \epsilon} = d_{\epsilon, \epsilon}^- = d_{\epsilon, \epsilon}^+ = 0 \quad (6a)$$

$$d_{a,b} = \max_{\substack{\tilde{a} \circ x = a \\ b \circ y = b \\ (x,y) \in M}} \begin{cases} d_{\tilde{a}, \tilde{b}}^- + \tilde{s}(x, y) \\ d_{\tilde{a}, \tilde{b}} + \tilde{s}(x, y) \\ d_{\tilde{a}, \tilde{b}}^+ + \tilde{s}(x, y) \end{cases} \quad (6b)$$

$$d_{a,b}^- = \max_{\substack{b \circ y = b \\ y \in C}} \begin{cases} d_{a, \tilde{b}}^+ + \alpha + \tilde{s}(---, y) \\ d_{a, \tilde{b}} + \alpha + \tilde{s}(---, y) \\ d_{a, \tilde{b}}^- + \tilde{s}(---, y) \end{cases} \quad (6c)$$

$$d_{a,b}^+ = \max_{\substack{\tilde{a} \circ x = a \\ x \in \bar{C}}} \begin{cases} d_{\tilde{a}, b}^+ + \tilde{s}(x, ---) \\ d_{\tilde{a}, b} + \alpha + \tilde{s}(x, ---) \\ d_{\tilde{a}, b}^- + \alpha + \tilde{s}(x, ---) \end{cases} \quad (6d)$$

The recursion can be solved in the usual way with dynamic programming and therefore uses a matrix of size $\Theta(|a||b|)$ which is completed in the same time complexity.

For the number of comparisons, we see that the number of previous cells (a cell being a variable within a matrix element) for each element in the dynamic programming matrix for the above recursion is $3 \times 5 \times 5 = 75$ from (6b) plus $3 \times 5 = 15$ each from (6b) and (6d), giving a total of 105 precursors and thus 104 comparisons. However, we can make a significant speed-up by also introducing $\hat{d}_{a,b} = \max\{d_{a,b}, d_{a,b}^-, d_{a,b}^+\}$, since in equation (6b), this comparison will be made (implicitly) once for each reference to an element (a, b) . If this comparison is pre-computed, only 25 instead of 75 cells are considered in (6b), giving 24 comparisons. Also note that a similar comparison is done five times for each (a, b) in equations (6b) and (6d). If we pre-compute $\hat{d}_{a,b}^- = \max\{d_{a,b} + \alpha, d_{a,b}^-, d_{a,b}^+ + \alpha\}$ and $\hat{d}_{a,b}^+ = \max\{d_{a,b} + \alpha, d_{a,b}^-, d_{a,b}^+\}$, only four comparisons in each of the two cases are needed. If we also note that $\hat{d}_{a,b}$, $\hat{d}_{a,b}^-$, and $\hat{d}_{a,b}^+$ can be computed with five comparisons, the number of comparisons have been reduced to $5 + 24 + 4 + 4 = 37$.

4 On choosing a codon scoring function

A first approach to choosing s is to find the best “interpretation” of codon fragments. This is achieved by first mapping a codon fragment x to possible amino acids,

$$C(x) = \{x' : \exists y \in I(x), x'_i = y_i \text{ if } y_i \neq - \text{ and } x'_i \in \mathbf{N} \text{ otherwise.}\} \quad (7)$$

and then choosing

$$s(x, y) = \max_{\substack{x' \in C(x) \\ y' \in C(y)}} (s_A(aa(x', y')) + |3 - |x||\gamma + |3 - |y||\gamma). \quad (8)$$

In practice, it is probably desirable to extend C to work with ambiguity codes; The natural extension is to map a codon (fragment) to all possible interpretations of the ambiguity and choose the most favorable.

There are adjustments we could make to improve this scoring scheme and we now discuss a few suggestions.

4.1 Silent substitutions

If several alignments are possible that give the same score if they are translated to amino acid sequences, we still want to minimize the amount of nucleotide substitutions. That is, between two matches of codons equivalent with respect to amino acids, we choose the one with less substitutions. It therefore seems reasonable to add the cost for nucleotide mismatches in codons coding for the same amino acid. Let δ_s be the cost for nucleotide transitions and δ_v the cost for nucleotide transversions. (More elaborate scoring schemes could be considered to account for the different nucleotide substitution rates.) We adjust s to

$$s'(x, y) = \max_{\substack{x' \in C(x) \\ y' \in C(y)}} (s_A(aa(x', y')) + |3 - |x||\gamma + |3 - |y||\gamma + n_s\delta_s + n_v\delta_v) \quad (9)$$

for n_s and n_v being the number of silent transitions and transversions, respectively. An unfortunate effect with this is that we now have given up the nucleotide scoring function requirement from equation (3). However, in practice there is little reason to expect a big difference in the end-result.

If the requirement is imperative and special scoring of silent mutations is only to be used to distinguish equivalent (under requirement (3)) alignments, the following method is suggested:

1. Compute all alignments with the optimal score. This is easily done by adjusting the current algorithm to use a technique like e.g. what Chao [1] propose to compute all alignments within ϵ of the optimal value (in this application $\epsilon = 0$). The set of alignments is stored in a graph such that any path in the graph is an alignment.
2. In this graph, use dynamic programming to find the optimal alignment using an adjusted scoring function that penalize silent mutations as in equation (9) above.

Only alignments optimal under requirement (3) are computed in step 1, and the best alignment with respect to silent mutations is computed in step 2, and hence the method computes what we wanted.

4.2 Context-dependent frame-shift costs

One of the sources for frame-shifts are sequencing errors and it is well known [9] that such errors typically come from compression effects in the sequencing gel; for a longer stretch of the same nucleotide it is difficult to correctly determine how many bases we have. Thus, the likelihood of a frame-shift is different in different positions in a sequence. The propensity for a frame-shift to occur in nature might also vary over the sequences, which would be nice to account for if it could be quantified. To that purpose we add *context-dependent frame-shift costs*.

Let $\gamma(i, j, l)$ be the cost of putting a frame-shift of length l (possibly zero) between positions j and $j + 1$ of sequence i and let $l_{i,j} \in \{0, 1, 2\}$ give the length of a frame-shift after position j of sequence i . We may now write a new version of S_N .

$$S_N(a', b') = x\alpha + \sum_{j=0}^{|a|} \gamma(a, j, l_{a,j}) + \sum_{j=0}^{|b|} \gamma(b, j, l_{b,j}) + \sum_{i=0}^{|a'|/3} s(a'_{3i} a'_{3i+1} a'_{3i+2}, b'_{3i} b'_{3i+1} b'_{3i+2}) \quad (10)$$

Letting the cost of a frame-shift vary across the sequence has the effect that the choice of codon-fragment interpretation, as given in equations (8) and (9), is no longer valid. This is rectified by pre-computing the codon scoring function for each possible pair of contexts, either by looking at each pair of positions in the two sequences or by simply enumerating possible pair of contexts from knowing which contexts can occur (the latter method is then sequence independent).

4.3 Out-of-frame gaps

So far we have assumed that gaps occur in the reading-frame. If they were not in frame, frame-shifts would have to be inserted in both ends of the gap to compensate and this is probably quite costsome since frame-shifts should be expensive in order to produce reasonable alignments.

Out-of-frame gaps has been observed in HIV [9] and they should not be unexpected since they do not have a much larger impact on the interpretation than in-frame gaps. In particular, an out-of-frame insertion basically exchange one codon c for an inserted string of codons with the end-codons biased by the remaining, split, fragments of c . An out-of-frame deletion may be seen as a deletion followed by an insertion of one codon, because the two affected codons at the ends of the deletion join.

There are two out-of-frame gaps to be distinguished and we call them ± 1 and ± 2 , with the sign telling us in which sequence they occurred. If a gap is -1 , then the gap is in the first sequence delayed with one nucleotide with respect to the reading frame. A type $+2$ gap occurs in the second sequence and is delayed by two nucleotides, see also figure 1. Note that we can consider in-frame gaps to be of type ± 0 .


```

...xxx|x--|---|-xx... ..xxx|xxx|xxx|xxx...
...xxx|xxx|xxx|xxx... ..xxx|xx|---|--x...

```

Fig. 1. *The two types of out-of-frame gaps. To the left, a type -1 gap, while on the right a type $+2$ gap.*

As with the in-frame gaps, this is solved by the technique Gotoh suggested [3]. For each gap we introduced a variable, d^{-1} , d^{-2} , d^{+1} and d^{+2} , that contains the cost of ending the alignment with a gap of the respective type. We omit the details since they are more lengthy than informative.

4.4 Starting and ending with frame-shifts

It is often desirable to let alignments start and end with gaps without any cost, a so called end-free alignment. This is convenient, for instance, if the sequences have been unevenly sequenced. For the same reason, it is interesting to allow end-free frame-shifts. If we for some reason don't have the sequences starting in the correct reading frame, we don't want any strange behavior with insertion of frame-shifts to try to compensate. Instead, any frame-shifts in the beginning or end of an alignment should be for free.

This is easily achieved by using context-dependent frame-shift costs and choosing γ appropriately for the starting and ending positions.

5 Experimental results

We have implemented a simple version of the described algorithm, only considering lost nucleotides. s is chosen in the most direct way, based on a user-chosen amino acid scoring-scheme (e.g. PAM- or BLOSSUM-matrices [2, 6]). No concern about the improvements to the scoring scheme mentioned in section 4 is given, but ambiguity codes are honored.

HIV data from the ENV-V3 region, 13 sequences kindly supplied by Thomas Leitner [9], was used to test the robustness of the algorithm. Aligning any two of these sequences yields an alignment that is about 261 bases long. About 72 % of the columns are mismatches and about 9 columns contains indels.

Four tests was set up by randomly introducing errors in pairs of sequences and the algorithms ability to find the position of the removed nucleotide(s) was then tested. The codon scoring function s was based on PAM-100 [2], gap-cost and gap extension-cost was set to 20, and single-indel frame-shifts cost 40 while double-indel frame-shifts cost 50. The results are shown in table 5. As seen, the guessed position of the frame-shift was usually only a few bases from the correct position.

Table 1. The average displacement, i.e. the number average of bases away from the known position of the removed nucleotide(s) that the algorithm put a frameshift. The four tests was set up by removing one nucleotide or two adjacent nucleotides from one or both sequences. The positions of the removed nucleotides was chosen at random for each pair sequences. In one case, having deleted a pair of nucleotides in both sequences, the algorithm chose to insert two single indel frame-shifts.

Displacement	Removed nucleotides	
	1	2
In one sequence	3.2	5.8
In both sequences	2.4	5.7

6 Discussion

As has been shown, our algorithm is very robust. However, the price paid is in computation time, with 37 comparisons for each element in in the dynamic programming matrix (with none of the improvements from section 4).

As presented, the algorithm has a quadratic space need. A linear space approach, similar to Hirschberg's technique [7] (popularized by [10]), is straightforward to use also on this algorithm.

In the above, we have used amino acid scoring schemes to base our nucleotide scoring function on for the simple reason that they are well understood and well investigated. But instead of relying on statistics on amino acid substitution probabilities, we could make the same statistics on codons. Basically, this would imply using a 64 by 64 scoring matrix instead of a 23 by 23 matrix. It is noteworthy that codons have different frequencies, which further justifies this approach.

7 Acknowledgments

Thanks to John Kececioğlu for helping me understand the problem. Also thanks to Bill Bruno, Aaron Halpern, Johan Håstad and David Torney for discussions and suggestions.

References

1. K.-M. Chao. Computing all suboptimal alignments in linear space. In *5th Symposium on Combinatorial Pattern Matching*, pages 31–42. Springer-Verlag LNCS 807, 1994.
2. M. O. Dayhoff, R. M. Schwartz, and B. C. Orcott. A model of evolutionary change in proteins. *Atlas of Protein Sequence and Structure*, 5:345–352, 1978. National Biomedical Research Foundation, Silver Spring, Maryland, USA.

3. O. Gotoh. An improved algorithm for matching biological sequences. *Journal of Molecular Biology*, 162:705–708, 1982.
4. X. Guan and E. C. Uberbacher. Alignments of DNA and protein sequences containing frameshift errors. *Comp. Appl. Bio. Sci.*, 12(1):31–40, 1996.
5. J. Hein. An algorithm combining DNA and protein alignment. *Journal of Theoretical Biology*, 167:169–174, 1994.
6. S. Henikoff and J. G. Henikoff. Amino acid substitution matrices from protein blocks. *Proc. Natl. Acad. Sci.*, 89:10915–10919, 1992.
7. D. S. Hirschberg. A linear space algorithm for computing longest common subsequences. *Communications of the ACM*, 18:341–343, 1975.
8. L. J. Knecht. *Alignment and Analysis of Genes Coding for Proteins*. PhD thesis, Swiss Federal Institute of Technology, 1996.
9. T. Leitner. Personal communication. Until recently at the Department of Biochemistry, Royal Institute of Technology, Stockholm, now at Los Alamos National Laboratory, USA, Theoretical Biology and Biophysics Group.
10. E. W. Myers and W. Miller. Optimal alignments in linear space. *Comp. Appl. Bio. Sci.*, 4(1):11–17, 1988.
11. H. Peltola, H. Söderlund, and E. Ukkonen. Algorithms for the search of amino acid patterns in nucleic acid sequences. *Nuclear Acids Research*, 14(1):99–107, 1986.
12. D. Sankoff and J. Kruskal. *Time warps, string edits, and macromolecules: The theory and practice of sequence comparison*. Addison-Wesley, 1983.
13. P. H. Sellers. On the theory and computation of evolutionary distances. *SIAM Journal on Applied Mathematics*, 26:787, 1974.
14. D. J. States and D. Botstein. Molecular sequence accuracy and the analysis of protein coding regions. *Proc. Natl. Acad. Sci.*, 88:5518–5522, July 1991.
15. M. S. Waterman. *Introduction to computational biology: Maps, sequences and genomes*. Chapman & Hall, 1995.